System Verification Via Generic Games

Behavioural Equivalence and Model Checking Games

Von der Fakultät für Ingenieurwissenschaften,
Abteilung Informatik und Angewandte Kognitionswissenschaften der
Universität Duisburg-Essen

zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften
(Dr. rer. nat.)

genehmigte Dissertation

von
Christina Mika-Michalski
aus
Zabrze

1. Gutachterin: Prof. Dr. Barbara König
2. Gutachter : Prof. Dr. Paolo Baldan
Tag der mündlichen Prüfung: 27. Juli 2021

DOI: 10.17185/duepublico/75302
URN: urn:nbn:de:hbz:464-20220201-070334-2

System Verification Via Generic Games - Behavioural Equivalence and Model Checking Games

This thesis is based on the following original publications:

▷ Chapter 3,4 & 5:

[KM17b] Barbara König and Christina Mika. "Bisimulation Games on Coalgebras*". In: *CALCO Early Ideas '17*. 2017.

[KM18] Barbara König and Christina Mika-Michalski. "(Metric) Bisimulation Games and Real-Valued Modal Logics for Coalgebras". In: *29th International Conference on Concurrency Theory (CONCUR 2018)*. Ed. by S. Schewe and L. Zhang. Vol. 118. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2018, 37:1–37:17. ISBN: 978-3-95977-087-3. DOI: 10.4230/LIPIcs.CONCUR.2018.37.

[KMS20b] Barbara König, Christina Mika-Michalski, and Lutz Schröder. "Explaining Non-Bisimilarity in a Coalgebraic Approach: Games and Distinguishing Formulas". In: *Coalgebraic Methods in Computer Science - 15th International Workshop, CMCS 2020, Colocated with ETAPS 2020, Proceedings*. Ed. by Daniela Petrisan and Jurriaan Rot. Vol. 12094. Lecture Notes in Computer Science. Springer, 2020, pp. 133–154. DOI: 10.1007/978-3-030-57201-3_8.

▷ Chapter 6:

[BK+19b] Harsh Beohar, Barbara König, Sebastian Küpper, and Christina Mika-Michalski. "Coalgebraic Games in Kleisli Categories". In: *CALCO Early Ideas '19*. 2019.

[BK+20] Harsh Beohar, Barbara König, Sebastian Küpper, and Christina Mika-Michalski. *Coalgebraic modal logic and games: an indexed category framework*. unpublished. 2020.

▷ Chapter 7:

[BK+19a] Paolo Baldan, Barbara König, Christina Mika-Michalski, and Tommaso Padoan. "Fixpoint Games on Continuous Lattices". In: *Proc. ACM Symposium on Principles of Programming Languages (POPL)* 3 (2019), 26:1–26:29. DOI: 10.1145/3290339.

Abstract

Verification *should be* a major building block in any complex development process. In general, *software systems* grow over the years and become almost unmanageable over time. Therefore, theoretical modelling and verification techniques have become an integral part in the computer science community.

In the last decades, the question whether an implementation meets the *specification* serves as motivation to focus on *model checking* or *logical equivalence* where requirements are formulated via logical formulas.

In order to enable these verification methods, state-based systems and notions of *behavioural equivalence* or *distance* provide suitable models and techniques to analyze and verify program code. At this point, *coalgebra* – a concept of *category theory* – enters the scene. *Coalgebra* enables the modelling of different state-based systems including branching types such as *non-determinism*, *determinism*, or *probabilistic systems*. Furthermore, the coalgebraic framework provides an abstract definition of *modalities* which are used to construct formulas. Therefore, the study of verification problems from a *coalgebraic* (*i.e. abstract*) point of view enables the development of *generic* algorithms.

Another focus in this thesis is given in terms of *games* which provide alternative semantics with respect to behavioural notions such as *bisimulation* or *language equivalence* or in terms of *model checking*.

Therefore, this dissertation aims at contributing to the research area of *logic* and *games* via generic frameworks given by *category theory* or *lattice theory*.

Acknowledgment

In my case, I have to emphasize that words can not express how thankful I am for the last five years. Especially, I can not find words for the enthusiasm and patience of my supervisor *Barbara König*. Briefly said: the appreciation that *Barbara* shows when dealing with teaching, research, or fellow human beings has influenced me positively.

Next, I want to thank Sebastian Küpper for pushing me into the *right* direction and all the excellent discussions before, after, and during the time where we share one office. Another memorable experience for me was my research visit in Italy, where we met *Paolo Baldan* and *Tommaso Padoan*. The collaboration in research was a great time. The fish served by *Paolo* and his family also remains unforgettable.

My husband *Michael* is also a great cook and has done an incredibly fantastic job all along. You managed all my stress-eating attacks and much more.

There are many other people I would like to thank, but as I said above, I lack the words. Therefore, I stick to the formal notation and define the following acknowledgment *morphism* $\mathcal{A}$. Please, interpret the terms as you feel:

$$\mathcal{A} : \textit{people around me} \to \{\infty^{\bigstar}, \infty^{\heartsuit}, \infty, \ominus, \ominus^{\heartsuit}, \ominus_{\infty}\}$$

$$
\begin{array}{ll}
x \,\mathcal{A}\, \infty^{\bigstar} & \textit{if } x = \textit{Barbara König} \\
x \,\mathcal{A}\, \infty^{\heartsuit} & \textit{if } x = \textit{My Michael Michalski} \\
x \,\mathcal{A}\, (\infty + 1) & \textit{if } x = \textit{Sebastian Küpper} \\
x \,\mathcal{A}\, \infty & \textit{if } x \in \{\textit{Paolo Baldan}, \textit{Tommaso Padoan}, \textit{Harsh Beohar}\} \\
 & \quad \cup \{\textit{Thorsten Wißmann}, \textit{Lutz Schröder}\} \\
x \,\mathcal{A}\, \ominus_{\infty} & \textit{if } x \in \{\textit{Rebecca Bernemann}, \textit{Benjamin Cabrera}, \\
 & \quad \textit{Henning Kerstan}, \textit{Dennis Nolte}, \textit{Lars Stoltenow}\} \\
x \,\mathcal{A}\, \ominus^{\heartsuit} & \textit{if } x \in \textit{MyFamily} \cup \textit{MyFriends} \\
x \,\mathcal{A}\, \ominus_{\infty} & \textit{if } x \in \textit{Eifel} - \textit{Connection} \\
x \,\mathcal{A}\, \ominus & \textit{if } x \in \textit{Voigtländer \& König} - \textit{Group Seminar} \\
x \,\mathcal{A}\, \ominus & \textit{if } x \in \textit{CALCO or DCON} - \textit{Community}
\end{array}
$$

Moreover, I enjoyed the work with all my colleges. The atmosphere in our group seminar and at every conference was great (and not just because of the cakes). Our research community really facilitates the introduction, and as a *young researcher*, I always felt welcome.

Z najgłębszą wdzięcznością:

Maria i Ludwik Włodarczyk, Katarzyna i Antoni Mika, Piotr i Valerie,

Jolanta i Tomasz Michalski,

&

Meike Müller i Markus Grosche

Contents

1 Introduction

The word *game* has different meanings depending on the language or culture, but most of these interpretations have one association in common: joy and pleasure. Thus, it is not a big surprise that games have already been recognized as a didactic concept for children in ancient times, which can be found on the one hand in Platos and the works of Aristotle [Van13]. And on the other hand, today, games for didactic purposes at school or higher education are widely studied (cf. [Van13; May07; Mar19]).

Moreover, games provide an alternative representation of economic processes (i.e. *Nash equilibrium*), social interaction [Pau01], or truth in a model of a first order formula [Hin68].

Therefore, it is decisive enough for us to consider *system verification* from a *game-theoretical* perspective.

1.1 Motivation

Any multiplayer game is determined by a set of rules that tells the players how they have to interact with each other. Therefore, such playing rules can be seen as a recipe that offers an intuitive way to model interactions. To analyze the observable behaviour of software systems, any interaction between individual system components or between a system and its environment (e.g. a user providing some input) plays a significant role. Therefore, the thesis focuses on the concept of modelling interactions between software components using game rules, which has already been investigated beforehand [Sti99; DLT08; Bal99].

However, before we move to the verification of programs via games, we need to take care of the right modelling techniques of software systems. Since we are interested in program states and their possible interactions, we consider so-called *labelled transition systems* (LTS). These state-based models have proven their reliability as suitable representations for software systems, where labelled transitions represent actions between the states. A state of such a graph-based representation can be seen as an abstract interpretation of a system state [San11].

Games for LTS: The advantages of comparing the behaviour of two states in terms of two-player games has already been discussed in [Sti99; San11] which introduce a *game-theoretical* characterization of *bisimulation* for labelled transition systems. *Bisimulation* is a notion of *behaviour equivalence* which is weaker than graph isomorphism but stronger than *language equivalence* (see [Gla01]). Informally, one could say that two states in an LTS are bisimilar if they can mutually simulate their actions where after a successful imitation the *spoiler* can change the site. In this way, *bisimulation* takes care of the non-deterministic branching in LTS where two states are *language equivalent* if both only enable the same set of words.

For any LTS, the game's initial situation is given by a state pair (x, y). One player is called *Spoiler* (**S**) and in the first step of the game he chooses one of the states e.g. x and determines a transition $x \xrightarrow{a} x'$ (if available). Since in the second step of the game, the opponent has to match the move dictated by the *Spoiler* from the other state e.g. y, she is known as the *Duplicator* (**D**). If no transition $y \xrightarrow{a} y'$ exists the game terminates and **S** is winning. Otherwise, the game proceeds in a new round given by (x', y'), where **S** can independently from his previous choice switch between x' and y'. If the game never terminates because **D** can mimic all the moves by **S**, she is the winner. Additionally, **D** is winning in the first step if neither of the states has an outgoing transition.

Intuitively, the condition described in the second step guarantees that **D** has a winning strategy iff the state pair admits the same behaviour with respect to the actions and the branching of the system, which can be non-deterministic. Besides, the possibility to choose between the two states in the first step enables a winning strategy for **S**. More precisely, either the state pairs are distinguishable by at least one labelled transition, or for one state there exists after some path p a reachable state z so that for all matching paths p the other state has no successor state z', which matches all the actions of z. Therefore, the strategy for **S** is to follow p such that it leads to z, which is enabled by the option to switch the side in step one of the game. It depends on p, how often **S** has to switch the side.

But strictly speaking, the origin of *bisimulation games* lies in the realm of logic analogously to the notion of *bisimulation* [San09]. It is slightly difficult to say who exactly is the pioneer in linking logic and games since some people refer to the rules of debating by Aristotle's and "The Games of Logic" by Lewis Caroll in 1887 also serves as an early example for teaching logic via games [HV19].

The Link between Logic and Games: In terms of first order logic the work of Henkin [Hen61] addresses game-theoretical semantics for formulas with

infinitely many quantifier alternations. Henkin recognized that choosing objects of the underlying domain – which refers to Tarski's definition of truth – enables an interpretation via moves in a two-player game. One player is the $\forall$-player who has to choose objects for variables bound by the universal quantifier. And the positions for the other player, commonly known as the $\exists$-player, are determined by the variables bound by the existential quantifier. Henkin observed that the $\exists$-player has a winning-strategy if a sequence of Skolem functions exists.

Later, Hintikka [Hin68] extended the game approach of Henkin to conjunction, disjunction and negation. A quite important fact is, that Hintikka discovered the intuition of negation in terms of games, namely that it causes the switch of roles between the two players (see [HV19], [MPT09, Introduction] for more details).

If we move from Hintikka's game over a single first order logic formula to the question if two models are distinguishable, we arrive at the *Ehrenfeucht-Fraïssé games*. These games are based on Fraïssé's theory which was developed for Tarski's idea to define the equivalence of two structures in such a way that the sentences which are true in one structure also hold in the second [HV19]. Moreover, *Ehrenfeucht-Fraïssé games* are used in the proof of the Van Benthem/Rosen theorem stating that propositional modal logic is the bisimulation invariant fragment of first order logic [Ott04].

Returning to the verification of program code, the previously described *bisimulation games* over relational models (LTS) provide a game-based technique to decide if a piece of code satisfies the requirements under the assumption that the systems are finitely branching (i.e. no state has infinitely many outgoing transitions). Therefore, one can model the requirements and the implementation via two labelled transition systems and decide if such structures are *bisimilar* or one can say equivalently that they satisfy the same set of modal logic formulas. This one-to-one correspondence between *logical equivalence* and *bisimulation* for finitely branching LTS is known under the name of *Hennessy-Milner* theorem [HM80] where the normal semantic for *modal logic* is known as relational semantics, which include the $\Box$ and $\Diamond$-operators. Intuitively, one can associate

$$\Box \text{ with necessity and } \Diamond \text{ with possibility.}$$

Furthermore, any such modal formula can be converted into a predicate logic formula of first order logic [BB07]. This means, that the core of the bisimulation game is given by the comparison of two models (i.e. structures) and therefore *Ehrenfeucht-Fraïssé games* can be seen as a generalization of the bisimulation games [Sti99; GO14].

A Generic Perspective: Besides logic and games, the second dimension of this thesis is *generalization* and since state-based systems serve as a suitable modelling framework for software [San11], our focus moves to "Universal coalgebra" [Rut00] by Rutten. This coalgebraic concept offers a universal way to handle different transition systems such as deterministic, non-deterministic, or probabilistic branching.

More concretely, pure non-determinism as for unlabelled transition systems corresponds to a mapping $\alpha : X \to \mathcal{P}X$ where X represents the state space and in case a state x has two outgoing transitions $x \to y$, $x \to z$ we write $\alpha(x) = \{y, z\} \in \mathcal{P}X$ (see left system of Figure 1.1).

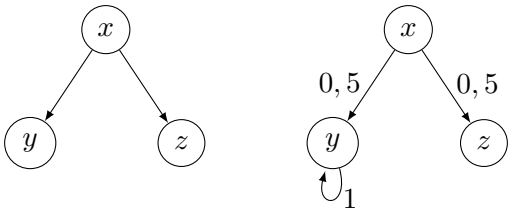


Figure 1.1: Two different systems over $X = \{x, y, z\}$: a non-deterministic system on the left. A probabilistic system with termination on the right.

A probabilistic branching with termination can be interpreted as a mapping $\beta : X \to \mathcal{D}X + 1$ where $\mathcal{D}X$ denotes the set of all probability functions over X with finite support (i.e. for each $p \in \mathcal{D}X$ there are finitely many $x \in X$ with $p(x) > 0$). We use $+$ to denote disjunct union and a state maps to $1 = \{\bullet\}$ if no transitions are available. In the right system of Figure 1.1 state x is mapped to the probability distribution $p : X \to [0, 1]$ with $p(y) = p(z) = 0.5,\ p(x) = 0$ and $\beta(z) = \bullet$.

In addition, even structures such as *neighbourhood frames*, where a state $x \in X$ can be mapped to different subsets of X, fall into this framework. Such systems are used as semantical structures for *non-normal modal logic* and are modeled via functions of type $X \to \mathcal{Q}\mathcal{Q}X$ (where $\mathcal{Q}$ maps a set to its powerset and a function f to the inverse image $f^{-1}[\_]$) [HK04].

Intuitively, the terms generic and coalgebraic can be used interchangeably in the scope of state-based systems since components as $\mathcal{P}, \mathcal{D} + 1$ or $\mathcal{Q}\mathcal{Q}$ of the coalgebras introduced above are so-called *functors*. The categorical concept of a *functor* is a morphism which maps between (different) mathematical structures such that certain properties of the source structure are preserved. Therefore, in general we write $\alpha : X \to FX$ where F is a placeholder for the branching type of the system.

Moreover, the theory of coalgebras allows us to generalize the notion of behavioural equivalence [Rut00] in a very elegant way, which in turn enables generic algorithms

to compute behaviour equivalences. These are known under *partition refinement algorithms* and are extensively studied in the coalgebraic framework [KK18; DM+17; WD+18]. A concrete instantiation of such an algorithm is the minimization technique for deterministic finite automata (DFA) by Hopcroft [Hop71].

Furthermore, *coalgebraic modal logic* also generalizes modal logics since modalities correspond to so-called *predicate liftings* [Pat04; Sch08] first introduced by Hermida and Jacobs [HJ98] and [Röß00; Jac01]. A *predicate lifting* is a *natural* mechanism which maps a subset of a state space X to a subset of FX where the term natural means, that it is a structure preserving map between *functors*.

As mentioned above, *modal logics* have already been studied from the categorical perspective, but work on *coalgebraic games* is rare. We are mainly aware of Baltag's game where the interaction between the two players is based on bisimulation relations [Bal99] and these games are more reminiscent of Moss' coalgebraic logics [Mos99]. We are interested in a coalgebraic game characterization which directly refers to the concept of *coalgebraic modal logics* via *predicate liftings* [Pat04; Sch08]. One reason to consider predicate liftings is given by an automatism to derive distinguishing formulas for non-equivalent states as worked out for LTS presented by Cleaveland in [Cle90]. This algorithm is based on partition refinement techniques and includes the $\Diamond$-operator. Distinguishing formulas provide an explanation for the non-bisimilarity of two states and we are not aware of a generalized version of Cleaveland's ideas [Cle90].

A Quantitative Perspective: In the third dimension of this thesis we refrain from classifying systems or system states as either equivalent or non-equivalent, which is often too strict, but rather measure their behavioural distance. This makes sense in probabilistic systems, systems with time or real-valued output. For instance, we might obtain the result, that the running times of two systems differ by 10 seconds, which might be acceptable in some scenarios (departure of a train), but inacceptable in others (delay of a vending machine). On the other hand, two states are behaviourally equivalent in the classical sense if and only if they have distance 0. Such notions are for instance useful in the area of conformance testing [KM15] and differential privacy [CG+14].

Behavioural metrics have been studied in different variants, for instance in probabilistic settings [Des99; DG+04; CGT16] as well as in the setting of metric transition systems [AFS09; FLT11], which are non-deterministic transition systems with quantitative information. The groundwork for the treatment of coalgebras in metric spaces was laid by Turi and Rutten [TR98].

Again, work on quantitative games is rather rare where the work in [DLT08] presents a probabilistic game but pairs it with a qualitative logic. Besides a Van Benthem theorem for fuzzy logic, the work in [WS+18a] introduces a game over probabilistic systems which is based on the Wasserstein lifting. There are two standard approaches to lift a metric over a state space X to a metric over the set of probabilistic functions (i.e. $\mathcal{D}X + 1$): the Wasserstein lifting uses couplings and the *Kantorovich lifting* is based on *non-expansive* functions. A generalization of both concepts in terms of category theory is studied in [BB+14].

We are only aware of quantitative coalgebraic games presented in [KK+19] which does not treat *coalgebraic modal logics* at all. To our knowledge, it is a new contribution to use the results in [BB+14] to obtain a quantitative version of the *Hennessy-Milner* theorem equipped with a metric game and quantitative logic based on the *Kantorovich lifting*.

Bisimulation vs. Model Checking: On some occasions one is rather interested whether a state satisfies a certain specification than analyzing the whole state space using *bisimulation* or *behavioural distances* [FV99]. Therefore another approach to reason about the behaviour of system states is given by *model checking* via parity games [Jur00; BW18; Sti95; EJ91] originating from the work in [Koz83] for the *modal μ-calculus*.

In *model checking* an implementation is provided via a transition system as for *bisimulation*. The specification is given in terms of a *modal μ-calculus* formula φ and the goal is to verify if the model satisfies φ.

The *modal μ-calculus* is an extension of *propositional modal logic* to the greatest and least fixpoint operators, denoted respectively ν and μ. Such operators are used in *linear temporal logic (LTL)* and *computational tree logic (CTL)* which are basic languages for verifying *linear temporal* and *branching* properties (see [BK08]). The usage of greatest and smallest fixpoints enable more expressive statements as for example "infinilty often p on some path" [BW18] where p is some proposition.

In analogy to Henkin's and Hintikka's game-theoretical approach, parity games include an existential $\exists$ and universal player $\forall$, called respectively *Eve* and *Adam*. The game board is derived from the μ-calculus formula with respect to the underlying model where each position is equipped with a *rank* based on the alternation depths of the fixpoint operators [BW18]. The rules for conjunction, disjunction, and negation including the basic winning conditions refer to the concepts of Hintikka for first order logic. The winning conditions due to the fixpoint operators μ, ν are defined in terms of parity conditions. More precisely, in case of an infinite game sequence *Eve*

is winning if the highest rank occurring infinitely often is even. Otherwise, *Adam* is the winner of such an infinite game [BW18; EJ91].

Since it is quite hard to interpret such formulas, an alternative representation is given by systems of fixpoint equations [HSC16] based on the equivalence between a formula and its equational form [CKS92; Sei96]. Results to compute winning strategies for *parity games* in terms of so-called *progress measures* are already studied [Jur00; HSC16], but we are not aware of a game version for a wide range of lattices.

1.2 Structure, Contributions and Publications

First, the structure of my thesis is described via an overview in Figure 1.2. Afterwards, the contributions and corresponding publications of this thesis are listed and an overview of the *self-citations* in analogy to the structure of the thesis is illustrated in Figure 1.4.

The publications in this overview (Figure 1.4) are based on the joint work with my supervisor Barbara König and other researchers. In this thesis I reused the contents of our publications, which are the results of several iterative processes by all authors. Whenever I had the feeling, that some additional examples or explanations would improve the overall understanding of the chapter, I added examples, definitions or extended the clarifying text. At this point, I would like to point out that I have not included those parts of the papers [BK+20; BK+19a] that I have not actively worked on.

The material of the papers wrt. the chapters 3-5 is an outcome of a close cooperation with my supervisor. For the other two chapters (6,7) I was mainly contributing on the game-theoretical parts in [BK+20; BK+19a].

In the spirit of "Abstract and Concrete Categories - The Joy of Cats" [AHS09] by Jirí Adámek I use a similar metaphor "Joy with Cats" to emphasize that our work is mainly based on category theory.

My thesis begins with this introductory Chapter 1 including the motivation and an additional section, which describes how the contents are structured and gives an overview of the related publications.

Mathematical Foundations

Chapter 2, which splits into three parts, introduces the mathematical foundations needed for our work. Section 2.1 includes general mathematical notation, some set theory, algebra and standard fixpoint theory for applications in software verifica-

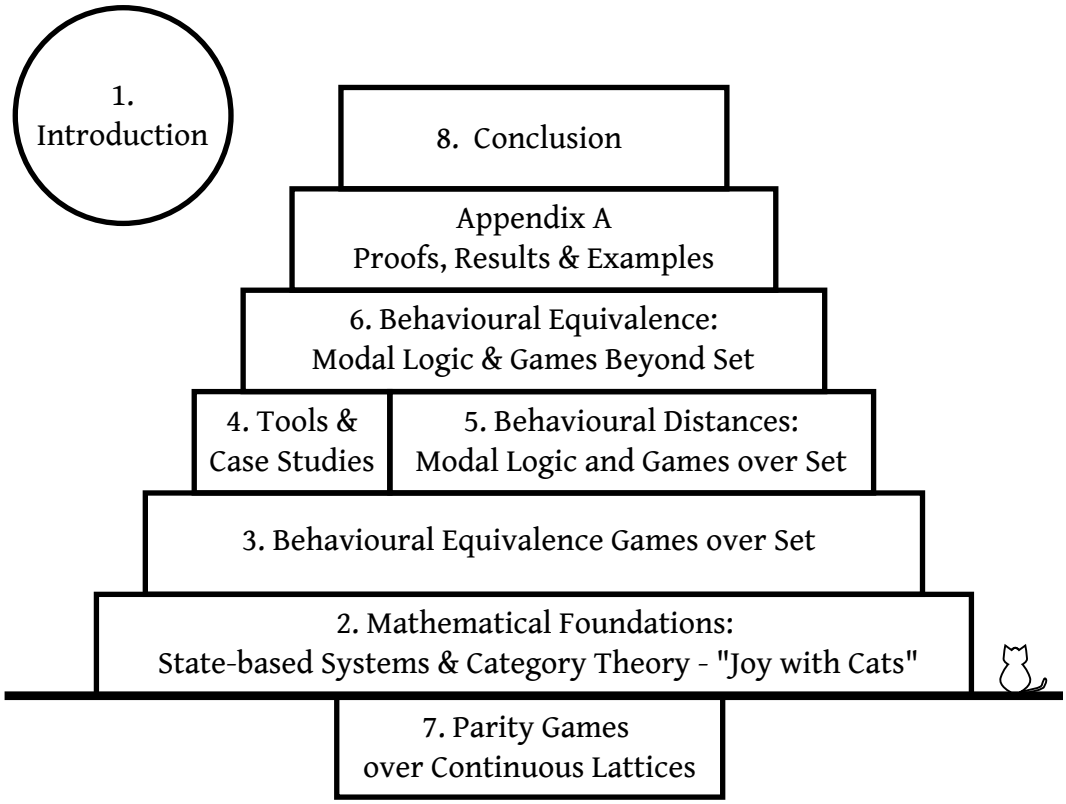


Figure 1.2: Relation between the chapters. All chapters are based on the mathematical foundations. The work presented in Chapters 3-6 and the appendix build on each other. The content in Chapter 7 is a stand-alone chapter i.e. an additional treasure hidden in the burial chamber. The conclusion sums everything up and the introduction brings clarity into the whole material.

tion. Section 2.2 is dedicated to transitions systems and the notion of behavioural equivalence. The last Section 2.3 splits into five subsections where the first three parts refer to basics of category theory as *classes*, *functors*, *natural transformations*, and *coalgebras*. The last two subsections can be seen as an advanced course where Section 2.3.4 treats *adjunctions* and *monads*, followed by *indexed categories* and *fibrations* presented in Section 2.3.5.

Contributions

After formally introducing the *Hennessy-Milner* theorem for LTS and the foundations of *coalgebraic modal logic* over the *category* **Set** – which is a standard setting dealing with sets and functions – in Chapter 3, Section 3.3 presents a *coalgebraic game* that is directly based on *coalgebraic modal logic* via *predicate liftings* (cf. Figure 1.3).

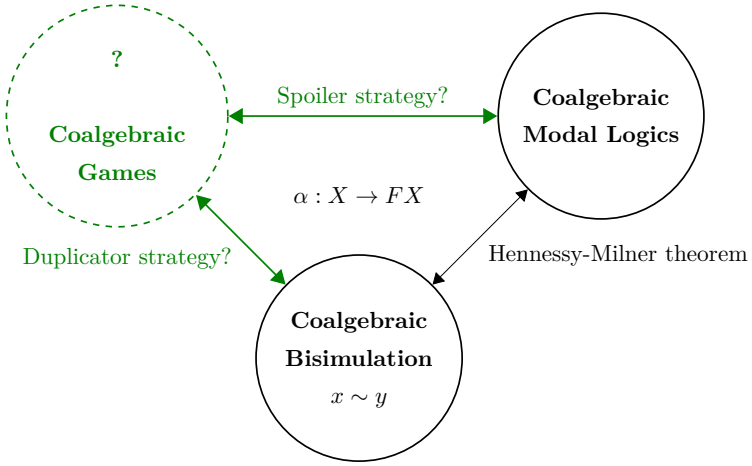


Figure 1.3: The classical coalgebraic triad: bisimulation, modal logic, and games over **Set**.

Section 3.4 first concludes with a new *categorical* understanding of the different partition refinement techniques based on the approach in [DM+17]. The second part of Section 3.4 introduces a generalized version of the automatic explanation for non-bisimilar states inspired by Cleaveland's work [Cle90]. In addition, a prototype implementation of our theoretical techniques called T-Beg (*Tool for Behavioural Equivalence Games*) is presented in Chapter 4 [KMS20b; KM18].

In Chapter 5 the focus lies on quantitative system verification where first the necessary preliminaries and some motivating examples (see Section 5.2) are provided. Afterwards, Section 5.3 presents a quantitative modal logic and a quantitative version of the *Hennessy-Milner* theorem. Finally, in Section 5.4 the corresponding metric game [KM18] is explained.

Chapter 6 comes up with *indexed categories* to study an approach of generalized and game-theoretical characterizations of behavioural notions, which works also beyond **Set** [BK+19b; BK+20].

The last contributions are given in Chapter 7, presenting our parity games over *continuous lattices* and the equational form of fixpoint formulas [BK+19a].

Besides some proofs, the Appendix A includes some additional results and examples which mainly refer to the contents of Chapter 3. In Appendix A.4 some of the proofs referring to Chapter 6 are given.

Publications

Finally, the publications this thesis is based on are listed. In order to indicate the corresponding chapters I refer to the number of the chapter in Figure 1.4.

Note that I was primarily working on the material of the publications presented in the pyramid body. Therefore, Chapter 7 mainly focuses on the parts in [BK+19a] that I have been involved in.

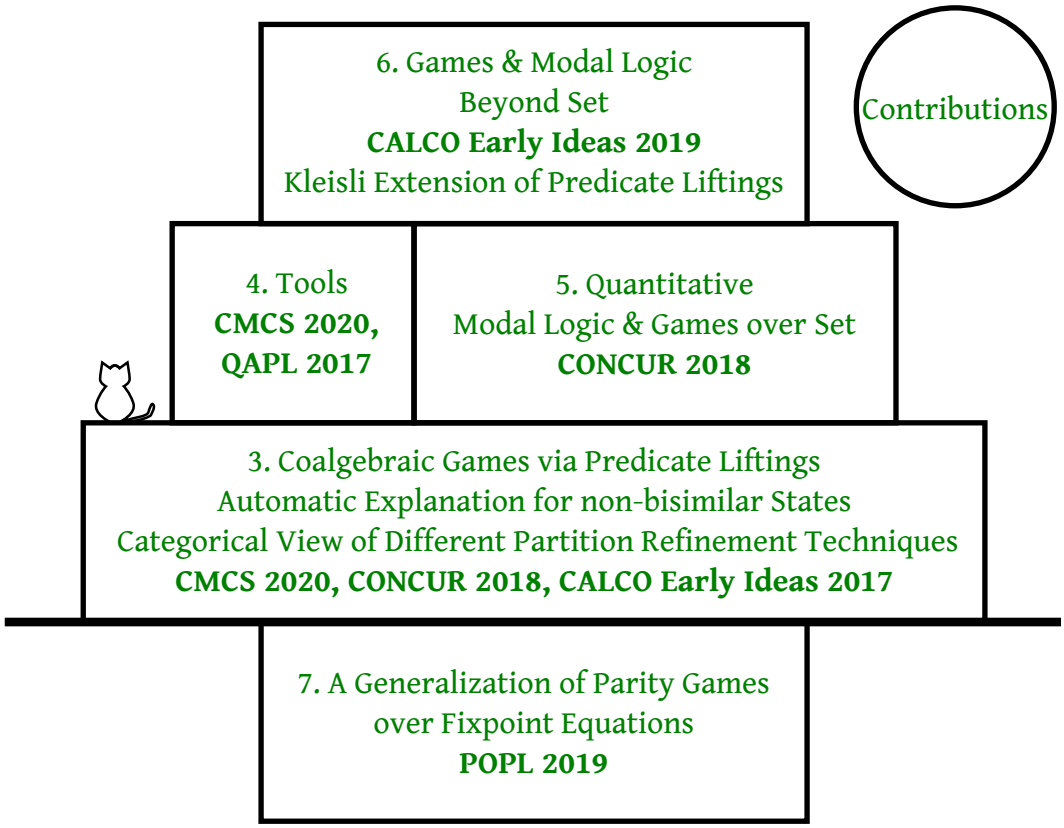


Figure 1.4: Publications related to the contributions referring to the corresponding chapter.

- CALCO Early Ideas '17: Barbara König and Christina Mika, Bisimulation Games on Coalgebras*, 2017

- 29th International Conference on Concurrency Theory (CONCUR 2018): Barbara König and Christina Mika-Michalski, (Metric) Bisimulation Games and Real-Valued Modal Logics for Coalgebras, Leibniz International Proceedings in Informatics (LIPIcs), 2018, 10.4230/LIPIcs.CONCUR.2018.37

- Coalgebraic Methods in Computer Science - 15th International Workshop, CMCS 2020, Colocated with ETAPS 2020, Proceedings: Barbara König, Christina Mika-Michalski, and Lutz Schröder, Explaining Non-Bisimilarity

in a Coalgebraic Approach: Games and Distinguishing Formulas, Lecture Notes in Computer Science, 2020, Springer, 10.1007/978-3-030-57201-3_8

- CALCO Early Ideas '19: Harsh Beohar, Barbara König, Sebastian Küpper, and Christina Mika-Michalski, Coalgebraic Games in Kleisli Categories, 2019
- unpublished (cf. [BK+21]): Harsh Beohar, Barbara König, Sebastian Küpper, and Christina Mika-Michalski, Coalgebraic modal logic and games: an indexed category framework, 2020
- Proc. ACM, Symposium on Principles of Programming Languages (POPL): Paolo Baldan, Barbara König, Christina Mika-Michalski, and Tommaso Padoan, Fixpoint Games on Continuous Lattices, 2019, 10.1145/3290339

2 Mathematical Foundations

The basis of this thesis is formed by parts of *category theory* and various model-theoretical concepts summarized under the term *state-based systems*. Therefore, this chapter introduces all relevant definitions and theorems that are required to present our results and contributions. Furthermore, additional building blocks are given by set theory as well as lattice theory.

The chapter starts with Section 2.1 which includes elementary notations, definitions, and theorems form set theory and universal algebra. Section 2.2 serves as an introduction to state-based systems and the characterization of system behaviour. The first half of the last Section 2.3 covers basic categorical definitions. In the second half, further concepts of categorical theory are introduced. Through all the sections most of the definitions and concepts are explained using standard examples.

2.1 Notations and Elementary Definitions

In this section we introduce some notations, elementary definitions and theorems. While most of the following symbols and definitions belong to the mathematical standards known from set theory and universal algebra, we refer to [DP02; GH+03] for the part dealing with lattice theory.

We use standard notations and definitions from set theory, e.g. we write $x \in X$ ($x \notin X$) in order to denote that x is (not) an element in X. We denote the set of natural numbers with $\mathbb{N}$, rational numbers with $\mathbb{Q}$ and the set of reals with $\mathbb{R}$ (where $0 \in \mathbb{R}_0$).

Given a binary relation $R \subseteq X \times Y$ we use xRy instead of $(x, y) \in R$ to indicate that x, y are related. For a non-empty set X we denote with id_X the identity relation over X. Furthermore, we consider relations on sets called *partial ordered sets* or *posets*.

Definition 2.1.1: Poset

A poset $\langle X, \sqsubseteq \rangle$ is a set endowed with a binary relation over X which represents an order with the following three properties:

1. *Reflexivity*: For all $x \in X$ we have $x \sqsubseteq x$.
2. *Antisymmetry*: If $x \sqsubseteq y$ and $y \sqsubseteq x$ then $x = y$ (where symmetry means that for all $x \sqsubseteq y$ it holds that $y \sqsubseteq x$).
3. *Transitivity*: If $x \sqsubseteq y$ and $y \sqsubseteq z$ then $x \sqsubseteq z$.

An order that is reflexive and transitive is called *preorder*. Other relations used within this thesis are given by reflexive, symmetric, and transitive relations also known under the term *equivalence relations*. Given such a relation $R \subseteq X \times X$, we use $[x]_R = \{z \in X \mid \exists\, (x, z) \text{ such that } (x, z) \in R\}$ to denote equivalence classes. In addition, if we factor a non-empty set X through an equivalence relation $R \subseteq X \times X$ we denote this with X/R i.e. $X/R = \{[x]_R \mid x \in X\}$.

A preordered or partially ordered set $(P, \sqsubseteq)$ is often denoted simply as P, omitting the (pre)order relation. It is *well-ordered* if every non-empty subset $X \subseteq P$ has a minimum. The *join* and the *meet* of a subset $X \subseteq P$ (if they exist) are denoted $\bigsqcup X$ and $\bigsqcap X$, respectively.

Now we are ready to consider special partially ordered sets, the so-called *lattices*, and some well-known theorems of lattice theory.

Definition 2.1.2: Lattice

A *lattice* $(L, \sqsubseteq)$ is a partially ordered set such that the $\bigsqcup$ and $\bigsqcap$ of any two-element subset $S \subseteq L$ exist in L.

This way we obtain two binary operations $\sqcap, \sqcup : L \times L \to L$ satisfying the following axioms:

1. *Associativity*: both operators $\sqcap, \sqcup$ are associative.
2. *Commutativity*: both operators $\sqcap, \sqcup$ are commutative.
3. *Absorption*: for all $a, b \in L$ we have $a \sqcap (a \sqcup b) = a$ and $a \sqcup (a \sqcap b) = a$

A *bounded lattice* additionally includes two elements $\top$ and $\bot$ where for all $l \in L$, $\bot \leq l$ and $l \leq \top$ hold and L satisfies the following two identity axioms:

$$x \sqcap \top = x \qquad x \sqcup \bot = x$$
$$x \sqcap \bot = \bot \qquad x \sqcup \top = \top$$

A *complemented lattice* is a bounded lattice where for each element $l \in L$ at least one complement $c \in L$ exists such that $l \sqcup c = \bot$ and $l \sqcap c = \top$.

A *complete lattice* is a partially ordered set $(L, \sqsubseteq)$ such that each subset $X \subseteq L$ admits a join $\bigsqcup X$ and a meet $\bigsqcap X$. A complete lattice $(L, \sqsubseteq)$ always has a least element $\bot = \bigsqcap L$ and a greatest element $\top = \bigsqcup L$.

A lattice is *completely distributive* if it is complete and for any family $\{l_{j,k} : j \in J, k \in K(j)\}$ of L we have

$$\bigsqcap_{j \in J} (\bigsqcup_{k \in K(j)} l_{j,k}) = \bigsqcup_{g \in M} (\bigsqcap_{j \in J} l_{j,g(j)})$$

where M is the set of choice functions over J with values $g(j) \in K(j)$ [GH+03].

A function $f \colon L \to L$ is *monotone* if for all $l, l' \in L$, if $l \sqsubseteq l'$ then $f(l) \sqsubseteq f(l')$. By Knaster-Tarski's theorem [Tar55], any monotone function on a complete lattice has a least and a greatest fixpoint, denoted respectively μf and νf, characterized as the infimum of all pre-fixpoints respectively the supremum of all post-fixpoints:

$$\mu f = \bigsqcap \{l \mid f(l) \sqsubseteq l\} \qquad \nu f = \bigsqcup \{l \mid l \sqsubseteq f(l)\}$$

The least and greatest fixpoint can also be obtained by iterating the function on the bottom and top elements of the lattice. This is often referred to as Kleene's theorem (at least for continuous functions) and it is one of the pillars of abstract interpretation [CC79]. Consider the (transfinite) ascending chain $(f^\beta(\bot))_\beta$ where β ranges over the ordinals, defined by $f^0(\bot) = \bot$, $f^{\alpha+1}(\bot) = f(f^\alpha(\bot))$ for any ordinal α and $f^\alpha(\bot) = \bigsqcup_{\beta<\alpha} f^\beta(\bot)$ for any limit ordinal α. Then $\mu f = f^\gamma(\bot)$ for some ordinal γ. The greatest fixpoint νf can be characterized dually, via the (transfinite) descending chain $(f^\alpha(\top))_\alpha$. Note also that $f^\alpha(\bot)$ is always a post-fixpoint and $f^\alpha(\top)$ is always a pre-fixpoint [BK+19a].

Given a lattice L, we define its height λ_L as the supremum of the length of any strictly ascending, possibly transfinite, chain. Then we have the following well-known result:

Theorem 2.1.3: Kleene's Iteration [CC79]

Let L be a lattice and let $f \colon L \to L$ be a monotone function. Consider the (transfinite) ascending chain $(f^\beta(\bot))_\beta$ where β ranges over the ordinals, defined by $f^0(\bot) = \bot$, $f^{\alpha+1}(\bot) = f(f^\alpha(\bot))$ for any ordinal α and $f^\alpha(\bot) = \bigsqcup_{\beta<\alpha} f^\beta(\bot)$ for any limit ordinal α. Then $\mu f = f^\gamma(\bot)$ for some ordinal $\gamma \leq \lambda_L$. The greatest fixpoint νf can be characterized dually, via the (transfinite) descending chain $(f^\alpha(\top))_\alpha$.

Note, that lattices can also be interpreted as algebraic structures [DP02]. We list some of the algebraic structures used within this thesis.

Given a set M equipped with a binary operation $\circ : M \times M \to M$ and some element $e \in M$, we call $(M, \circ, e)$ a *monoid* if it satisfies the following two axioms:

1. *Associativity*: For all $x, y, z \in M$ we have $(x \circ y) \circ z = x \circ (y \circ z)$.
2. *Identity element*: There exists one element $e \in M$ such that for all $x \in M$ the equations $e \circ x = x \circ e = x$ hold.

A *Boolean algebra* is given by a distributive and complemented lattice equipped with two binary operations $\bigsqcap$, $\bigsqcup$, and the unary operation of the complement.

Definition 2.1.4: Boolean Algebra

A Boolean algebra contains a set B, two elements 0 and 1, two binary operations $\sqcup, \sqcap : B \times B \to B$, and a unary operation $\neg : B \to B$ such that the *boolean laws* given by the following axioms hold:

1. *Distributivity*: $a \sqcap (b \sqcup c) = (a \sqcap b) \sqcup (a \sqcap c)$ and $a \sqcup (b \sqcap c) = (a \sqcup b) \sqcap (a \sqcup c)$ i.e. $(B, \sqcup, \sqcap)$ is a distributive lattice.
2. *Identity*: $x \sqcup 0 = x$ and $x \sqcap 1 = x$ for all $x \in B$.
3. *Double negation*: $\neg(\neg x) = x$ for all $x \in B$.
4. *Complementation*: $x \sqcup \neg x = 1$ and $x \sqcap \neg x = 0$ for all $x \in B$.

Therefore, a complete boolean algebra can be seen as a complete lattice.

On some occasions we consider concepts of *measure theory* and therefore we introduce *σ-algebras*.

Definition 2.1.5: σ-Algebra

Let X be a non-empty set and $\mathcal{P}X$ the corresponding powerset. A subset $\mathcal{A} \subseteq \mathcal{P}X$ is called a σ- algebra if the following three properties are satisfied by $\mathcal{A}$:

$$X \in \mathcal{A}$$
$$\text{if } A \in \mathcal{A} \text{ then } X \setminus A \in \mathcal{A}$$
$$\text{if } A_1, A_2, A_3, \ldots \in \mathcal{A} \text{ then } \bigcup_{n \in \mathbb{N}} A_n \in \mathcal{A}$$

2.2 State-Based Systems

Computer science is all about software systems. And such systems usually interact in some way with their environment, which can have different forms. The interaction

can take place between several kinds of actors: humans, other systems, or literally in a natural environment, as it is often the case with sensor-based monitoring systems.

Already two simple fragments of code [San11] make it clear that system behaviour depends on interaction, and therefore can be modelled nicely via state-based systems. Sangiorgi uses just two lines of source code, including addition, to demonstrate that a system adopts different memory states depending on concurrency (see Figure 2.1).

$$x := 3 \quad | \quad x := 1; x := x + 2$$

As a consequence, different behaviour can be observed from the outside, since the output of a simple addition is different depending on the moment of access to the memory state.

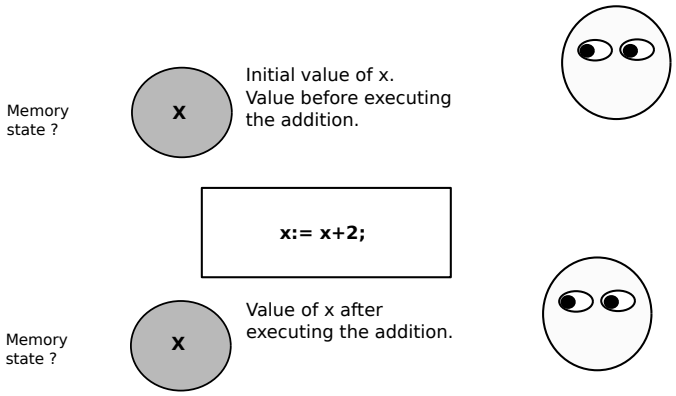


Figure 2.1: The value observable from the outside depends on the memory state. Therefore, concurrent accesses yield different observations [San11].

There are a variety of scenarios in which complex software systems with thousands lines of code play a role. Therefore, the next two subsections serve as an introduction to the topic of state-based systems, which are also known as transition systems. This is followed by an introduction to the term behavioural equivalence.

2.2.1 Three Different State-Based Systems

As already mentioned, state-based systems can be represented via transition systems, which mainly consist of the states and the transitions between these states. To facilitate the introduction, we start with the so-called labelled transition systems.

Definition 2.2.1: Labelled Transition System (LTS) [San11]

A labelled transition system is a triple $(S, \Sigma, \rightarrow)$ where S is a non-empty set called the domain of the LTS, Σ is the set of actions (or labels), and $\rightarrow \subseteq S \times \Sigma \times S$ is the transition relation.

LTS can be used to design systems to enable a clear presentation of requirements or implementations. Therefore, transition systems offer a powerful tool to check whether an implementation complies with the system specifications. The following example is intended to give an idea of how the comparison of system specification and implementation can work. In the real world, such designs encompass hundreds of states and numerous transitions. For the sake of simplicity, we limit ourselves to a rather abstract Example 2.2.2.

Example 2.2.2

A conference registration website is to be designed and implemented. There are two requirements that the system must meet: Firstly, the user should be able to pay after registration; secondly, the option to book a hotel should be available.

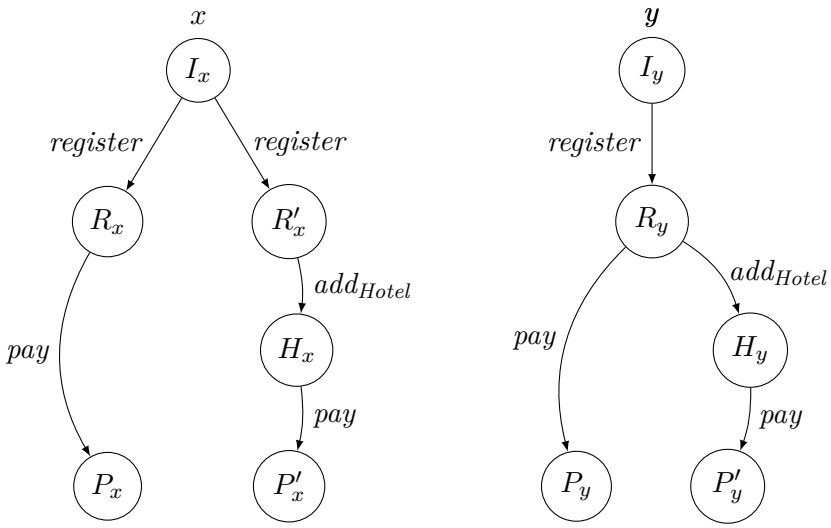


Figure 2.2: Two different concepts modelling a conference registration interface.

Two different designs are illustrated in Figure 2.2. A concrete specification is given by model y, and x represents a possible implementation. Both enable the sequences of actions: $register - pay$ and $register - add_{hotel} - pay$.
However model x forces the user to decide whether he/she wants to book a hotel right at the beginning of the registration process. But concerning specification y,

the user is supposed to make this decision right after the registration.

Example 2.2.2 makes it fairly clear that although both models support the same sequences of actions, they cannot be classified as behaviourally equivalent. However, this changes, when we think of deterministic finite automata (DFA), where the behaviour of two states is identified as equivalent if the same set of words is accepted from both states. In addition, non-deterministic (finite) automata (NFA) or in short finite automata can be extended in adding weights to the transitions. Expressing the behaviour in terms of (weighted) words is known as (weighted) language equivalence [BB+12a; Moh09].

An example application for weighted automata is digital text and language processing. The composition of various transducers, finite automata that can produce an output, is used for the representation of phonetic, acoustic and linguistic information. At the end of that composition, a weighted automaton is inserted that checks the outputs. In order to save computing time and storage space, minimization algorithms are used for the weighted automata [MPR96].

State-based systems are in general optimized by minimizing the states. The number of states can be reduced if states are behaviourally equivalent. Algorithms for minimizing a wide variety of transition systems can be found in numerous publications (see [Hop71; PT87; KS90]). The probably best-known algorithm is the minimization method for deterministic finite automata by Hopcraft [Hop71], which, in contrast to weighted automata, have a unique minimal representation [Moh09]. Generic approaches for minimizing transition systems are studied in more detail in Section 3.4.1.

To remain in the subject of weighted systems, we now consider discrete probabilistic systems. In general, there is a difference between discrete and continuous systems. In a discrete system, changes happen only at discrete time snippets and continuous systems update continuously over time. This results in state spaces that can no longer be countable. Systems dealing with continuous spaces are Markov processes [Sok11] or hybrid systems [KM15]. Hybrid or cyber-physical systems are used to enable the modelling of application scenarios, where a continuous system (physical environment) is connected to a control system (e.g. a system measuring coal combustion [MR+17]).

For the unfamiliar reader, we limit ourselves to a rather simple setting and concentrate on probabilistic systems over a countable state space. Probabilistic weights provide the right tools to model transition systems where we can not assume that the occurrence of a transition is always guaranteed. This approach is much more

realistic instead of simply assuming that a transition is available or not, since there are several application scenarios, where e.g. random events play a major role [Sok11].

Larsen and Skou [LS89] motivated their work about probabilistic transition systems by the fact that systems are build on other systems, from which it is assumed that they satisfy several properties. Testing whether such estimations hold leads to probabilistic modelling, where probability distributions are created based on the observations during several executions of a test.

Example 2.2.3

Given a system, consider the following assumption: there is a b-transition after executing an a-transition. But testing this system could yield the observation that this b-transition occurs only with a probability of 90%.

Definition 2.2.4: Discrete Probabilistic System [LS89]

A discrete probabilistic system is a quadruple (S, Σ, C, p) where S is a countable set of states, Σ is a set of labels, C is a Σ-indexed family of sets, with $C_a \subseteq S$ for $a \in \Sigma$ being the set of states which can perform an a-transition. Furthermore, p is a family of probability distributions $\mu_{s,a} : S \to [0, 1]$ for any $a \in \Sigma$ and $s \in C_a$.

Note, that a probability distribution $\mu_{s,a} : S \to [0, 1]$ satisfies the normalization property:

$$\sum_{s' \in S} \mu_{s,a}(s') = 1$$

Such probabilistic systems are also known as discrete reactive systems. If a state $s \notin C_a$ for $a \in \Sigma$, the a-behaviour of the state can be interpreted with termination. Otherwise, if a state $s \in C_a$, $\mu_{s,a}(s') = p$ means that s can perform the action a to the successor state s' with a probability of p [Sok11; LS89].

Example 2.2.5

The example in Figure 2.3 illustrates the weighted branching type working with an abstract set of actions $\Sigma = \{a, b\}$, where a possible action $l \in \Sigma$ from a state s to t is equipped with a probability. For example, the probability for making a b-transition from state 2 to state 5 is 0.8.

Furthermore, the states 3 and 5 do not admit the same weighted behaviour, since 3 does not enable a b-transition (3 is terminating for b), while 5 can even do a b-transition with probability 1. Therefore, one can also distinguish 1 and 2, since the probability reaching a state like 5 from 1 is 0 for all labels and we already

have mentioned, that the probability of 2 to reach 5 is clearly greater.

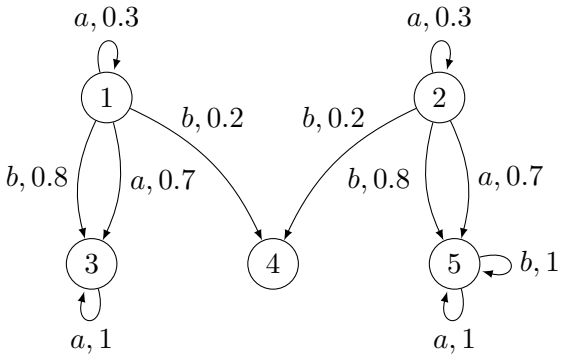


Figure 2.3: A probabilistic system where states can terminate for a label or have weighted *l-transitions* which sum up to 1 for each label $l \in \{a, b\}$ [KMS20b].

Another interesting branching type is captured by the so-called *Mealy machines*, where a transition from one state to another does not only model a possible action itself, as with LTS, but an input also determines a so-called output. Therefore, *Mealy machines* are eminently suited for the design of logical circuits [Mea55] or other application scenarios [Yan96; ARP13; MK16].

Definition 2.2.6: Mealy Machine

A Mealy machine is given by a 5-tuple $(S, s_0, \Sigma, \Gamma, \delta)$ where S is a non-empty set representing the state space, s_0 is the initial state, Σ is the set of input labels, Γ is the set of output labels and the transitions are given by a function $\delta : S \times \Sigma \to S \times \Gamma$.

For the modelling of logical circuits, we restrict to special Mealy machines and set both the input Σ and the output Γ alphabet to $\{0, 1\}$. There are two common ways to represent Mealy machines. First of all, we deal with the so-called state diagram, which is very close to the representation of LTS apart from the fact that the transitions are labelled with I/O where I represents the input separated by a "/" from the output O. The second concept is given by state transition tables. To get a little more familiar with the first concept, let us consider some standard example:

Example 2.2.7: A simple Mealy Machine

A simple Mealy machine represented as a state diagram is given in Figure 2.4, where we leave it unspecified which state is the initial one (while this plays a

significant role when looking at the question of what output the machine generates for a given input sequence, it plays a less important role if we are interested in the behaviour of individual states). To investigate this matter further, we take a closer look at the states $0, 1$ and 2.
For a given input sequence 000, the state 1 produces the output sequence 010. Furthermore, for the same input the states 0 and 2 generate the output 001. So you can tell from a short input that it makes a difference whether we choose state 0 or 1 as the initial state.
Although state 0 and 2 behave the same for the input 000, it is not clear from this one example whether 0 and 2 differ or not. In fact, a closer look at $0, 2$ reveals that these two states will also produce the same output for each other input sequence and therefore, it makes no difference which of the two states is taken as the initial state.

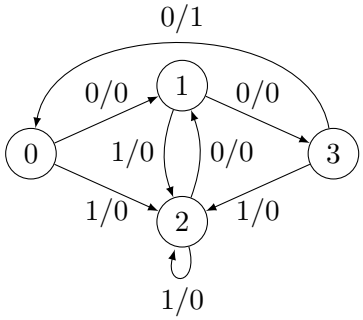


Figure 2.4: A simple Mealy machine represented as a state diagram based on [Mea55].

Similar to the case of LTS, one gets an idea of when two states in a given Mealy machine behave equivalently. Therefore, we will take a more formal look into this in the next Section 2.2.2.

First of all, the next section considers the meaning of behavioural equivalence in the individual scenarios. Secondly, an intensive introduction into a general definition of the term behavioural equivalence is presented in Section 2.3, since it plays a central role in the verification and optimization of systems.

2.2.2 Behavioural Equivalence

From the various types of branches discussed in the previous section (non-deterministic, probabilistic, and deterministic I/O branching), it is already clear that the sequences of transitions possible from a state are not sufficient to characterize the behaviour of

a state (see Example 2.2.2). The situation, therefore, differs from the case of finite automata belonging to the class of acceptance models (or string recognizers [San11]), where two states are called equivalent if they accept exactly the same set of words. This is also referred to as language-equivalence or trace-equivalence (see [Nic11; HU79; Gla01]). The question that arises here is whether it is even possible to define a general term for behavioural equivalence that covers different concepts.

Therefore, we first consider a notion of behavioural equivalence for each of the three branching types from the previous section. We start again with LTS and introduce bisimulation that covers the following intuition: two states (in the same system or different systems) are behaviour equivalent if the states can mutually simulate each other. In addition, after each successful simulation the side that determines the action – the *spoiler* side – can be swapped.

Definition 2.2.8: Bisimulation [San11]

Given a labelled transition system $(X, \Sigma, \to)$, a relation $\mathcal{R} \subseteq X \times X$ is called a bisimulation if, whenever $(x, y) \in \mathcal{R}$, for all $a \in \Sigma$ we have:

1. for all x' with $x \xrightarrow{a} x'$, there is a y' such that $y \xrightarrow{a} y'$ and $(x', y') \in \mathcal{R}$; and
2. for all y' with $y \xrightarrow{a} y'$, there is a x' such that $x \xrightarrow{a} x'$ and $(x', y') \in \mathcal{R}$.

Since bisimulations are closed under union, we can also speak about the greatest bisimulation given as the union of all bisimulations, called bisimilarity [San11]. Finally we conclude, that two states x, y are bisimilar (i.e. $x \sim y$) if there exists a bisimulation $\mathcal{R}$ containing the state pair (x, y).

Bisimilarity can be characterized alternatively by the greatest fixpoint of a monotone function over a powerset lattice.

Given an LTS $(X, \Sigma, \to)$. Let $R \subseteq X \times X$ we define $\mathcal{F}(R) : \mathcal{P}(X \times X) \to \mathcal{P}(X \times X)$ as follows:

$$\begin{aligned} \mathcal{F}(R) = \quad & \{(x, y) \in X \times X \mid \forall\, a \in \Sigma : \\ & \quad \forall\, x' \text{ with } x \xrightarrow{a} x', \exists\, y' \text{ s.t. } y \xrightarrow{a} y' \text{ and } (x', y') \in R; \qquad \text{and} \\ & \quad \forall\, y' \text{ with } y \xrightarrow{a} y', \exists\, x' \text{ s.t. } x \xrightarrow{a} x' \text{ and } (x', y') \in R \\ & \} \end{aligned} \tag{2.1}$$

Since we are working with monotone functions over complete lattices [San11], the iterative computation starting form the $\top$-element yields the greatest fixpoint.

Here, $\top$ is $X \times X$ with X being the state space of the underlying system. Therefore, bisimilarity is the largest post-fixpoint of $\mathcal{F}$ and can be computed if X is finite (see Knaster-Tarski [Tar55] and Theorem 2.1.3).

Proposition 2.2.9: Bisimilarity ($\sim$) [San11]

We call $\mathcal{F}$ the functional associated to bisimulation and thus we get:

1. $\sim$ is the greatest fixpoint of $\mathcal{F}$ (i.e. $\sim = \nu\mathcal{F}$).
2. $\sim$ is the largest relation $\mathcal{R}$ such that $\mathcal{R} \subseteq \mathcal{F}(\mathcal{R})$; thus $\mathcal{R}' \subseteq \sim$ for all $\mathcal{R}'$ with $\mathcal{R}' \subseteq \mathcal{F}(\mathcal{R}')$.

Proving Proposition 2.2.9, Sangiorgi also demonstrates that bisimilarity can be defined via coinduction [San11].

For probabilistic systems, we have already discussed in Example 2.2.5 the intuition, why the two states $(1, 2)$ do not admit the same weighted behaviour. To describe this more formally, two states are probabilistically bisimilar and therefore related by some relation R, if their probability to reach any equivalence class of R is the same for all labels.

Definition 2.2.10: Probabilistic Bisimulation [LS89]

Let $M = (S, \Sigma, C, p)$ be a discrete probabilistic system. Then a probabilistic bisimulation $\mathcal{R} \subseteq S \times S$ is an equivalence relation on S such that, whenever $(x, y) \in \mathcal{R}$, the following holds:

$$\forall\, a \in \Sigma. \forall\, Cl \in S/\mathcal{R}, \sum_{s' \in Cl} \mu_{x,a}(s') = \sum_{s' \in Cl} \mu_{y,a}(s').$$

Returning to Mealy machines, the first part defines when two states are equivalent according to Mealy and Moore (cf. [Moo56]), which also covers the intuition one might get from Example 2.2.7.

Definition 2.2.11: Equivalent States in a Circuit [Mea55]

Two states, s_0 in circuit S and t_0 in circuit T, are called equivalent if, given S initially in state s_0 and T initially in state t_0, there is no sequence of input combinations which, when presented to both S and T, will cause S and T to produce different sequences of output combinations.

Intending to build a formal bridge to the approach used for LTS, an equivalent definition given by [BRS08] with the restriction to the output $\Gamma = \{0, 1\}$ is given

below.

Definition 2.2.12: Bisimulation for Mealy Machines

Let two Mealy machines $S_M = (S, t_0, \Sigma, \Gamma, \delta)$ and $T_M = (T, s_0, \Sigma, \Gamma, \delta)$ be given. We call a relation $\mathcal{R} \subseteq S \times T$ a bisimulation if for all $(x, y) \in S \times T$ and $a \in \Sigma$ the following two conditions hold:

1. $(x, y) \in \mathcal{R} : \delta(x, a) = (x', o_x)$ and $\delta(y, a) = (y', o_y) \Rightarrow$ $o_x = o_y$ and $(x', y') \in \mathcal{R}$.
2. $(y, x) \in \mathcal{R} : \delta(y, a) = (y', o_y)$ and $\delta(x, a) = (x', o_x) \Rightarrow$ $o_y = o_x$ and $(x', y') \in \mathcal{R}$.

Therefore, s_0 and t_0 are bisimilar if there exists no input sequence that leads to a pair of successors that produce different outputs.

It is obvious that in analogy to LTS, a monotone function over a powerset lattice can also be defined here and for the weighted setting, where the greatest fixpoint characterizes bisimilarity.

Example 2.2.13: Logical Circuits and Bisimulation

Let's go back again to our previous Example 2.2.7. Now, we want to construct a truth table based on this Mealy machine and therefore we need to recode the states into binary numbers. Truth tables are an alternative representation of Mealy machines and are used to develop logical circuits, where this example only focuses on the theoretical connection between transitions systems and logical circuits (cf. [Mea55]).

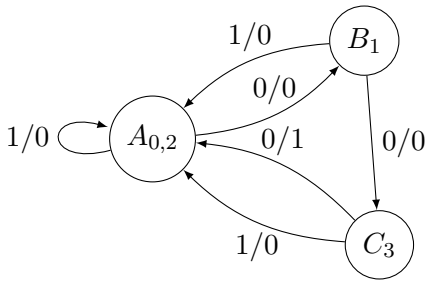


Figure 2.5: A minimized Mealy machine after merging the states $0, 2$ given in Figure 2.4.

First, we apply Definition 2.2.12 to recognize equivalent states. The states $0, 2$ produce for each binary input sequence the same output. More precisely, both

states induce the output 1 in case a single series of three zeros (i.e. 000) is detected. Merging these bisimilar states results in the state diagram in Figure 2.5.
Next, for a truth table we need to encode the three states $A_{0,2}, B_1, C_3$ where the indices indicate the states from the state diagram in Example 2.2.7. Thus two variables q_1, q_2 are needed to encode the three states $A_{0,2} \to 00$, $B_1 \to 01$, $C_3 \to 10$ as illustrated in Figure 2.1.

	q_1	q_2	X	$\bar{q}_1$	$\bar{q}_2$	Y
$A_{0,2}$	0	0	0	0	1	0
	0	0	1	0	0	0
B_1	0	1	0	1	0	0
	0	1	1	0	0	0
C_3	1	0	0	0	0	1
	1	0	1	0	0	0
	1	1	0			
	1	1	1			

Table 2.1: Minimized Mealy machine represented in a truth table.

Consider the row of $A_{0,2}$ then $q_1 = 0, q_2 = 0$ denote this state and X is the input variable, where the input either can be 0 or 1. Depending on X, the successor state is given by $\bar{q}_1, \bar{q}_2$ and the output Y is given in the last column on the right. For input 0 the state $A_{0,2}$ has a transition to B_1 and so $\bar{q}_1 = 0$, $\bar{q}_2 = 1$ producing the output $Y = 0$. The last row is not needed to model the Mealy machine and such a state is treated as *Don't care*.
That way we obtain Table 2.1, which is called a truth table and is another representation of the behaviour modelled by a Mealy machine. Working with such tables, boolean expressions are derived (e.g. via Karnaugh maps), which serve as a base in the design process of logical circuits satisfying the modelled behaviour (cf. *synthesizing sequential circuits* [Mea55]).
Therefore, our Mealy machine can be converted into a sequential circuit, which is discussed in [Mea55]. The size of such a circuit depends on the number of variables necessary to represent the states of the model and a minimization based on bisimulation *may* reduce the effort of the synthesis procedure (cf. [BRS08; Mea55]).

Moreover, the three introductory case studies demonstrate how important it is to

study behavioural equivalence in the development and implementation of (software) systems.

Furthermore, despite the differences from branching type to branching type, there are analogies in the way how bisimulation is defined. It is precisely these relationships that one would like to generally grasp in order to be able to develop generic techniques for the verification of state-based systems.

The foundations of such a generic framework are presented in detail in the next section.

2.3 Category Theory – Joy with Cats

In this section, we mainly work towards a generic definition of behavioural equivalence. But before we get to that, we need to introduce the basics of *category theory*.

In 1942 Eilenberg and McLane introduced the foundations of category theory [EM42; ME45]. To put it concisely: their concept brings common properties of different mathematical structures under one roof, and therefore category theory can be seen as a kind of (meta) language about mathematics [AHS09]. We will focus here on the basic definitions and background of these ideas.

But before we get lost in the details, let us first turn to the almost philosophical meaning, which helps to understand the difference between category and set theory. More precisely, we aim at the following paradoxical situation: Can there be a set containing all sets? This problematic question is better known under Russell's paradox and leads to the fact that such a set can not exist. However, if we want to study mathematical structures from a higher perspective, we have to circumvent this limitation, because to keep things general, statements have to be made that apply to all sets or all groups, etc. [AHS09].

Therefore, in the next Section 2.3.1 we introduce some formal definitions and start with some intuition how the limitation described above can be circumvented.

2.3.1 Categories and Morphisms

Instead of working with sets, in category theory, one considers collections of *objects* called *classes*, where an object can be a set. Therefore, one can define a class that contains all sets without getting into contradictions.

Most of the definitions and examples presented on the following pages are derived from [AHS09], but there also exist some variants in the literature.

Definition 2.3.1: Category

A **category** is a quadruple $\mathbf{C} = (\mathcal{O}, hom, id, \circ)$ consisting of

1. a class $\mathcal{O}$, whose members are called **C-objects** denoted with $Ob(\mathbf{C})$,
2. the class $Mor(\mathbf{C})$ of all **C-morphisms** which are given by arrows mapping from C to D and (C, D) is a pair of **C-objects**. The expression $hom(C, D)$ is an alternative notation for the **C-morphisms** mapping from C to D. The statement "f is a $hom(C, D)$" is expressed more graphically via arrows (i.e. $C \xrightarrow{f} D$). In addition, the domain C is denoted with $dom(f)$ and we write $cod(f)$ for the codomain of f.
3. a morphism $C \xrightarrow{id_C} C$, called the **C-identity** on C, for each **C-object** C.
4. a (partial) composition operation $\circ$ associating with each pair (f, g) of **C-morphisms** $A \xrightarrow{f} B$, $B \xrightarrow{g} C$ a **C-morphism** $A \xrightarrow{g \circ f} C$.

such that the following three axioms hold:

- Associativity: For **C-morphisms** f, g, h with $A \xrightarrow{f} B$, $B \xrightarrow{g} C$ and $C \xrightarrow{h} D$, the equation $h \circ (g \circ f) = (h \circ g) \circ f$ holds.
- Identity: For all **C-objects** A, B and all morphisms $A \xrightarrow{f} B$ it holds that $f \circ id_A = f = id_B \circ f$.
- Disjointness: The classes $hom(C, D)$ are pairwise disjoint.

In order to simplify the notations in concrete instantiations of the abstract definition above, we will mainly use $C, f \in \mathbf{C}$ for $C \in Ob(\mathbf{C})$ and $f \in Mor(\mathbf{C})$.

A set is a *small class* while classes such as the class of all sets are called *proper classes*. In case, the $hom(C, D)$ classes are restricted to sets, we have a so-called *locally small* category. Before we introduce some illustrative examples, we would like to conclude that if more than one category is considered simultaneously, subscripts are used to distinguish the objects, morphisms, compositions and so on.

Example 2.3.2: Three Categories

1. **Set** is the category, which specifies the framework for Chapter 3. The object class is the class of all sets and the morphisms are all functions. Therefore the identities are just the identity functions $id_S : S \to S$ on some set S and $\circ$ is the usual composition of functions.

2. **Rel** is a category, which serves as an example in Chapter 6. The object class is the class of all sets and the morphisms are all binary relations. Therefore, the identity for some set S is given by the identity relation $id_S = \{(x, x) \mid x \in S\}$ and the composition of two binary relations $A \xrightarrow{f} B$ and $B \xrightarrow{g} C$ is relational composition $A \xrightarrow{g \circ f} C$ with $g \circ f = \{(a, c) \in A \times C \mid \exists\, b \in B \text{ s.t. } (a, b) \in f, (b, c) \in g\}$.

3. **Graph** is a category, where the object class is the class of all finite graphs and the morphisms are all graph morphisms.

   Given two graphs $G_A = (V_A, E_A)$ and $G_B = (V_B, E_B)$, a graph morphism $f : (V_A, E_A) \xrightarrow{f} (V_B, E_B)$ is a structure preserving map defined by a morphism $\varphi_V : V_A \to V_B$ and a morphism $\varphi_E : E_A \to E_B$, where for each edge e mapping the vertices of e by φ_V is equal to the vertices of $\varphi_E(e)$ respecting the structure. In case of labelled graphs an edge in E_A is mapped by φ_E to an edge with the same label.

   Therefore, the identity for some graph $G = (V, E)$ is given by the identity morphisms $\varphi_V : V \to V$ and $\varphi_E : E \to E$.

   Furthermore, let two graph morphisms $f = (\varphi_{V_1}, \varphi_{E_1})$, $g = (\varphi_{V_2}, \varphi_{E_2})$ with $(V_A, E_A) \xrightarrow{f} (V_B, E_B)$, $(V_B, E_B) \xrightarrow{g} (V_C, E_C)$ where $\varphi_{V_1} : V_A \to V_B$, $\varphi_{V_2} : V_B \to V_C$ be given (analogous for the edge-morphisms).

   The composition $g \circ f$ is given by the composition of the vertex-morphisms $\varphi_{V_2} \circ \varphi_{V_1}$ and the composition of the edge-morphisms $\varphi_{E_2} \circ \varphi_{E_1}$ [BB+13]

The first category refers to the class of all sets, the second is intended to associate with morphisms something else than functions, and the third category serves as an example that objects can be different from sets.

These three examples already show how three completely different mathematical structures and their properties can be elegantly reconciled as categories.

Let us briefly come back to set theory and consider one property of functions. With a function $A \xrightarrow{f} B$ each element in the domain is mapped to an element in the codomain. One wonders whether this concept can be extended to categories. The next section looks at such an approach, which also brings the features that are particularly helpful in terms of our original motivation to design generic and suitable algorithms for system verification.

2.3.2 Functors, Natural Transformations and Limits

A category consists of a class of objects and a class of morphisms. If you think about a mapping between categories, it obviously makes sense to preserve the underlying structures (as composition of morphisms).

Definition 2.3.3: Functor

If $\mathbf{C}$ and $\mathbf{D}$ are categories, then a functor F from $\mathbf{C}$ to $\mathbf{D}$ denoted with $F : \mathbf{C} \to \mathbf{D}$ assigns to each **C-object** C a **D-object** D, and to each **C-morphism** $C \xrightarrow{f} C'$ a **D-morphism** $FC \xrightarrow{Ff} FC'$, in such a way that

1. F preserves composition; i.e., $F(f \circ g) = Ff \circ Fg$ whenever $f \circ g$ is defined, and
2. F preserves identity morphisms; i.e., $Fid_C = id_{FC}$ for each **C-object** C.

We have for every functor $F : \mathbf{C} \to \mathbf{D}$ the *opposite functor* $F^{\mathrm{op}} : \mathbf{C}^{\mathrm{op}} \to \mathbf{D}^{\mathrm{op}}$ which works exactly like F. The *opposite* $\mathbf{C}^{\mathrm{op}}$ of a category $\mathbf{C}$ has the same objects as $\mathbf{C}$ but includes the reversed morphisms (i.e. $(\mathbf{C}^{\mathrm{op}})^{\mathrm{op}} = \mathbf{C}$).

Before we also extend the functional concept to mappings between functors, let us first discuss a few simple functor examples followed by some significant properties of these special arrows. Therefore, we start with a specific class of functors.

Definition 2.3.4: Endofunctor

A functor which maps a category to itself is called *endofunctor*.

Example 2.3.5: Three Functors

Here are three examples that will serve, among other things, to reconcile the different definitions of bisimulations mentioned in the previous Section 2.2.2. But first, let us take a look at the following *endofunctors* over **Set** to see how mappings between categories work:

1. **Linear functors**: Here, we consider a very basic class of functors and also choose this one subclass as a representative of the so-called class of polynomial functors (see [GK09] for more information).

   Given a (finite) set A, the functor $F = A \times \_$ maps an object $X \in \mathbf{Set}$ to $A \times X$ and each function $X \xrightarrow{f} Y$ to the function $A \times X \xrightarrow{Ff} A \times Y$ with $Ff(a, x) = (a, f(x))$.

2. **Covariant power set functor**: Via the power set functor $F = \mathcal{P}$, a set X is

assigned to its power set $\mathcal{P}X = \{X' \mid X' \subseteq X\}$. Furthermore, each function $X \xrightarrow{f} Y$ is mapped to the function $\mathcal{P}X \xrightarrow{\mathcal{P}f} \mathcal{P}Y$ with $\mathcal{P}f(X') = f[X']$; i.e. the image of the function.

There are some necessary variants of this functor: via the power set functor $F = \mathcal{P}_\kappa$ a set X is assigned to the power set

$$\mathcal{P}_\kappa X = \{X' \subseteq X \mid card(X') < \kappa\}$$

where κ is a cardinal number. We denote the finite powerset functor with $\mathcal{P}_f$ where $\kappa = \omega$.

3. **Distribution functor**: The last functor we consider is the (finitely or countably supported) probability distribution functor $\mathcal{D}$ with $\mathcal{D}X = \{p\colon X \to [0,1] \mid \sum_{x \in X} p(x) = 1\}$ where either finitely or countably many $x \in X$ with $p(x) > 0$ exist.

   For a function $X \xrightarrow{f} Y$ we get $\mathcal{D}X \xrightarrow{\mathcal{D}f} \mathcal{D}Y$ with $\mathcal{D}f(p) = q$ where $q : Y \to [0,1]$ and $q(y) = \sum_{x, f(x)=y} p(x)$.

Next, we consider the following useful closure property, which enables the composition of different functors.

Proposition 2.3.6: Composition of Functors

If $F : \mathbf{C} \to \mathbf{D}$ and $G : \mathbf{D} \to \mathbf{E}$ are functors, then the composite $G \circ F : \mathbf{C} \to \mathbf{E}$ defined by

$$(G \circ F)(A) = G(F(A))$$

and

$$(G \circ F)(A \xrightarrow{f} A') = G(FA) \xrightarrow{G(Ff)} G(FA')$$

is a functor.

The Proposition above allows us to construct functors such as $\mathcal{P}(A \times X)$ or $\mathcal{D}(A \times X)$ where X is a set. We will discuss examples of such composed functors in more detail in the next section, but let us first complete the basic knowledge about functors.

Definition 2.3.7

Let $F : \mathbf{C} \to \mathbf{D}$ be a functor

1. F is called *faithful* provided that all hom(-set) restrictions

$$F : hom_{\mathbf{C}}(C, C') \to hom_{\mathbf{D}}(FC, FC')$$

   are injective.

2. F is called *full* provided that all hom(-set) restrictions are surjective.

Faithful functors are used to define a special category. Such categories are summarized as *concrete categories* over **Set**, so categories whose objects can be represented by a set via a faithful functor $U : \mathbf{C} \to \mathbf{Set}$ where U also somehow *forgets* the structure of the object.

Definition 2.3.8: Concrete Category

Let $\mathbf{C}$ be a category. A concrete category over $\mathbf{C}$ is a pair $(\mathbf{A}, U)$ where $\mathbf{A}$ is a category and $U : \mathbf{A} \to \mathbf{C}$ is a faithful functor. Sometimes U is called the *forgetful* (or *underlying*) *functor* of the concrete category and $\mathbf{C}$ is called the *base category* for $(\mathbf{A}, U)$.

There are numerous of these concrete categories, such as **Poset**, whose objects consist of a set and an underlying structure.

Example 2.3.9: Poset

The category **Poset** is defined as follows:

Objects The objects are all pairs $(M, \leq_M)$ of sets M which are partially ordered by $\leq_M$.

Arrows The arrows are all order-preserving functions.

$\circ$ **operator** Concatenation is function composition: $f\colon (A, \leq_A) \to (B, \leq_B)$, $g\colon (B, \leq_B) \to (C, \leq_C)$ can be composed as $g \circ f\colon (A, \leq_A) \to (C, \leq_C)$ according to $g \circ f(x) = g(f(x))$ for all $x \in (A, \leq_A)$.

Identity The identity arrows $id_{(A,\leq_A)}\colon (A, \leq_A) \to (A, \leq_A)$ are just the identity functions where $id_{(A,\leq_A)}(x) = x$ for all $x \in A$.

A forgetful functor $U : \mathbf{Poset} \to \mathbf{Set}$ maps an object $(M, \leq_M)$ to M.

Before we turn to a generic concept of behavioural equivalence based on the knowledge presented so far, we would like to close this section by looking at mappings between functors. This concept is fundamental for most of the contents within this thesis, especially for Section 3.3.1, since it serves as the basis for the definition of generic modal logics.

Definition 2.3.10: Natural Transformation

Let $F, G : \mathbf{C} \to \mathbf{D}$ be functors. A *natural transformation* τ from F to G (denoted by $\tau : F \to G$ or $F \xrightarrow{\tau} G$) is an arrow that assigns to each $\mathbf{C}$-object C a $\mathbf{D}$-morphism $\tau_C : FC \to GC$ in such a way that the following *naturality condition* holds: for each $\mathbf{C}$-morphism $C \xrightarrow{f} C'$, the square

$$\begin{array}{ccc} FC & \xrightarrow{\tau_C} & GC \\ {\scriptstyle Ff}\downarrow & & \downarrow{\scriptstyle Gf} \\ FC' & \xrightarrow{\tau_{C'}} & GC' \end{array}$$

commutes. In the following we denote an identity natural transformation from F to F with Id_F.

Another basic categorical definition (*limit* or dually *colimit*) covers universal properties of several constructions. Probably the most well-known example of a *limit* is given by the *product*.

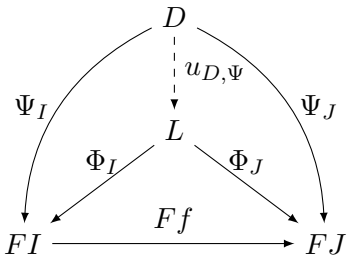
Let $\mathbf{C}$ and $\mathbf{D}$ be categories. A diagram of type $\mathbf{C}$ is a functor $F : \mathbf{C} \to \mathbf{D}$ where the category $\mathbf{C}$ is also called *index category*. A diagram from some object A to some other object B given by $A \leftarrow C \to B$ is called *span* with index category $(* \leftarrow \circ \to \star)$ and dually $A \to C \leftarrow B$ is called *cospan* with index category $(* \to \circ \leftarrow \star)$.

Definition 2.3.11: Cone, Limit

Let $F : \mathbf{C} \to \mathbf{D}$ be a diagram. A *cone* (to F) is a tuple (D, Ψ) with $D \in \mathbf{D}$ and a family of morphisms $\Psi_I : D \to FI$ for each object $I \in \mathbf{C}$ such that each triangle as follows commutes for all $\mathbf{C}$-objects I, J and every $f \in (I, J)$:

$$\begin{array}{ccccc} & & D & & \\ & {\scriptstyle \Psi_I}\swarrow & & \searrow{\scriptstyle \Psi_J} & \\ FI & & \xrightarrow{Ff} & & FJ \end{array}$$

We call a cone (L, Φ) a *limit* if it is *universal* in the sense that for any cone D, Ψ there is always a unique arrow $u_{D,\Psi} : D \to L$ such that the following diagram commutes for any $I, J \in C$ and for any $f \in (I, J)$:



In case $u_{D,\Psi}$ exists but it is not unique the limit is called a *weak* limit.

By the way, a very nice feature of category theory is that many properties of mathematical constructions can be represented via diagrams, which simplify the proof argumentation and improve the readability.

In the next section, we will discuss a generic way to model transition systems and study bisimulation from a categorical perspective. And finally, we will see how the notions of bisimulation we have introduced earlier (see Section 2.2.2) are related to a categorical notion of behavioural equivalence over state-based systems.

2.3.3 Behaviour Coalgebraically

In order to handle system verification in a generic way, we not only need a more general definition of behavioural equivalence. But above all, we require a formalism in order to generically model state-based systems for different branching types as listed in Section 2.2.2.

Here, we are studying a framework that gives us exactly the tools we need and all the content presented in the following mainly originates from Rutten's work known under the title *"universal coalgebra"* [Rut00]. The definition we want to start with makes use of functors to capture different branching types:

Definition 2.3.12: F-coalgebra

Let $F : \mathbf{Set} \to \mathbf{Set}$ be a functor. An F-coalgebra or F-system is a pair (X, α) consisting of a set X and a function $\alpha : X \to FX$. The set X is called the carrier of the system (i.e. the state space); the function α is called the F-transition structure of the system.

Note that F can also be defined over a category different from **Set**. However, to

avoid over-complicating, for now, let us stay with endofunctors over **Set**.

In order to bring the power of this definition and all the knowledge from the previous sections together in one place, let us take a closer look at α. We have a state space X and each $x \in X$ is assigned to an element $\alpha(x) \in FX$. Assume that X is finite and $F = \mathcal{P}$, then one state is related to a subset of X. Exactly this happens to a state in an unlabelled transition system (i.e. a labelled system with $|\Sigma| = 1$, so there is no need to use labels at all).

Example 2.3.13

As already indicated after Proposition 2.3.6 we come back to some examples of composed functors and link this to our introductory examples of state-based systems which are modelled via coalgebras $\alpha : X \to FX$ with F defined as follows:

1. **Labelled transition systems**: One can define an LTS via the composed functor $\mathcal{P}(A \times X)$, where X is the state space and A the set of actions. In the example of Figure 2.2 this means that the state I_x is assigned to $\alpha(I_x) = \{(\mathit{register}, R_x), (\mathit{register}, R'_x)\}$, since under the action *register*, transitions to the two states R_x, R'_x are possible and no further transitions exist.

2. **Probabilistic systems**: Here, we combine termination represented via $1 = \{\bullet\}$ with the space $\mathcal{D}X$ to $(\mathcal{D}X + 1)^A$. So for each action $a \in A$, x is either terminating or maps to a probability distribution $p \in \mathcal{D}X$, which coincides with the branching illustrated in Figure 2.3.

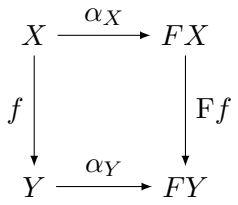
3. **Mealy machines**: To model this kind of branching we restrict again to $\Sigma = \Gamma = \{0, 1\}$ and define $FX = (\Gamma \times X)^\Sigma$. Therefore, for each input signal $i \in \Sigma$, x is associated with some successor state and the corresponding output.

Note that we use A instead of Σ in terms of coalgebras.

Finally, we move to a coalgebraic view on bisimulation to enable a generalization of system behaviour analysis. Therefore, we first introduce morphisms between *F-coalgebras* that preserve and reflect *F-transition* structures.

Definition 2.3.14: F-homomorphism

Let (X, α_X) and (Y, α_Y) be F-coalgebras, where F is again an arbitrary functor. A morphism $f : X \to Y$ is called an F- or **coalgebra homomorphism** if $Ff \circ \alpha_X = \alpha_Y \circ f$, i.e. the following diagram commutes:



On some occasions we will need to construct a coalgebra homomorphism $h : X \to Z$ in **Set** via factorizing through an equivalence relation R (i.e. quotient by an equivalence relation). The generalization of this construction is given by a coequalizer.

Definition 2.3.15: Coequalizer

Given a category **C** and two parallel morphisms $f, g : A \to B$ where A, B, $f, g \in \mathbf{C}$. An f, g-coequalizer is given by an object Q together with a morphism $q : B \to Q$ such that $q \circ f = q \circ g$. Moreover, for any other pair (Q', q') there must exist a unique morphism $u : Q \to Q'$ (i.e. $\exists!\ u$) such that $u \circ q = q'$ (illustrated in Figure 2.6).

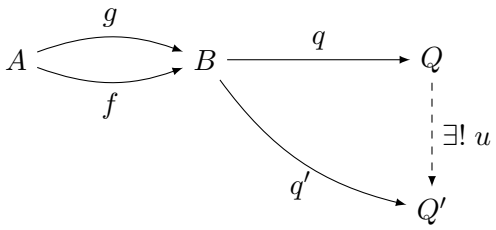
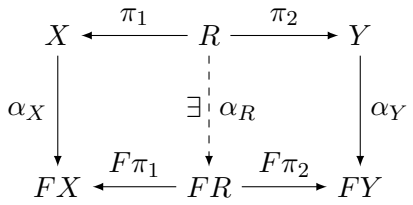


Figure 2.6: The triangle commutes and u is a unique morphism.

The most common coalgebraic bisimulation definition is taken from [Rut00] but originates from the work presented in *"A final coalgebra theorem"* by Aczel and Mendler [AM89]. Intuitively, given two coalgebras $(X, \alpha_X), (Y, \alpha_Y)$, this definition requires that for the relation R a coalgebra α_R exists, such that for each xRy the transition structures of $\alpha_X(x)$ and $\alpha_Y(y)$ are both matched by $\alpha_R(x, y)$.

Definition 2.3.16: F-bisimulation

Let (X, α_X) and (Y, α_Y) be F-coalgebras. A subset $R \subseteq X \times Y$ is called an F-bisimulation between X and Y if there exists an F-transition structure $\alpha_R : R \to FR$ such that the projections π_i for $i \in 1, 2$ from R are coalgebra homomorphisms $F\pi_1 \circ \alpha_R = \alpha_X \circ \pi_1$ and $F\pi_2 \circ \alpha_R = \alpha_Y \circ \pi_2$ i.e. the following two diagrams commute:



Having found a generic view of bisimulation, we could consider generic algorithms that compute F-bisimulations. One idea would be to construct a generic monotone function $\mathcal{F}_\alpha : \mathcal{P}(X \times X) \to \mathcal{P}(X \times X)$ analogous to the case of LTS and to determine the most efficient way to compute the greatest fixpoint. For now, we delay this discussion until the preliminaries of Chapter 3. First, we need to review whether F-bisimulation captures the three instances of bisimulations regarding our three introductory examples. Luckily, our expectations are confirmed by the work in [Rut00; dR99].

Therefore, one might think that we arrived at the end of our journey in the world of generic definitions, but there exists another coalgebraic definition for behavioural equivalence (cf. [Kur00]):

Definition 2.3.17: Coalgebraic Behavioural Equivalence

Let $\mathbf{C}$ be a concrete category and $(X, X \xrightarrow{\alpha} FX)$,$(Y, Y \xrightarrow{\beta} FY)$ two F-coalgebras. We call the elements $x \in UX$ the states of (X, α). Two states x, y with $x \in UX$ and $y \in UY$ are behaviourally equivalent if and only if there exists an F-coalgebra (Z, γ) and two coalgebra-homomorphism $f : (X, \alpha) \to (Z, \gamma)$, $g : (Y, \beta) \to (Z, \gamma)$ such that $Uf(x) = Ug(y)$.

In the next example [BB+12a] we will show that *coalgebraic behavioural equivalence* captures weighted bisimulation where F-bisimulation does not. Unfortunately as a consequence, one can conclude from this, that in general *coalgebraic behavioural equivalence* and F-bisimulation do not coincide.

Example 2.3.18: F-bisimulation vs. Weighted Bisimulation

The authors in [BB+12a] study weighted automata with weights from a field $\mathbb{K}$. First, consider the functor $\mathcal{W} : \mathbf{Set} \to \mathbf{Set}$ with $\mathcal{W}X = \mathbb{R}_0 \times (\mathbb{R}_0^X)_\omega^A$ (where $(\mathbb{R}_0^X)_\omega$ is the set of functions $X \to \mathbb{R}_0$ with finite support). Thus, weighted automata are modelled as coalgebras $\alpha : X \to \mathcal{W}X$ where each $x \in X$ is mapped to a weighted termination behaviour and a function $A \xrightarrow{t_x} (\mathbb{R}_0^X)_\omega$, which determines its weighted transition structure.
As can been seen in Figure 2.7, x_1 and y_1 are weighted bisimilar, witnessed by the following equivalence classes: $C_1 = \{x_1, y_1\}$, $C_2 = \{x_2, x_3\}$.

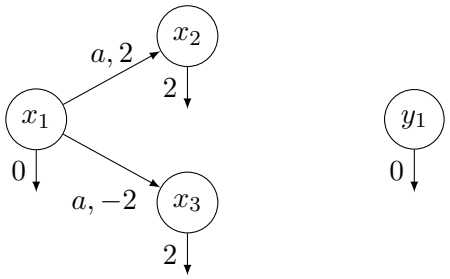


Figure 2.7: Two weighted systems for the functor $\mathcal{W}X = \mathbb{R}_0 \times (\mathbb{R}_0^X)_\omega^A$ with $X = \{x_1, x_2, x_3\}, Y = \{y_1\}$ and $A = \{a\}$, where x_1, y_1 capture the same weighted behaviour, but there exists no $\mathcal{W}$-bisimulation $\mathcal{R} \subseteq X \times Y$ including (x_1, y_1).

The states in class $C_2 = \{x_2, x_3\}$ admit the same behaviour given by their 2-*weighted* terminating transition. In addition, the termination weights of x_1 and y_1 are equal and for both classes C_i with $i \in \{1, 2\}$ we have that

$$\sum_{c \in C_i} \alpha_2(x_1)(c) = \sum_{c \in C_i} \alpha_2(y_1)(c)$$

holds, where $\alpha_2(x)$ denotes the second component of the functor $\mathcal{W}$. Therefore, we are able to define two coalgebra homomorphisms:
$g : (\{x_1, x_2, x_3\}, \alpha_X) \to (\{c_1, c_2\}, \alpha_C)$ and $h : (\{y_1\}, \alpha_Y) \to (\{c_1, c_2\}, \alpha_C)$ with $g(x_1) = c_1 = h(y_1)$, but we can not derive a $\mathcal{W}$-bisimulation from this. The only relation $R \subseteq X \times Y$, that can be defined including (x_1, y_1) is $\{(x_1, y_1)\}$, which obviously does not include pairs of non-equivalent terminating states like (y_1, x_2) or (y_1, x_3).
Therefore, no matter what we choose as mediating morphisms $\alpha_R : R \to \mathcal{W}R$ regarding the definition of F-bisimulation, the left square will not commute. For $\pi_1^{-1}(x_2) = \emptyset$ we get a function $\mathcal{W}(\pi_1) \circ \alpha_R$ mapping (x_1, y_1) into an element

$(t, f) \in \mathbb{R}_0 \times (\mathbb{R}_0^X)_\omega^A$ such that $f(a)(x_2) = f(a)(x_3) = 0$.
On the other hand $\alpha_X \circ \pi_1(x_1, y_1)$ maps to a different element (t', f') with $f'(a)(x_2) = 2$ defined by the structure given in Figure 2.7 (see [BB+12a]).

The example above illustrates that two states are weighted bisimilar, but fail to be *F-bisimilar* regarding $X \times Y$.

Remark 2.1. *Example 2.3.18 also serves as a witness, that in general F-bisimulation regarding $X \times Y$ behaves differently compared to $(X \cup Y) \times (X \cup Y)$ for two given coalgebras $\alpha : X \to FX$, $\beta : Y \to FY$. Indeed, there exists an F-bisimulation including (x_1, y_1) over the state space union, namely $\mathcal{R} = id_{X \cup Y} \cup \{(x_1, y_1), (x_2, x_3), (x_3, x_2)\}$.*

For other non-weak pullback preserving functors it is already known that they preserve kernel pairs [HKP07]. In case a functor preserves kernel pairs one can show that both notions coincide over single systems. In addition, it is well-known that a monoid functor with a refinable monoid *preserves kernel pairs (see [GS01]).*

Moreover, we observed that the weighted bisimilarity of these states can be captured by two coalgebra homomorphisms. In [BB+12a] the authors proved, that for weighted automata described via the functor $\mathcal{W} = \mathbb{K} \times (\mathbb{K}-)_\omega^A$ weighted bisimulation and coalgebraic behaviour equivalence are equivalent.

It is also well known that when a functor preserves *weak pullbacks*, coalgebraic behaviour equivalence and F-bisimulation coincide (cf. [Rut00; BSd04]).

Definition 2.3.19: (Weak) Pullback

Let two morphisms $g : X \to Z$, $h : Y \to Z$ be given. A pullback of h and g consist of an object P and two morphism $p_1 : P \to X$, $p_2 : P \to Y$ such that the diagram in Figure 2.8 commutes.

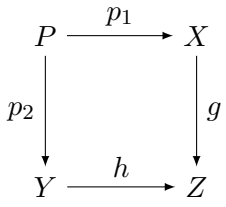


Figure 2.8: P is a limit of the cospan $\langle Z, g, h \rangle$.

Moreover, for any other triple (Q, q_1, q_2) for which the diagram in Figure 2.9 commutes, there must exist a unique morphism $u : Q \to P$ such that $q_i = p_i \circ u$ with $i \in \{1, 2\}$.

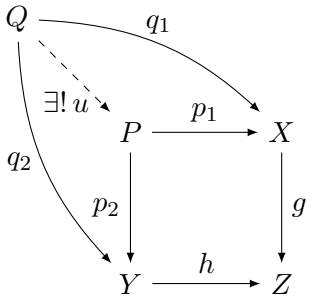


Figure 2.9: P is a pullback.

In case this morphism u exists, but is not unique P is a weak pullback.

A commonly known example for a pullback in **Set** is a kernel of a morphism.

Example 2.3.20

Given a function $f : X \to Y$ in **Set**. A kernel is defined as follows:

$$ker(f) = \{(x, x') \in X \times X \mid f(x) = f(x')\}$$

Alternatively one can define $ker(f)$ as a pullback P along f equipped with the usual projections (P, π_1, π_2) with $p_i : P \to X$ for $i \in \{1, 2\}$.

We proceed with a detailed discussion of the fact that in case a functor preserves weak pullbacks, this implies that our two notions of behaviour equivalence coincide. Although this result is very well known and mentioned in numerous publications (e.g. [Rut00; BSd04]), we explore it here again, since it is one of the limitations that we use within Chapter 3. Besides, it also demonstrates how categorical definitions help to prove interesting theorems since one can use properties in a more general way. For instance, one can assume that a certain morphism exists if one requires that the underlying functor preserves pullbacks.

Given two F-coalgebras $(X, \alpha_X), (Y, \alpha_Y)$ and two coalgebra homomorphisms $g : X \to Z, h : Y \to Z$ we now try to identify how these homomorphisms can be used to construct an F-bisimulation over $X \times Y$ in case F preserves weak pullbacks while the opposite direction is straightforward. The first idea is to construct a bisimulation via $R = \{(x, y) \mid g(x) = h(y)\}$ as depicted in the commuting left diagram of Figure 2.10. Note, that R is also a limit of the cospan $\langle Z, g, h \rangle$.
Let us turn our attention back to functors and assume that applying a functor to a pullback preserves this property in a weaker sense. This means, that FR becomes a

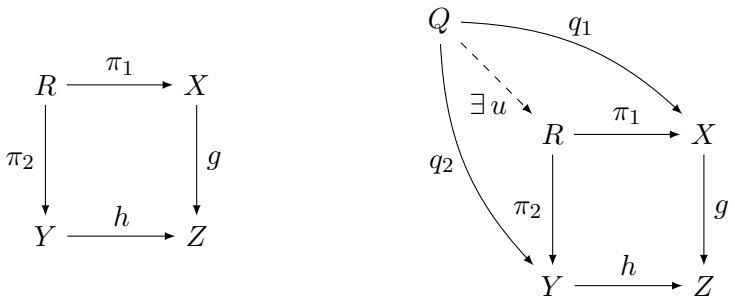


Figure 2.10: Diagram to the left: $R = \{(x, y) \mid g(x) = h(y)\}$ is a limit of the cospan $\langle Z, g, h\rangle$. Diagram to the right: In addition R is a pullback.

weak pullback and so for each triple Q, q_1, q_2 such that the diagram $\langle Q, FX, FY, FZ\rangle$ commutes, there exists a not necessarily unique morphism $Q \xrightarrow{m} FR$ such that $q_1 = F\pi_1 \circ m$ and $q_2 = F\pi_2 \circ m$ hold, as represented in Figure 2.11.

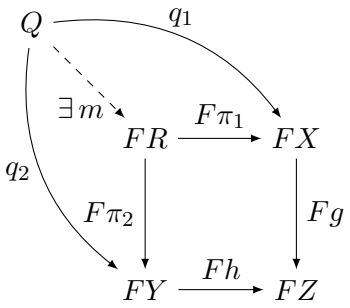


Figure 2.11: A functor F transforms a pullback R into a weak pullback FR.

Considering the definition of F-bisimulation, a mapping $m : R \to FR$ is required, such that the projections π_i with $i \in \{1, 2\}$ are coalgebra homomorphisms and therefore the left and right square in Definition 2.3.16 commute. For the triple (R, q_1, q_2) with $Fg \circ q_1 = Fh \circ q_2$ we construct the required morphism m under the assumption that $(FR, F\pi_1, F\pi_2)$ is a weak pullback as illustrated in Figure 2.11. Remember, that $Fg \circ \alpha_X \circ \pi_1 = Fh \circ \alpha_Y \circ \pi_2$ since g, h are coalgebra homomorphisms with $f(\pi_1(r)) = g(\pi_2(r))$ for all $r \in R$ and R is a pullback. In case F preserves weak pullbacks, R is an F-*bisimulation* shown in Figure 2.12.

In [BB+12a] it is mentioned that negative weights cause the problem described in Example 2.3.18 and that in general, there seems to be a connection between weak pullback preserving functors and zero-sum-free monoids, which is extensively discussed in [GS01]. At this point, we close this subject and summarize it in the following well-known Theorem 2.3.21.

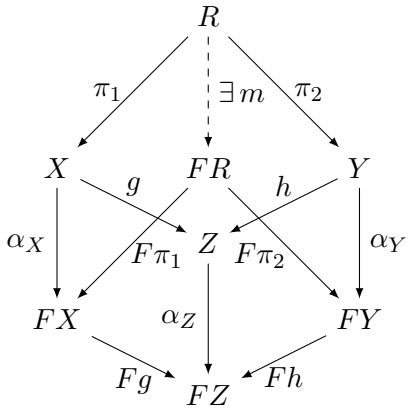


Figure 2.12: A functor F preserves weak pullbacks and therefore R is an F-bisimulation.

Theorem 2.3.21: (cf. [BSd04, Corollary 8])

When the functor F preserves weak pullbacks, the notion of F-bisimulation and behavioural equivalence for F-coalgebras coincide.

Finally, we want to emphasize that for our three functors introduced above (LTS, Mealy machines and discrete probabilistic systems) it is well-known, that they preserve weak pullbacks.

An iterative and categorical concept to define behavioural equivalence between states in a given system $\alpha : X \to FX$ is based on the *terminal object* and *final chain* with respect to the functor F [Wor05; AHS09].

Definition 2.3.22: Terminal Object and Final Chain

Let $\mathbf{C}$ be a category. An object $T \in \mathbf{C}$ is terminal, if there exists a unique morphism $t_X : X \to T$ for all objects $X \in \mathbf{C}$. (A terminal object is usually denoted with 1.)

Let $F : \mathbf{C} \to \mathbf{C}$ be given. A construction of a sequence (*called final chain*) based on the objects $F^i 1$ with $i \in \mathbb{N}_0$ is obtained by applying F iteratively starting with the unique morphism:

$$1 \overset{!}{\leftarrow} F^1 1 \overset{F!}{\leftarrow} \cdots F^i 1 \overset{F^i!}{\leftarrow} F^{i+1} 1 \cdots$$

Example 2.3.23

In **Set** any singleton $1 = \{s\}$ is a terminal object and $t_X : X \to 1$ is the corresponding unique morphism [AHS09].

To see how this is related to the behaviour of a given coalgebra $\alpha : X \to FX$, we have to link the coalgebra to the final chain starting with the unique morphism $\alpha^0 = X \to 1$:

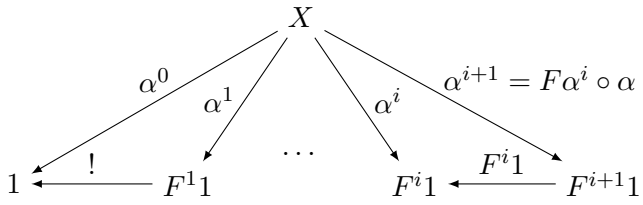


Figure 2.13: Final chain equipped with a sequence $\alpha^i : X \to F^i 1$ of morphisms [KK18].

The final chain formalizes the intuition that two states are equal if they can not be distinguished by their α-step behaviour. The codomains of each morphism $X \to F^i 1$ can be interpreted as the i-step behaviours for a given coalgebra α [Pat04; WD+20]. It is well-known that in case of **Set** and X is finite, two states $x, y \in X$ are behaviourally equivalent iff $\alpha^i(x) = \alpha^i(y)$ for all $i < \omega$ [Wor05]. We will see in Section 3.4.1 how one can derive *partition refinement* algorithms from the final chain to compute behavioural equivalence relations.

Regarding language equivalence mentioned in Section 2.2.2 the following question arises: *"Do we always want to consider F-bisimulation? Maybe sometimes language equivalence plays a more important role? And we already know that in general they do not coincide."*

Definitely, these are important questions and it still remains open how the definition of *coalgebraic behaviour equivalence* is related to *language equivalence*. To briefly touch some of the contents presented in the next section, there are different coalgebraic ways to model the same branching type. By changing the category one can achieve that *coalgebraic behaviour equivalence* is *language equivalence* [PT99; KK18].

2.3.4 Adjunctions and Monads

In this subsection, we study the categorical concepts *adjunction* and *monad*. Monads can be used to model side effects, for instance in connection with trace equivalence (see Section 2.2.2) and we will address such behavioural notions in Chapter 6.

The following definitions, examples, and explanations are taken from [Mac98; AHS09; Ker16] and from the work by Power and Turi, which presents a coalgebraic treatment of trace semantics for labelled transition systems [PT99].

We already introduced natural transformations in Definition 2.3.10 to obtain structure preserving maps between functors. As the name indicates, a functor is transformed into another functor in a natural way, i.e. the diagram of Definition 2.3.10 commutes. Next, we want to focus on the relation between two categories $\mathbf{C}, \mathbf{D}$ in case there exist two functors, where one maps from $\mathbf{C}$ to $\mathbf{D}$ and the other functor goes from $\mathbf{D}$ to $\mathbf{C}$. The next Definition 2.3.24 is taken from [Ker16] based on [Awo06].

Definition 2.3.24: Adjunction, Adjoint Functor

Let $\mathbf{C}$ and $\mathbf{D}$ be categories. An *adjunction* between $\mathbf{C}, \mathbf{D}$ consists of

1. a functor $L : \mathbf{C} \to \mathbf{D}$, called *left adjoint*,
2. a functor $R : \mathbf{D} \to \mathbf{C}$, called *right adjoint*,
3. a natural transformation $\eta : ID_{\mathbf{C}} \to RL$, called *unit*, and
4. a natural transformation $\varepsilon : LR \to ID_{\mathbf{D}}$, called *counit*,

such that the two diagrams in Figure 2.14 commute.

Figure 2.14: Two commuting diagrams.

Such an adjunction is denoted by $(L \dashv R, \eta, \varepsilon) : \mathbf{C} \to \mathbf{D}$. A functor F is a left or right adjoint if it is a left or right adjoint of some adjunction.

At a first reading of this definition, it may seem difficult to understand what an adjunction means intuitively. Since we will work with adjunctions in Chapter 6, we consider the following concrete adjunction in Example 2.3.25.

Example 2.3.25

This example of a well-known adjunction serves as an introduction. We consider the categories **Set** and **Rel** defined in Example 2.3.2. Let a functor $L : \mathbf{Set} \to \mathbf{Rel}$ map a set X to X and a function $f : X \to Y$ to the corresponding relation $R_f = \{(x, y) \mid f(x) = y\}$. Obviously, L is the inclusion functor $I : \mathbf{Set} \to \mathbf{Rel}$. Next, we consider the functor $R : \mathbf{Rel} \to \mathbf{Set}$ (i.e. $R = \mathcal{P}$) which maps each object X to $\mathcal{P}X$ and each relation $R \subseteq X \times Y$ to the function $f_R : \mathcal{P}X \to \mathcal{P}Y$ defined as follows:

$$f_R(X') = \{y \in Y \mid \exists x \in X' : xRy\}$$

Furthermore, we define the *unit* $\eta : ID_{\mathbf{Set}} \to \mathcal{P}$ by the functions

$$\eta_X : X \to \mathcal{P}X,\ \eta_X(x) = \{x\}$$

for every $X \in \mathbf{Set}$. In addition, we obtain the *counit* $\varepsilon : \mathcal{P} \to ID_{\mathbf{Rel}}$ from the relations

$$\varepsilon_X : \mathcal{P}X \to X,\ (X', x) \in \varepsilon_X \iff x \in X'$$

for each set X where $X' \subseteq X$. Finally, we summarize the four components of the adjunction $(I \dashv \mathcal{P}, \eta, \varepsilon)$.

Now we are ready to explain some useful properties of an adjunction, which we will later use within our proofs (see Chapter 6). The properties we are interested in are illustrated by Schema (2.2), which shows, that there is a one-to-one correspondence between specific morphisms of the underlying categories **C** and **D**.

$$\frac{h : LX \to A \in \mathbf{D}}{f : X \to RA \in \mathbf{C}} \quad (L \dashv R) \tag{2.2}$$

To explain how Schema (2.2) is guaranteed by an adjunction, we will first give a general proposition followed by a demonstration using the adjunction $(I \dashv \mathcal{P}, \eta, \varepsilon)$ of Example 2.3.25.

Proposition 2.3.26

Given an adjunction $(L \dashv R, \eta, \varepsilon)$ the natural transformation η is universal to R. This means, that for every $f : X \to RA$ there is a unique $h : LX \to A$ such that the diagram in Figure 2.15 commutes.

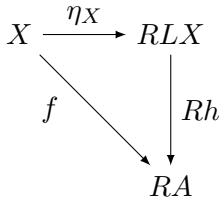


Figure 2.15: A unique transpose h of f wrt. $(L \dashv R, \eta, \varepsilon)$.

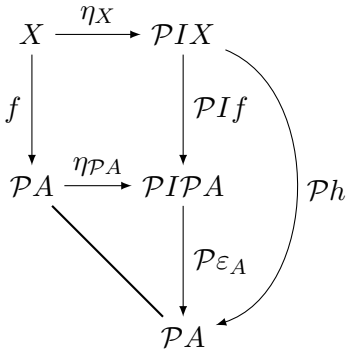
Now, we explain this using the standard adjunction $(I \dashv \mathcal{P}, \eta, \varepsilon)$ where the general proof of Proposition 2.3.26 can be found in [Vel17].

Example 2.3.27

Given the adjunction $(I \dashv \mathcal{P}, \eta, \varepsilon)$, in this example we construct for each function $f : X \to \mathcal{P}A \in \mathbf{Set}$ the unique arrow $h \in \mathbf{Rel}$ which makes the triangle of Proposition 2.3.26 commute. Therefore, we set

$$h := IX \xrightarrow{If} I\mathcal{P}A \xrightarrow{\varepsilon_A} A$$

with $If = \{(x, f(x)) \mid x \in X\}$ and we get $(A', a) \in \varepsilon_A \iff a \in A'$ from the counit. Summed up, we have a relation $h \subseteq IX \times A$ with $(x, a) \in h$ if $(f(x), a) \in \varepsilon_A$. Note, that due to η and ε given in Definition 2.3.24 we have $Id_{\mathcal{P}A} = \mathcal{P}\varepsilon_A \circ \eta_{\mathcal{P}A}$ (i.e. the triangle below commutes) and therefore $f = Id_{\mathcal{P}A} \circ f = \mathcal{P}\varepsilon_A \circ \eta_{\mathcal{P}A} \circ f$. In addition, the square commutes due to the naturality of η (i.e. $\eta_{\mathcal{P}A} \circ f = \mathcal{P}If \circ \eta_X$) and we can conclude that $f = \mathcal{P}h$ holds.



Furthermore, that h is unique can be shown via the naturality of ε and by the commuting triangles in Definition 2.3.24 (see [Vel17]). In the literature h is called the transpose of f.

Analogously to Proposition 2.3.26, the same result can be explored for each $h \in \mathbf{Rel}$

with $h : LX \to A$, where the construction of the transpose $f \in \mathbf{Set}$ is based on the unit η of the adjunction, i.e. $X \xrightarrow{\eta_X} RLX \xrightarrow{Rh} RA$. Thus, in summary the *one-to-one* correspondence between the arrows depicted in Schema (2.2) holds (for more details see [Vel17]).

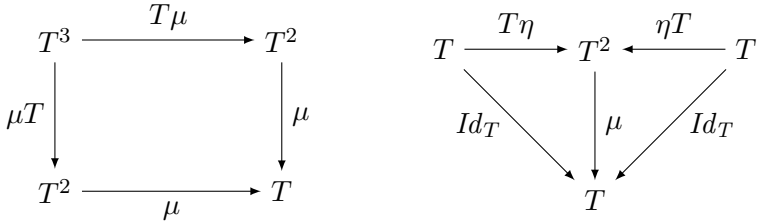
Moreover, from each adjunction $(L \dashv R, \eta, \varepsilon)$ we get a special mathematical structure called *monad* [Mac98; AHS09].

Definition 2.3.28: Monad

A monad on a category $\mathbf{C}$ is a triple $T = (T, \eta, \mu)$ where T is a endofunctor on $\mathbf{C}$ and η, μ are two natural transformations

$$\eta : ID_{\mathbf{C}} \to T \text{ and } \mu : T^2 \to T$$

such that the following diagrams commute:



As stated above, a monad can be derived from an adjunction. Here, we only give the proposition, the proof can be found in [AHS09].

Proposition 2.3.29

Let $L : \mathbf{C} \to \mathbf{D}$ and $R : \mathbf{D} \to \mathbf{C}$ be two adjoints and $(L \dashv R, \eta, \varepsilon)$ an adjunction. A monad (T, η, μ) is constructed as follows:

1. $T = RL : \mathbf{C} \to \mathbf{C}$
2. $\eta : ID_{\mathbf{C}} \to T$
3. $\mu = R\varepsilon L : T^2 \to T$

For the sake of completeness, it should be pointed out that from each monad one can also derive several adjunctions (see [Mac98]).

Example 2.3.30

Here, we transform the adjunction given in Example 2.3.25 into the covariant powerset monad $T = \mathcal{P}$.

1. $T = RL : \mathbf{Set} \to \mathbf{Rel} \to \mathbf{Set}$ where first each set is mapped to itself and each function $f : X \to Y$ to the corresponding relation R_f. Afterwards, the set is mapped to its powerset and the relation R_f to $\mathcal{P}f : \mathcal{P}X \to \mathcal{P}Y$ denoted with f_R in Example 2.3.25.
2. $\eta : ID_{\mathbf{C}} \to T$ as in Example 2.3.25.
3. $\mu = R\varepsilon L : T^2 \to T$ where the inclusion functor L maps X to the object $X \in \mathbf{Rel}$. The natural transformation ε maps this object to $\varepsilon_{L(X)} : \mathcal{P}X \to X$. Next, the right adjoint maps this relation to the function $f_{\varepsilon_{L(X)}} : \mathcal{P}^2X \to \mathcal{P}X$ with $\mu_X(A) = \cup A$.

Example 2.3.31

Another monad is given by the multiset functor [JSS15]. Let S be a semiring with a commutative additive monoid $(S, +, 0)$ and a multiplicative monoid $(S, \cdot, 1)$ where multiplication distributes over addition.
The multiset functor $\mathcal{M}_S : \mathbf{Set} \to \mathbf{Set}$ is defined as follows:

1. On objects X: $\mathcal{M}_S(X) = \{g : X \to S \mid \mathit{supp}(g) \text{ is finite}\}$ where $\mathit{supp}(g) = \{x \in X \mid g(x) \neq 0\}$.
2. On functions $f : X \to Y$: We define $\mathcal{M}_S(f) : \mathcal{M}_S(X) \to \mathcal{M}_S(Y)$ by $\mathcal{M}_S f(g)(y) = \sum_{x \in f^{-1}(y)} g(x)$

The multiset functor $\mathcal{M}_S$ is a monad $(\mathcal{M}_S, \eta, \mu)$ where the unit is given by $\eta_X : X \to \mathcal{M}_S(X)$ defined as follows:

$$\eta_X(x)(y) = \begin{cases} 1 & y = x \\ 0 & y \neq x \end{cases}$$

and the multiplication $\mu : \mathcal{M}_S(\mathcal{M}_S(X)) \to \mathcal{M}_S(X)$ is defined in the following way:

$$\mu(\Phi)(x) = \sum_{g \in \mathit{supp}(\Phi)} \Phi(g) \cdot g(x)$$

where $\Phi \in \mathcal{M}_S(\mathcal{M}_S(X))$.

Now, we have all the fundamental building blocks to create a category based on a monad. The so-called *Kleisli* categories originate from the fact, that each monad is given by an adjunction.

Definition 2.3.32: Kleisli Category

We denote the Kleisli category of a monad (T, η, μ) in $\mathbf{C}$ with $\mathbf{K}\boldsymbol{\ell}(T)$ which has the same objects as $\mathbf{C}$. For any two objects C and D, a Kleisli arrow $f : C \to D$ is a $\mathbf{C}$-arrow $f : C \to TD$.
Identity for any Kleisli object C is $\eta_C : C \to TC$ and composition of Kleisli arrows $f : C \to TD$, $g : D \to TE$ is defined based on the multiplication μ:

$$g \circ_T f = \mu_E \circ Tg \circ f$$

Example 2.3.33

In this example we construct the Kleisli category based on the powerset monad $\mathcal{P}$ presented in Example 2.3.30. The objects are given by the class of all sets. A morphism $X \to Y$ is given by a function $f : X \to \mathcal{P}Y$.
As mentioned in [HJS07], $\mathbf{K}\boldsymbol{\ell}(\mathcal{P})$ corresponds to the category **Rel**, since any function f can be interpreted as a relation and any relation of type $R \subseteq X \times Y$ can be converted into a function $f_R : X \to \mathcal{P}Y$ (cf. [Ker16]).

The idea to work with a Kleisli category instead of the category **Set** is motivated by the intuition that coalgebraic behavioural equivalence corresponds to another notion of behavioural equivalence than in **Set**. But before we can move to trace (language) equivalence, we need to show that an arrow $X \to FC \in \mathbf{K}\boldsymbol{\ell}(T)$ corresponding to a coalgebra $\alpha : C \to TFC \in \mathbf{C}$ is also a coalgebra in $\mathbf{K}\boldsymbol{\ell}(T)$. Therefore, we need to lift F to $\bar{F}$ such that the diagram in Figure 2.16 commutes.

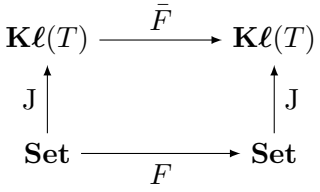
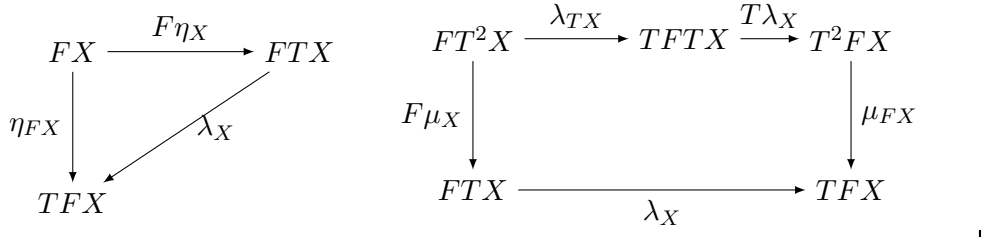


Figure 2.16: Lifting of F to $\bar{F}$ wrt. $\mathbf{K}\boldsymbol{\ell}(T)$.

Therefore, we refer to [HJS07; Mul94] and only present the necessary results here. Once again, we need a natural transformation:

Definition 2.3.34: Distributive Law

A distributive law λ is a natural transformation $\lambda : FT \to TF$ which is compatible with the monad structure of T in the following way:



To guarantee a lifting of F to the corresponding Kleisli category $\mathbf{K}\ell(T)$ where T is a monad on **Set**, such a distributive law is necessary and $\bar{F}$ is constructed as follows:

1. On objects: $\bar{F}X = FX$
2. On morphisms: Given $f : C \to D \in \mathbf{K}\ell(T)$ we get $\bar{F}f$ as follows

$$FC \xrightarrow{Ff} FTD \xrightarrow{\lambda_D} TFD \ \in \mathbf{Set}$$

It has been already shown that every polynomial endofunctor on **Set** has a canonical distributive law over any commutative monad on **Set** [HJS07, Lemma 2.4]. In addition, this result has been extended to analytic functors [MPS09].

Considering **Set** and the functor $\mathcal{P}$ the branching is *non-determinism* and it becomes clear that in Kleisli an arrow $X \to Y$ *hides* this branching (see [HJS07] for a very illustrative explanation of this phenomenon).

Where the lifting to Kleisli hides the specific side-effects (e.g. for $T = \mathcal{P}$ the non-determinism) without changing the branching type of the coalgebra, the *hiding* has an important effect on the coalgebra homomorphisms $h : X \to Y$. Now, for $T = \mathcal{P}$ a state is mapped to a subset of Y instead of a single state (as by a coalgebra homomorphism in **Set**).

How coalgebraic behavioural equivalence becomes language equivalence

In order to explore the results mentioned above on a concrete example, namely on non-deterministic (finite) automata (NDA), we consider a coalgebra $\alpha : X \to FX \in \mathbf{K}\ell(\mathcal{P})$ for the functor $F = A \times \_ + 1$ where $1 = \{\bullet\}$. Furthermore we recall from [PT99] (cf. [JSS15, Section 7.1]), that α can be *linearized* into a deterministic system $\bar{\alpha}$ as follows:

1. $\mathcal{P}X$ is the state space. (In [PT99] they omit the $\emptyset$ as a *state* due to a theorem about the extension of final coalgebras and they interpret a transition system with state space S as a deterministic one if for all $a \in A$ and all states there exists at most one a-successor state.)

2. An a-labelled transition $U \xrightarrow{a} U'$ is given by

$$U' = \{x' \mid \exists x \in U \text{ s.t. } x \xrightarrow{a} x'\}$$

   for each $a \in A$ and $U, U' \in \mathcal{P}X$.

3. U is terminating ($U\downarrow$) if there exists a $u \in U$ such that $\bullet \in \alpha(u)$.

We consider two systems $\alpha, \beta \in \mathbf{K}\boldsymbol{\ell}(\mathcal{P})$, the determinized versions $\bar{\alpha}, \bar{\beta}$ and a morphism $g : X \to Y \in \mathbf{K}\boldsymbol{\ell}(\mathcal{P})$ which is a function $g : X \to \mathcal{P}Y \in \mathbf{Set}$. We define the mapping of g to $g^{\#} : \mathcal{P}X \to \mathcal{P}Y \in \mathbf{Set}$ via $g^{\#} = \mu_Y \circ \mathcal{P}g$ as follows:

$$g^{\#}(X') = \bigcup_{x \in X'} g(x)$$

A function $g \in$ is a coalgebra homomorphisms iff for all $U \in \mathcal{P}X$ and all $a \in A$

$$g^{\#}(U) \xrightarrow{a} Y' \in \mathcal{P}(Y) \iff U \xrightarrow{a} U' \text{ and } Y' = g^{\#}(U')$$

where the transitions $\xrightarrow{a}$ are taken from $\bar{\beta}, \bar{\alpha}$ and

$$g^{\#}(U)\downarrow \iff \exists x \in U \text{ such that } \bullet \in \alpha(x).$$

hold [PT99]. Thus two states $X, X' \in \mathcal{P}X$ in the state space of the determinized version $\bar{\alpha}$ are interpreted as equivalent if there exists a coalgebra homomorphism $g \in \mathbf{K}\boldsymbol{\ell}(\mathcal{P})$ with $g^{\#}(X) = g^{\#}(X')$ (compare with Definition 2.3.17).

Now, we first give some example and later we summarize everything in a more intuitive statement.

Example 2.3.35

In this example we will consider a coalgebra homomorphism in $\mathbf{K}\boldsymbol{\ell}(\mathcal{P})$ between α on state space $X = \{x_1, \ldots, x_7\}$ and β on state space $Y = \{A, B, C\}$ (see Figure 2.17):

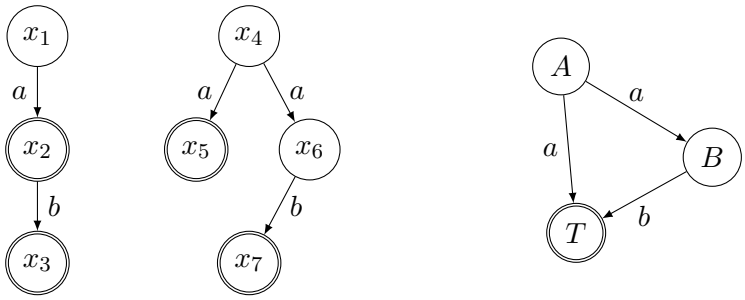


Figure 2.17: Two non-deterministic automata: x_1 and x_4 accept the same language $L = \{a, ab\}$. The system β on the right is a minimization of α.

Next, we define a relation $g : X \to Y$ as follows:

$$\begin{array}{lll} (x_1, A) \in g & (x_2, B) \in g & (x_3, T) \in g \\ (x_4, A) \in g & (x_2, T) \in g & (x_5, T) \in g \\ & (x_6, B) \in g & (x_7, T) \in g \end{array}$$

This relation g can be interpreted as a function $g : x \to \mathcal{P}Y \in \mathbf{Set}$ and since $g^\#$ satisfies the properties mentioned above, g is a coalgebra homomorphism. We can validate the condition for g to be a coalgebra homomorphism for each subset in $\mathcal{P}X$ but we just list the most interesting cases:

$$\begin{array}{lll} g^\#(\{x_1\}) = \{A\} : & \{A\} \xrightarrow{a} \{B, T\} & \\ \iff & \{x_1\} \xrightarrow{a} \{x_2\} & \text{with } g^\#(\{x_2\}) = \{B, T\} \\ g^\#(\{x_4\}) = \{A\} : & \{A\} \xrightarrow{a} \{B, T\} & \\ \iff & \{x_4\} \xrightarrow{a} \{x_5, x_6\} & \text{with } g^\#(\{x_5, x_6\}) = \{B, T\} \\ g^\#(\{x_2\}) = \{B, T\} : & \{B, T\} \xrightarrow{b} \{T\} & \\ \iff & \{x_2\} \xrightarrow{b} \{x_3\} & \text{with } g^\#(\{x_3\}) = \{T\} \\ \ldots & & \end{array}$$

Next, consider that $g^\#(\{x_1\}) = \{A\} = g^\#(\{x_4\})$ and therefore we derive the language equivalence for x_1, x_4. Moreover, for $x_1 \xrightarrow{a} x_2$ with $g(x_2) = \{B, T\}$ and $\alpha(x_4) = \{(a, x_5), (a, x_6)\}$ we observe that $g(x_2)$ with respect to a is present in the image of $Fg(\alpha(x_4))$ (i.e. $\{(a, T)\}, \{(a, B)\}$) where $Fg(a, x) = \{(a, y) \mid y \in g(x)\}$. Note, that the elements B, T are obtained through two different a-succesors of x_4 and not only by a single one. This is different for x_1 where one a-successor yields $\{(a, B), (a, T)\}$ but since the non-determinism is treated as a side-effect,

this branching-behaviour is ignored in general by the coalgebra homomorphisms in $\mathbf{K}\ell(\mathcal{P})$ (cf. [Küp17]). Therefore, modelling an NDA within Kleisli enables that behavioural equivalence coincides with language equivalence [PT99].

2.3.5 (Bi)fibrations and Indexed Categories

In this section we introduce *contravariant functors*. We want to emphasize that all the definitions, examples and explanations are taken from [LR19; Bén85; Jac99] and partly originate from the work by Alexander Grothendieck [Gro71; GR02].

Given a category $\mathbf{C}$ we consider a specific class of contravariant functors $\mathbf{C}^{op} \to \mathbf{Cat}$ where $\mathbf{Cat}$ is the category of all small categories (and the class of morphisms is given by the functors between these cats). Such a functor simply reverses the directions of the morphisms.

Definition 2.3.36: Contravariant Functor

A contravariant functor F from $\mathbf{C}$ to $\mathbf{D}$ is simply a functor from the opposite category $\mathbf{C}^{op}$.

The motivation to study such functors lies in the nature of predicate liftings which play a major role in logic and games (see Definition 3.2.9). These logical components can be organized via a functor $\mathbf{Set}^{\mathrm{op}} \xrightarrow{\tilde{\mathcal{Q}}} \mathbf{Cat}$ [Jac10], where $\tilde{\mathcal{Q}}X$ is the poset $(\mathcal{P}X, \subseteq)$ viewed as a category. In general, as observed by Jacobs [Jac10], *predicate logic on a category is given by an indexed category* and *predicate liftings are nothing but (endo)morphisms of indexed categories* [BK+20].

Definition 2.3.37: Indexed Categories [Jac10]

An indexed category is a contravariant functor Φ from $\mathbf{C}$ to $\mathbf{Cat}$. In addition, a morphism between two indexed categories $\mathbf{C}^{\mathrm{op}} \xrightarrow{\Phi} \mathbf{Cat}$ and $\mathbf{D}^{\mathrm{op}} \xrightarrow{\Psi} \mathbf{Cat}$ is a pair of a functor $\mathbf{C} \xrightarrow{G} \mathbf{D}$ and a natural transformation $\Phi \xrightarrow{\lambda} \Psi \circ G^{\mathrm{op}}$.
We denote the application of Φ on an arrow $f \in \mathbf{C}$ as $f^* = \Phi f$. We also omit the use of superscript 'op' on functors by writing them as contravariant functors. Moreover, we will use the phrases *indexed morphism* and *predicate lifting* interchangeably.

Example 2.3.38

As an example of an indexed category, consider the (contravariant) powerset functor $\Phi : \mathbf{Set}^{op} \to \mathbf{Cat}$ which maps a set X to the poset $(\mathcal{P}X, \subseteq)$.
Consider $F = \mathcal{P}$ over $\mathbf{C} = \mathbf{Set}$ which describes the branching type of unlabelled

transition systems. A simple indexed morphism $(\mathcal{P}, \lambda)$ of type $\Phi \xrightarrow{\lambda} \Phi \circ \mathcal{P}^{\mathrm{op}}$ is given by the natural transformation which assigns to each object $X \in \mathbf{Set}$ the following morphism:

$$\lambda_X(U) = \{U' \in \mathcal{P}X \mid U' \subseteq U\}$$

.

(It is well known that the above predicate lifting encodes the box modality $\Box$ from logic [Jac10].)

Via the so-called *Grothendieck* construction one can obtain a *fibration* from an indexed category. The crux of the matter here is: *indexed categories* or equivalently *fibrations* provide a categorical framework for predicate liftings. As Jacobs explains in [Jac10], the application of an *indexed category* on a state space X (i.e. ΦX) *may* contain more algebraic structure. (See [Jac99] for a detailed discussion of the link between type theory, logic, and fibrations.)

Before we introduce *Grothendieck fibrations* we need to understand the underlying base of these specific functors. The goal is to create functors between *fibres* and therefore the category has to admit *enough* cartesian morphisms:

Definition 2.3.39: P-cartesian [Jac99]

Given a functor $P : \mathbf{E} \to \mathbf{C}$, an arrow $g : E' \to E$ in $\mathbf{E}$ is P-cartesian over $u : I \to J$ in $\mathbf{C}$ if $Pg = u$ and every $g' : E'' \to E$ in $\mathbf{E}$ for which one has $Pg' = u \circ w$ for some $w : PE'' \to I$, uniquely determines an $h : E'' \to E$ above w with $g \circ h = g'$ (see Figure 2.18).

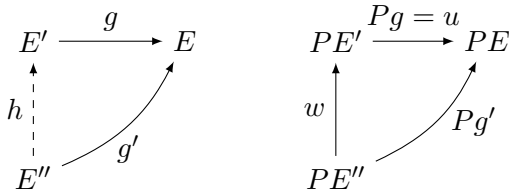


Figure 2.18: A cartesian arrow g wrt. the functor $P : \mathbf{E} \to \mathbf{C}$.

And *enough* means that our functor is cartesian for certain morphisms (cf. [Gra66, Definition 1.2, Definition 2.4]):

Definition 2.3.40: Fibration [Jac99]

A functor $P : \mathbf{E} \to \mathbf{C}$ is a fibration if for each $E \in \mathbf{E}$ and $u : I \to PE \in \mathbf{C}$, there is a cartesian morphism $f : E' \to E$ in $\mathbf{E}$ above u.
Similarly, P is an opfibration if $P : \mathbf{E}^{op} \to \mathbf{C}^{op}$ is a fibration.

Definition 2.3.41: Bifibration [Jac99]

We call a functor $P : \mathbf{E} \to \mathbf{C}$ that is both fibration and an opfibration an bifibration.

It is also known that fibrations, which are also bifibrations, satisfy a very specific property.

Definition 2.3.42: Fibre [Jac99]

Let $P : \mathbf{E} \to \mathbf{C}$ be a functor. For an object $C \in \mathbf{C}$ the fibre (or fibre category) $\mathbf{E}_C = P^{-1}C$ over C is the category with:

1. The objects are given by $X \in \mathbf{E}$ with $PX = C$
2. The morphisms $X \to Y \in \mathbf{E}_C$ are the arrows $f : X \to Y \in \mathbf{E}$ for which Pf is the identity map on $C \in \mathbf{C}$.

It is well known that a fibration is a bifibration iff each functor $f^* : E_C \to E_{C'}$ has a left adjoint $f_!$ which we call the *bifibration property*.

Finally, we introduce the so-called *Grothendieck* construction as well as the reverse transformation (cf. [Jac99, Section 1.4 and 1.10]):

C.1 *From indexed category to fibration:* Given an indexed category, i.e., a functor $\mathbf{C}^{\mathrm{op}} \xrightarrow{\Phi} \mathbf{Cat}$, then the so-called *category of elements* (aka the *Grothendieck construction*) $\mathbb{E}(\Phi)$ is defined as follows. The category $\mathbb{E}(\Phi)$ has pairs of the form (C, U) for each $C \in \mathbf{C}$ and $U \in \Phi(C)$ as objects. The arrows $(f, p) : (C, U) \to (C', U')$ of $\mathbb{E}(\Phi)$ are pairs of maps $C \xrightarrow{f} C' \in \mathbf{C}$ and $U \xrightarrow{p} \Phi f(U') \in \mathbf{\Phi} C$. In particular, given a functor $\mathbf{C}^{\mathrm{op}} \xrightarrow{\Phi} \mathbf{Cat}$ then the projection functor $\mathbb{E}(\Phi) \xrightarrow{\pi} \mathbf{C}$ mapping $(C, U) \mapsto C$ (for each $(C, U) \in \mathbb{E}(\Phi)$) forms a fibration (analogous for the arrows).

C.2 *From fibration to indexed category:* Note, that for a given fibration $P : \mathbf{E} \to \mathbf{C}$ the fibres $\mathbf{E}_C = P^{-1}(C)$ depend contravariantly on $C \in \mathbf{C}$. We require that the fibration is equipped with a *choice of Cartesian liftings* as described in [Jac99]. Thus, we have a so-called *substitution functor* u^* for a morphism $u \in \mathbf{C}$ between the fibres induced by u.

Next, we obtain a contravariant functor $\mathbf{C}^{op} \to \mathbf{Cat}$ which maps each $C \in \mathbf{C}$ to the category $\mathbf{E}_C = P^{-1}(C)$. For each $f : C \to C' \in \mathbf{C}$ define $f^* : \mathbf{E}_{C'} \to \mathbf{E}_C$ by mapping $E \in \mathbf{E}_{C'}$ to the domain of the unique lift of f with codomain E.

The indexed category given in Example 2.3.38 has the bifibration property given by the nature of the fibres in ΦC, which are just the subsets of an underlying state space $C \in \mathbf{Set}^{op}$. Next, we give an example of a well-known indexed category which fails to satisfy the bifibration property.

Definition 2.3.43: Meas [JS09]

The category **Meas** has measurable spaces (X, Σ_X) as objects. A measurable space is a set X equipped with a σ-algebra Σ_X.
Given two measurables spaces (X, Σ_X) and (Y, Σ_Y), a measurable function $f : (X, \Sigma_X) \to (Y, \Sigma_Y)$ is a function such that the preimage $f^{-1}(Y')$ is measurable in (X, Σ_X) whenever $Y' \subseteq Y$ is measurable in (Y, Σ_Y).

Example 2.3.44

Following the construction of an indexed category [Jac10; PS78] we obtain $\mathbf{Meas}^{op} \xrightarrow{\Phi} \mathbf{Cat}$ as follows:

1. Objects: (X, Σ_X) are mapped to the σ-algebra $(\Sigma_X, \subseteq)$ equipped with an order.

2. Morphisms: Any measurable function $f : (X, \Sigma_X) \to (Y, \Sigma_Y)$ is mapped to $f^{-1} : \Sigma_Y \to \Sigma_X$.

Proof: We want to show that Φ is a functor and therefore we need to prove that the identity functions $id_X : (X, \Sigma_X) \to (X, \Sigma_X)$ and reverse compositions are preserved:

1. For $id_X : (X, \Sigma_X) \to (X, \Sigma_X)$ with $\Phi(id_X) : \Sigma_X \xrightarrow{id_X^{-1}} \Sigma_X$ we obtain the same via $id_{\Phi X} : \Sigma_X \to \Sigma_X$ since $id_X^{-1}(X') = X'$ for any $X' \in \Sigma_X$.

2. For all measurable functions $f : (X, \Sigma_X) \to (Y, \Sigma_Y)$, $g : (Y, \Sigma_Y) \to (Z, \Sigma_Z)$ in **Meas** we get
$$f^{-1} : \Sigma_Y \to \Sigma_X \text{ and } g^{-1} : \Sigma_Z \to \Sigma_Y$$
Now, we reverse the direction for $f^{-1} \circ g^{-1}$ and get $\Sigma_Z \to \Sigma_Y \to \Sigma_X$ which is equal to $\Phi(g \circ f)$ given by the application of Φ to $\Sigma_X \xrightarrow{f} \Sigma_Y \xrightarrow{g} \Sigma_Z$.

□

Example 2.3.45

Now, we demonstrate why $\mathbf{Meas}^{op} \xrightarrow{\Phi} \mathbf{Cat}$ fails to admit the bifibration property. Therefore, we consider the following measurable spaces:

- $X = \{x, y, z\}$ with $\Sigma_X = \{\emptyset, X, \{x\}, \{y, z\}\}$ and
- $Y = \{0, 1\}$ with $\Sigma_Y = \{\emptyset, Y\}$.

Clearly, X, Y are measurable spaces. Consider the function f such that $f(x) = 0 = f(y)$ and $f(z) = 1$ where f is a measurable map. Unfortunately, a map $f_! : \Sigma_X \to \Sigma_Y$ such that $f_! X'$ becomes measurable whenever X' is measurable does not exist in general. The singleton set x is measurable in our example but

$$f_!\{x\} = \{0\}$$

and $\{0\}$ is a non-measurable set with respect to (Y, Σ_Y).

Example 2.3.45 shows that given an indexed category the left adjoint $f_!$ of f^* is not guaranteed and therefore one can not assume that in general the bifibration property holds.

3

Behavioural Equivalence: Games over Set

In concurrency theory the most frequently studied notion of behavioural equivalence is *bisimilarity* which is also the finest notion of the spectrum in [Gla01]. The term *bisimulation* first appeared when looking for answers to the following questions in the field of modal logic: "when is the truth of a modal formula preserved when the model changes?" or "which properties of models can modal logics express?" [San09]. This chapter considers the notion of bisimulation via game-theoretical characterizations based on coalgebraic modal logic.

3.1 Introduction

In the characterization of behavioural equivalences one encounters the following triad: First, such equivalences can be described via bisimulation relations, where the largest bisimulation (or bisimilarity) can be characterized as a greatest fixpoint. Second, a modal logic provides us with bisimulation-invariant formulas and the aim is to prove a Hennessy-Milner theorem which says that two states are behaviourally equivalent if and only if they satisfy the same formulas [HM80]. A third, complementary view is given by spoiler-duplicator games [Sti99]. Such games are useful both for theoretical reasons, see for instance the role of games in the Van Benthem/Rosen theorem [Ott04], or for didactical purposes, in particular for showing that two states are not behaviourally equivalent. The game starts with two tokens on two states and the spoiler tries to make a move that cannot be imitated by the duplicator. If the duplicator is always able to match the move of the spoiler we can infer that the two initial states are behaviourally equivalent. If the states are not equivalent, a strategy for the spoiler can be derived from a distinguishing modal logic formula.

Such games are common for standard labelled transition systems, but have been studied for other types of transition systems only to a lesser extent. For probabilistic transition systems there are game characterizations in [DLT08; FKP17], where the players can make moves to sets of states, rather than take a transition to a single state. Furthermore, in [CD08] a general theory of games is introduced in order to

characterize process equivalences of the linear/branching time spectrum.

Moreover, there are many contexts in which it is useful to check whether two system states are behaviourally equivalent respectively bisimilar. In this way one can compare a system with its specification, replace a subsystem by another one that is behaviourally equivalent or minimize a transition system (see also [MS19] for the study of algebraic effects).

Two states are bisimilar if they are related by a bisimulation relation. But this definition does not provide us with an immediate witness for non-bisimilarity, since we would have to enumerate all relations including that particular pair of states and show that they are not bisimulations. In games a proof of the non-bisimilarity of two states is given by a winning strategy of the spoiler. In logic the Hennessy-Milner theorem [HM85] guarantees for image-finite labelled transition systems that, given two non-bisimilar states x, y, there exists a modal logic formula φ such that one of the states satisfies φ and the other does not. The computation of such distinguishing formulas is explained in [Cle90].

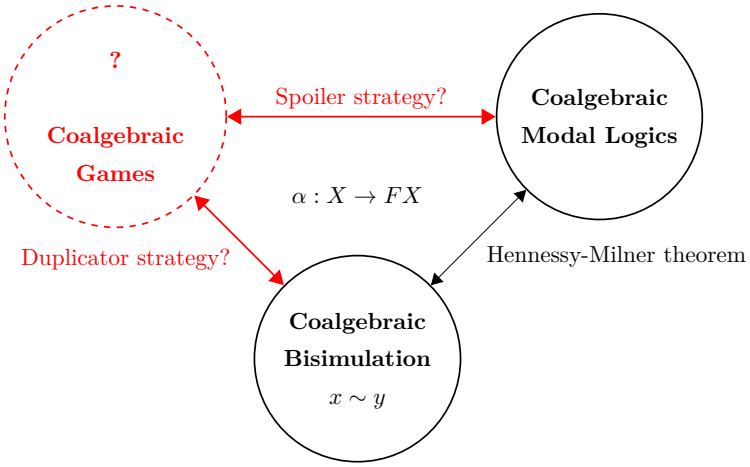


Figure 3.1: The coalgebraic triad capturing coalgebraic bisimulation and modal logics for $\alpha : X \to FX$ over endofunctors $F : \mathbf{Set} \to \mathbf{Set}$, where F specifies the branching type of a system α.

While the results and techniques above have been introduced for labelled transition systems, we are here interested in the more general setting of coalgebras [Rut00], which encompass various types of transition systems. The open questions studied in this chapter are visualized in Figure 3.1 and the contributions are:

- Firstly, we work in the general framework of coalgebras [Rut00], which allows to specify and uniformly reason about systems of different branching types (e.g. non-deterministic, probabilistic or weighted), parameterized over a functor.

While behavioural equivalences [Sta09] and modal logics [Sch08; Pat03] have been extensively studied in this setting, there are less contributions when it comes to games.

- Secondly, we will concentrate on coalgebraic methods for explaining that two given states in a transition system are *not* bisimilar. The idea is to provide a witness in form of a modal logic formula for non-bisimilarity. Such a witness can be used to explain (to the user) why an implementation does not conform to a specification and give further insights for adjusting it.

Apart from the introduction, this chapter is divided into four sections, where the first Section 3.2 introduces the necessary background information. The second Section 3.3 presents a coalgebraic game related to modal logics, which we denote as *the classical case* within this thesis. The second contribution is handled in the third Section 3.4 and finally, we close this chapter with a summary and conclusion.

3.2 Foundations for the Classical Case

This elementary section serves as a short introduction into game characterizations and category theory, where only the most relevant definitions are provided that specify the framework for our generic game. For more background information regarding the basics of category theory we refer to Section 2.3.

We study coalgebras in **Set** and therefore we introduce coalgebraic modal logic. But first, we start with the following section, which briefly discusses the game characterization of bisimulation and modal logics in the context of labelled transitions systems (LTS).

3.2.1 The Triad for Labelled Transition Systems

Bisimulation for labelled transition systems (see Definition 2.2.8) can be specified in terms of a two-player game. The spoiler (**S**) pursues to demonstrate the non-bisimilarity of two states x, y, while the duplicator (**D**) tries to show the opposite. To capture bisimulation, the rules of the game force the duplicator to mimic each move of the spoiler [Sti99].

Definition 3.2.1: Game for LTS

The initial situation is given by a labelled transition system $(X, \Sigma, \to)$ and a position $(x, y) \in X \times X$. (Note that $x = y$ is allowed.) From a position (x, y), the game play proceeds as follows:

- **Step 1: S** chooses one possible transition $x \xrightarrow{a} x'$ or $y \xrightarrow{a} y'$.
- **Step 2: D** must mimic the move of **S** depending on the move by the spoiler, i.e. $y \xrightarrow{a} y'$ or $x \xrightarrow{a} x'$.
- **Step 3:** The game continues with (x', y') in Step 1.

Indeed, the spoiler wins if the duplicator is not able to match his move in Step 2. On the other side the duplicator wins, if the game goes on forever or a pair of terminating (i.e. dead) states is reached. We omit the soundness and correctness proofs of the game and refer to [Sti99].

Next, we want to introduce a modal logic known under the name *Hennessy-Milner* logic.

Definition 3.2.2: Hennessy-Milner Logic (syntax)

A formula φ in Hennessy-Milner is inductively defined as follows:

$$\varphi \ ::= \ \mathbf{t} \ \mid \ \mathbf{f} \ \mid \ \neg\varphi \mid \ \varphi_1 \wedge \varphi_2 \ \mid \ \varphi_1 \vee \varphi_2 \ \mid \ \Box\varphi \ \mid \ \Diamond\varphi$$

The interpretation of a formula is a characteristic function $[\![\varphi]\!] : X \to \{0, 1\}$ and depends on the transition system $(X, \Sigma, \to)$. Characteristic functions can be also interpreted as subsets $\hat{[\![\varphi]\!]} \subseteq X$ and we sometimes use this representations interchangeable. We write $x \vDash \varphi$ in case x satisfies φ (i.e. $x \in [\![\varphi]\!]$). The formula $\mathbf{t}$ is satisfied by all states, where $\mathbf{f}$ does not hold for any state. Conjunction and disjunction work as usual. The *semantics* for $\Box$ (*box*) and $\Diamond$ (*diamond*) are evaluated over an LTS as follows:

$$[\![\Box\varphi]\!] = \{x \in X \mid \forall y \ (x \to y \ \Rightarrow y \in [\![\varphi]\!])\}$$

$$[\![\Diamond\varphi]\!] = \{x \in X \mid \exists y \ (x \to y \ \wedge y \in [\![\varphi]\!])\}$$

A logic is sound, if two bisimilar states (x, y) (i.e $x \sim y$) satisfy the same formulas. A logic is complete, if for each state pair (x, y), where both states satisfy the same formulas or, in other words, there exists no distinguishing formula $\varphi_{x,y}$ with $x \models \varphi_{x,y}$ and $y \not\models \varphi_{x,y}$, $x \sim y$ holds.

Theorem 3.2.3: Hennessy-Milner [HM85]

Let $(X, \Sigma, \to)$ be any image-finite (= finitely-branching) LTS and $x, y \in X$. Then $x \sim y$ if and only if for every **HM** formula φ: $x \models \varphi \iff y \models \varphi$.

In case a logic is sound and complete, it is convenient to say, that the logic is adequate and satisfies the *Hennessy-Milner* property [KL09]. We conclude that we have a triad: Bisimulation, **HM** logic, and a spoiler-duplicator game.

Furthermore, we list some notations used within this chapter. Let $R \subseteq X \times X$ be an *equivalence relation*, where the set of all equivalence relations on X is given by $Eq(X)$. By $E(R)$ we denote the set of all equivalence classes of R. Given $Y \subseteq X$, we define the *R-closure* of Y as follows: $[Y]_R = \{y \in X \mid \exists\, x \in Y\, (x, y) \in R\}$.

Besides, we will sometimes overload the notation and for instance write $[p]_R$ for the R-closure of a predicate p. Furthermore we will write $p_1 \cap p_2$ for the intersection of two predicates.

3.2.2 Categorical Foundations for the Classical Case

We restrict our setting to the category **Set**, where the objects are sets denoted with the class $Ob(\mathbf{Set})$ of all sets and the class of morphisms consists of all functions $hom(C, D)$ given for each pair of objects (C, D). Therefore, we assume an *endo-functor* $F\colon \mathbf{Set} \to \mathbf{Set}$, intuitively describing the branching type of the transition system under consideration. For more information we refer to Chapter 2, where the categorical basics are described in Section 2.3.

Since we are particularly interested in the behaviour of system states, we introduce the following coalgebraical definition to capture behavioural equivalence over **Set**, which is a special case of Definition 2.3.17. This definition depends on a function, which maps states to the same entity if their transition structure characterizes the same observable behaviour. Imagine you have some transition system $\alpha : X \to FX$ and two states have the same behaviour, then it is fine to merge them into one state and by doing so one obtains another system $\beta : Y \to FY$. From this, it follows, that two states $x, y \in X$ are behaviourally equivalent if another system over a state space Y exists, such that both are mapped to the same state in Y:

Definition 3.2.4: Behavioural Equivalence(cf. [Kur00])

Two states $x, y \in X$ are *behaviourally equivalent* ($x \sim y$) if there exists a coalgebra homomorphism f from $\alpha : X \to FX$ to some coalgebra $\beta\colon Y \to FY$ (i.e., a function $f\colon X \to Y$ with $\beta \circ f = Ff \circ \alpha$) such that $f(x) = f(y)$.

In addition, we assume that F preserves weak pullbacks, which means that behavioural equivalence and coalgebraic bisimilarity coincide, and therefore we will use the two terms interchangeably (see discussion on Theorem 2.3.21).

Furthermore we need to *lift* preorders under a functor F. To this end, we use the

lifting introduced in [BK11] (essentially the standard *Barr extension* of F [Bar70; Trn80]), which guarantees that the lifted relation is again a preorder provided that F preserves weak pullbacks:

Definition 3.2.5: Preorder Lifting

Let $\leq$ be a preorder on Y, i.e. $\leq \subseteq Y \times Y$. We define the preorder $\leq^F \subseteq FY \times FY$ as follows: given $t_0, t_1 \in FY$, it holds that $t_0 \leq^F t_1$ whenever there exists some $t \in F \leq$ such that $F\pi_i(t) = t_i$, where $\pi_i \colon \leq \to Y$ with $i \in \{0, 1\}$ are the usual projections.

Example 3.2.6

Actually, we are interested in lifting the order on $2 = \{0, 1\}$ given by $\leq = \{(0, 0), (0, 1), (1, 1)\}$.

In the case of the distribution functor $\mathcal{D}$ (see Definition 3 of Example 2.3.5) we have $\mathcal{D}2 \cong [0, 1]$ with $\mathcal{D}2 = \{f : 2 \to [0, 1] \mid f(0) + f(1) = 1\}$ and for each $f \in \mathcal{D}2$ with $f(1) = r$ we have $f(0) = 1 - r$. And thus, $\leq^{\mathcal{D}}$ corresponds to the order on the reals: for any two reals r_1, r_2 with $r_1 \leq r_2$ one can derive a $t \in \mathcal{D} \leq$ such that we get two functions $\mathcal{D}\pi_i(t) = f_i$ with $i \in \{1, 2\}$ and $f_i(1) = r_i$. But, for any pair of functions $f_1, f_2 \in \mathcal{D}2$ with $f_1(1) > f_2(1)$ a compatible $t \in F \leq$ does not exist since negative values are excluded in interval $[0, 1]$.

For the powerset functor $\mathcal{P}$ we obtain the order $\{0\} \leq^{\mathcal{P}} \{0, 1\} \leq^{\mathcal{P}} \{1\}$ where $\emptyset$ is only related to itself, so this corresponds to the Egli-Milner order (cf. [BK11, Section 3.2]).

At this point, we specify how this works for LTS over the alphabet $A = \{a, b\}$.

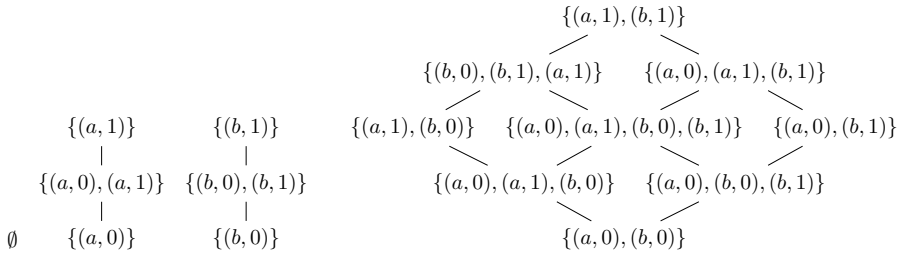


Figure 3.2: Preorder lifting of $\{(0, 0), (0, 1), (1, 1)\}$ with $F2 = \mathcal{P}_f(\{a, b\} \times 2)$ results in the order $\leq^F$ for labelled transition systems over the label set $A = \{a, b\}$.

In Figure 3.2 the lifting of $\leq$ results in an order consisting of the union of four disjoint partial orders: one for $\emptyset$ corresponding to the absence of any transition, one for transitions restricted to a, analogue to that one for *b-transitons* and the

fourth models the possibility of both *a*- and *b-transitons*.

The next lemma focuses on the order $\leq = \{(0,0),(0,1),(1,1)\}$ and is a special case of the results in [BK11].

Lemma 3.2.7

Consider our preorder $\leq$ over $2 = \{0,1\}$. Whenever a functor F preserves weak pullbacks $\leq^F$ is transitive.

Proof: For arbitrary relations $R \subseteq X \times Y, S \subseteq Y \times Z$ we have $R^F \circ S^F = (R \circ S)^F$ iff F preserves weak pullbacks [BK11] and with $\_^F$ we denote the standard *Barr extension* of F [Bar70; Trn80].

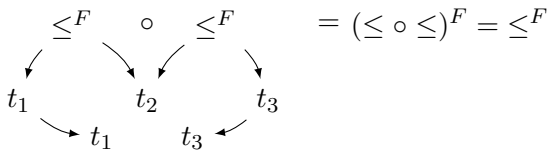


Figure 3.3: In case F preserves weak pullbacks the lifting $\leq^F$ is transitive.

Note, that $\leq = \leq \circ \leq$ holds and therefore we get $\leq^F \circ \leq^F = (\leq \circ \leq)^F = \leq^F$. Assume $(t_1,t_2),(t_2,t_3) \in \leq^F$ and therefore $(t_1,t_3) \in \leq^F \circ \leq^F$. Finally, we can conclude that $(t_1,t_3) \in \leq^F$ since the composition of the lifted relations is equal to the lifted relation composition (see Figure 3.3). □

Note that applying the functor is monotone with respect to the lifted order:

Lemma 3.2.8

Let $(Y,\leq)$ be an ordered set and let $p_1,p_2 \colon X \to Y$ be two functions. Then $p_1 \leq p_2$ implies $Fp_1 \leq^F Fp_2$ (with both inequalities read pointwise).

Proof: We have two predicates $p_1 \leq p_2$. Define $g \colon X \to \leq$ as follows: $g(x) = (p_1(x),p_2(x)) \in \leq$, which implies $\pi_i \circ g = p_i$.

Let $s \in FX$ and we have to show that $Fp_1(s) \leq^F Fp_2(s)$. For this, we require the existence of some $t \in F \leq$ with $F\pi_i(t) = Fp_i(s)$ for $i \in \{1,2\}$. Set $t = Fg(s)$, then it holds that $F\pi_i(t) = F\pi_i(Fg(s)) = F(\pi_i \circ g)(s) = Fp_i(s)$. □

Later we will see how the lifting of the order $\leq$ fits into our generic game. But to define coalgebraic modal logics, we need the notion of *predicate liftings* (also called *modalities*). A predicate lifting can be seen as a *tool* which enables us to analyze a state after performing an α-transition [Pat04].

For the sake of completeness, we give here the more common definition of a predicate lifting, but later we will switch to an alternative representation, which better fits to our game characterization.

Definition 3.2.9: Predicate Lifting [Pat03; GS13]

A predicate lifting for a functor F is a natural transformation

$$\lambda \colon \mathcal{Q} \to \mathcal{Q} \circ F^{\mathrm{op}}$$

with $\mathcal{Q}$ denotes the contravariant powerset functor $\mathcal{Q} : \mathbf{Set}^{\mathrm{op}} \to \mathbf{Set}$ (every set X is mapped to $\mathcal{P}X$ and for each $f : X \to Y$ we have $\mathcal{Q}f : \mathcal{P}Y \to \mathcal{P}X$ with $\mathcal{Q}f(B) = f^{-1}[B]$). Explicitly a predicate lifting assigns to each $A \subseteq X$ a set $\lambda_X(A) \subseteq FX$ such that following diagram commutes for all $f : X \to Y$:

$$\begin{array}{ccc} \mathcal{Q}Y & \xrightarrow{\lambda_Y} & \mathcal{Q}(FY) \\ {\scriptstyle f^{-1}}\downarrow & & \downarrow{\scriptstyle (Ff)^{-1}} \\ \mathcal{Q}X & \xrightarrow{\lambda_X} & \mathcal{Q}(FX) \end{array}$$

Note that we sometimes omit to write F^{op} when it is clear from the context.

Taking into account that each $Y \subseteq X$ corresponds to a *characteristic function*, we denote its *predicate* or *characteristic function* by $\chi_Y \colon X \to \{0, 1\}$. Furthermore, given a characteristic function $\chi \colon X \to \{0, 1\}$, its corresponding set is denoted $\hat{\chi} \subseteq X$.

We now switch to so-called *evaluation maps*, where the one-to-one correspondence between *predicate liftings* (as in Definition 3.2.9) and subsets of $F2$ is spelled out in [Sch08].

Example 3.2.10

Given a set X and the set $\mathcal{P}2$ for the (finite) powerset functor $\mathcal{P}$ we give a concrete example for such a one-to-one correspondence.
Since $|\mathcal{P}2| = 4$ we have 16 subsets of $\mathcal{P}2$ and by [Sch08] we get 16 predicate liftings. To see how a predicate lifting can be seen as a subset of $\mathcal{P}2$ we consider the following predicate lifing λ:

$$\lambda(X') = \{Y \mid Y \subseteq X'\}$$

where $X' \subseteq X$. Obviously, this predicate lifting corresponds to the subset

$\{\emptyset, \{1\}\} \subset \mathcal{P}2$ since for the corresponding evaluation map $ev : \mathcal{P}2 \to 2$ we have $ev \circ Fp(l) = 1$ only if $l \subseteq \hat{p}$. Moreover, this predicate lifting induces the $\Box$ operator of the Hennessy-Milner logic introduced in Definition 3.2.2.

As the branching type plays a major role in the whole story, it also makes sense to apply the functor on $p : X \to 2$ to derive $Fp : FX \to F2$. Now, combining $Fp(\alpha(x))$ where $x \in X$ is a state of interest, we get an element in $F2$. This trivially results in the fact, that we need to distinguish such elements and therefore consider so-called *evaluation maps*, which specify subsets of $F2$ via characteristic functions $ev\colon F2 \to 2$.

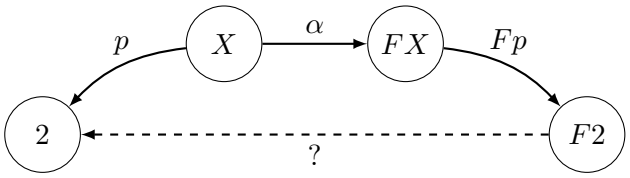


Figure 3.4: Given a coalgebra $\alpha\colon X \to FX$ and a predicate $p : X \to 2$ it appears purposeful to characterize predicate liftings via evaluation maps $ev\colon F2 \to 2$.

Thus, each predicate $p\colon X \to 2$ is lifted to a predicate $FX \to 2$ via $ev \circ Fp\colon FX \to 2$. Since *predicates* will also play a significant role in the generic game rules we switch from natural transformations to evaluation maps to obtain a uniform presentation of the material.

In the next section we introduce an adequate and expressive coalgebraic logic build on special sets of *evaluation maps*.

3.2.3 Coalgebraic Modal Logics for the Classical Case

In the scope of our framework a logic is adequate and complete (or has the Hennessy-Milner property 2.2.8) if and only if there exists no formula which distinguishes two behaviourally equivalent states. This means, that two states $x, y \in X$ satisfy the same formulas if and only if they are also behaviour equivalent, i.e. a coalgebra homomorphism $X \to Y$ exists such that $f(x) = f(y)$ holds.

For LTS, the Hennessy-Milner theorem requires a finitely branching system to ensure that a non-bisimilar state pair (x, y) is distinguishable at least by one formula $\varphi_{x,y}$ of finite length, i.e $x \models \varphi_{x,y}$ and $y \not\models \varphi_{x,y}$.

Naturally, it follows that we restrict to a coalgebraic logic with formulas of bounded conjunctions. More precisely, we want the size of the conjunctions to be big enough with respect to the branching described by F. This size is derived from the so-called

accessible-property of a functor originating from the work by Worrel [Wor05] in terms of behavioural equivalence [Pat04].

Definition 3.2.11: κ-accessible

A functor $F\colon \mathbf{Set} \to \mathbf{Set}$ is κ-accessible if for all sets X and all $x \in FX$ there exists $Z \subseteq X$, $|Z| < \kappa$ such that $x \in FZ \subseteq FX$ [AGT10]. (Note that we use the fact that **Set**-functors preserve injections $f\colon A \to B$ whenever $A \neq \emptyset$.) For details and a more categorical treatment see [AR94; Pat04].

Intuitively Pattinson says, an endofunctor F on **Set** is κ-accessible if the application of F on a set X depends on the action of F on subsets $X' \subseteq X$ with cardinality less than κ [Pat04].

Example 3.2.12

The **Set** functor $\mathcal{P}_f$ is ω- accessible.

Another necessary property satisfied by the modalities given in the Hennessy-Milner logic is called *separation*.

Definition 3.2.13: Separation [Pat04]

A set Λ of evaluation maps is separating for a functor $F\colon \mathbf{Set} \to \mathbf{Set}$ whenever for all sets X and $t_0, t_1 \in FX$ with $t_0 \neq t_1$ there exists $ev \in \Lambda$ and $p\colon X \to 2$ such that $ev(Fp(t_0)) \neq ev(Fp(t_1))$.

This means that every $t \in FX$ is uniquely determined by the set $\{(ev, p) \mid ev \in \Lambda, p\colon X \to 2, ev(Fp(t)) = 1\}$. Such a separating set of predicate liftings exists iff $(Fp\colon FX \to F2)_{p\colon X \to 2}$ is jointly injective [Sch08].

Example 3.2.14

For LTS already one evaluation map induced by $\Box$ or $\Diamond$ forms a separating set of predicate liftings.

Here we concentrate on *unary* predicate liftings: If one generalizes to *polyadic* predicate liftings, separation can be shown for every accessible functor [Sch08].

Moreover, $\Box$ and $\Diamond$ are monotone maps $ev : (F2, \leq^F) \to (2, \leq)$ (i.e. order-preserving). The relation between monotone modalities and monotone predicate liftings is already mentioned in [Sch08], where modalities induced from predicate liftings are considered. We provide a similar result here, only additionally we

involve $\leq^F$.

Proposition 3.2.15

Let $ev\colon FV \to V$ be an evaluation map, mapping to $(V, \leq)$. It corresponds to a monotone predicate lifting $(p\colon X \to V) \mapsto (ev \circ Fp\colon FX \to V)$ iff $ev\colon (FV, \leq^F) \to (V, \leq)$ is monotone.

Proof:

($\Leftarrow$) Assume that the predicate lifting is monotone, i.e., given two predicates $p_1 \leq p_2$ it holds that $ev \circ Fp_1 \leq ev \circ Fp_2$. In order to show monotonicity of ev take $v_1, v_2 \in FV$ such that $v_1 \leq^F v_2$. This means that there exists $r \in F \leq$ such that $F(\pi_1 \circ o)(r) = v_1$, $F(\pi_2 \circ o)(r) = v_2$, where $o\colon \leq \to V \times V$ is the embedding of the order into $V \times V$.

Now consider $\pi_1 \circ o$, $\pi_2 \circ o\colon \leq \to V$. It holds that $\pi_1 \circ o \leq \pi_2 \circ o$ and with monotonicity of the lifting we can conclude $ev \circ F(\pi_1 \circ o) \leq ev \circ F(\pi_2 \circ o)$.

Hence $ev(v_1) = ev(F(\pi_1 \circ o)(r)) \leq ev(F(\pi_2 \circ o)(r)) = ev(v_2)$, i.e., we have shown that ev is monotone.

($\Rightarrow$) Assume that ev is monotone and take $p_1, p_2\colon X \to V$ such that $p_1 \leq p_2$.

Hence $Fp_1(t) \leq^F Fp_2(t)$ by Lemma 3.2.8. Then we can conclude that $ev(Fp_1(t)) \leq ev(Fp_2(t))$, using the monotonicity of ev. □

We require that the lifting of $\leq$ is anti-symmetric i.e. for $t_0, t_1 \in F2$ it holds that $t_0 = t_1$ iff $t_0 \leq^F t_1$ and $t_1 \leq^F t_0$. In order to understand the relevance of this assumption we consider the following Example 3.2.16:

Example 3.2.16

The transition system given in Figure 3.5 serves as an example for a functor, where the lifted order is not anti-symmetric.

Let $F = \mathbb{R}_0 \times (\mathbb{R}_{\overline{0}})^A_\omega$ and x, y be two behaviourally different states. The state x associates with the word $w = ab$ the weight 2, but y assigns -2 to w. Note, all states terminate with weight 1 and therefore this transitions are omitted in the visual presentation. Recall, that for $t \in F2$ we denote with t_2 the second component of the tuple t and therefore $t_2(a)$ refers to the application of t_2 to the label $a \in A$.

Obviously, 2 and 7 are behaviourally equivalent and we consider the two values $s = Fp(\alpha(x)), t = Fp(\alpha(y))$ for $\hat{p} = \{2, 7\}$, where p is a characteristic function with $p(2) = p(7) = 1$ and $p(z) = 0$ for $z \in X \setminus \{2, 7\}$.

For simplicity we restrict to the a-transitions of s_2 and t_2, since x and y behave

the same for b, c no matter which predicate we use for the evaluation. We can also ignore the termination behaviour, since all states terminate with weight 1. Thus, we consider $t \neq s$ with $s_2(a) = [0 \to -2, 1 \to 2]$ and $t_2(a) = [0 \to 2, 1 \to -2]$, where the b, c-behaviour is given by $s_2(l) = t_2(l) = [0 \to 0, 1 \to 0] \in \mathbb{R}_0^2$ for $l \in \{b, c\}$ and therefore we define $m = [(0,0) \to 0, (0,1) \to 0, (1,1) \to 0]$. We observer that $s \leq^F t$ and $t \leq^F s$ are induced respectively by

$$(1, a \mapsto [(0,0) \to 2, (0,1) \to -4, (1,1) \to 2], b \mapsto m, c \mapsto m) \in F \leq$$

$$(1, a \mapsto [(0,0) \to -2, (0,1) \to 4, (1,1) \to -2], b \mapsto m, c \mapsto m) \in F \leq$$

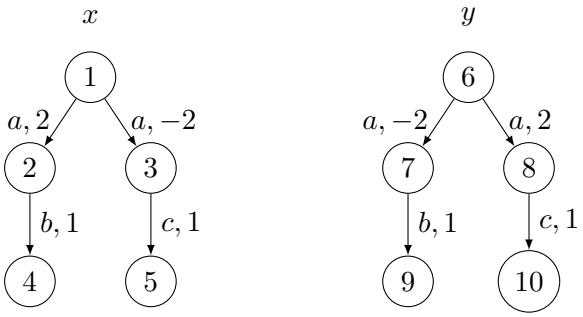


Figure 3.5: The inequivalent behaviour of x and y can not be distinguished by the lifted order $\leq^F$

As a consequence, $s, t \in F2$ are not distinguishable by $\leq^F$. Furthermore, a separating set of predicate liftings could not be monotone, since x and y behave different regarding 2 and 7, because x has an a-transition to state 2 with weight 2, where y has an a-transition to state 7 with weight -2. To distinguish $\alpha(x)$ and $\alpha(y)$ we need a pair (ev, p) such that $ev \circ Fp(\alpha(x)) \neq ev \circ Fp(\alpha(y))$. All suitable options for such a p are given in the following list, where only the states $2, 3, 7, 8$ play a role and we set $p(z) = 0$ for all other $z \in X$.
The remaining six predicates $p : X \to 2$ which yield $Fp(\alpha(x)) = Fp(\alpha(y))$ disable a separation via an evaluation map and are therefore not included in our list:

$$\hat{p} = \{2\} \qquad \hat{p} = \{2,7\} \qquad \hat{p} = \{2,3,7\} \qquad \hat{p} = \{2,3,8\} \qquad \hat{p} = \{8\}$$
$$\hat{p} = \{7\} \qquad \hat{p} = \{3,8\} \qquad \hat{p} = \{2,7,8\} \qquad \hat{p} = \{3,7,8\} \qquad \hat{p} = \{3\}$$

No matter which p we consider from the list above we get $s = Fp(\alpha(x)) \neq Fp(\alpha(y)) = t$ and based on this we can define a separating evaluation map $ev : \mathbb{R}_0 \times (\mathbb{R}_0^2)_\omega^A \to 2$, which distinguishes the two values s, t with $ev(s) = 1$ and

$ev(t) = 0$ or $ev(s) = 0$ and $ev(t) = 1$. But such an evaluation map ev can not be monotone due to the fact that $s \leq^F t$ and $t \leq^F s$ hold. We can obtain such a result for any p with $s = Fp(\alpha(x)) \neq Fp(\alpha(y)) = t$ since we can show that $s \leq^F t$ and $t \leq^F s$ are each induced by some $t \in F \leq$.
As a consequence, the pair $\alpha(x), \alpha(y) \in FX$ serves as a counterexample for the existence of a monotone and separating predicate lifting ev for this functor: in case ev is separating it violates the monotonicity due to the non-antisymmetry of the lifted preorder $\leq^F$.

De facto, this observation suggests that there are no monotone evaluation maps $ev : F2 \to 2$ in case the lifted order is not anti-symmetric.

Proposition 3.2.17

F has a separating set of monotone predicate liftings iff $\leq^F \subseteq F2 \times F2$ is anti-symmetric and $(Fp\colon FX \to F2)_{p\colon X \to 2}$ is jointly injective.

Proof:
($\Rightarrow$) It follows directly from [Sch08] that $\{Fp\colon FX \to F2\}_{p\colon X \to 2}$ is jointly injective.

We now show that $\leq^F$ is anti-symmetric. Let $t_1, t_2 \in F2$ with $t_1 \leq^F t_2 \leq^F t_1$ and $t_1 \neq t_2$. Then there exists a monotone $ev \in \Lambda$ and some $p : 2 \to 2$ such that $ev \circ Fp(t_1) \neq ev \circ Fp(t_2)$. We consider the following two cases:

- *p is monotone:* In this case Fp is also monotone, which can be shown as follows. Assume that $s_1 \leq^F s_2$, so there exists some $s \in F \leq$, such that $F\pi_i(s) = s_i$. Now, since p is monotone, $p \times p$ can be restricted to $p \times p\colon \leq \to \leq$. Furthermore $F\pi_i \circ F(p \times p)(s) = Fp \circ F\pi_i(s)$ for $i = 1, 2$ and $t' = F(p \times p)(s) \in F \leq$ is a witness for $Fp(s_1) \leq^F Fp(s_2)$. Now, since ev is also monotone, $t_1 \leq^F t_2 \leq^F t_1$ implies $ev \circ Fp(t_1) \leq ev \circ Fp(t_2) \leq ev \circ Fp(t_1)$, hence $ev \circ Fp(t_1) = ev \circ Fp(t_2)$, which is a contradiction.

- *p is not monotone:* since $p : 2 \to 2$, the only non-monotone such predicate is $p(0) = 1$, $p(1) = 0$, which is antitone. Similar to above we can argue that Fp is antitone as well and complete the argument. (Here $p \times p\colon \leq \to \geq$ and we choose $s' = F(sym \circ (p \times p))(s)$, where $sym\colon 2 \times 2 \to 2 \times 2$ switches its arguments.)

($\Leftarrow$) We assume that $\leq^F$ is anti-symmetric and that there exists a separating set of evaluation maps, i.e., $\{Fp\colon FX \to F2\}_{p\colon X \to 2}$ is jointly injective. Now let $t_1 \neq t_2$ and $t_1, t_2 \in F2$. Since $\{Fp\}_p$ is jointly injective, there exists a predicate such that

$Fp(t_1) \neq Fp(t_2)$. Due to the antisymmetry of $\leq^F$ two cases can occur and we show that in each case there exists a monotone evaluation map which separates the two:

- $Fp(t_1) \not\leq^F Fp(t_2)$: We define $ev\colon F2 \to 2$ as follows:

$$ev(t) = \begin{cases} 0 & \text{if } t \leq^F Fp(t_2) \\ 1 & \text{otherwise} \end{cases} \tag{3.1}$$

  Note that ev is monotone and that $ev(Fp(t_2)) = 0$, $ev(Fp(t_1)) = 1$.

- $Fp(t_2) \not\leq^F Fp(t_1)$: analogously.

□

We demonstrate the link between the **HM**-modalities $\Box$, $\Diamond$ and the lifted order $\leq^F$ based on $F \leq= \mathcal{P}_f(\{a,b\} \times \{(0,0),(0,1),(1,1)\})$ with $FX = \mathcal{P}_f(\{a,b\} \times X)$ in a more graphical way (Figure 3.6). The monotonicity of the modalities is observable in the corresponding cut through the elements $F2$, more precisely, the $\Box$ and $\Diamond$ sets are upward-closed.

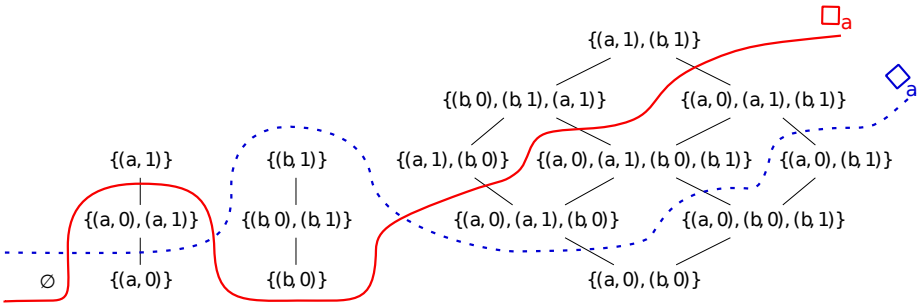


Figure 3.6: Set $F2 = \mathcal{P}_f(\{a,b\} \times 2)$ with order $\leq^F$ (for labelled transition systems). $\Box_a$ and $\Diamond_a$ are given by all values above the drawn (dashed) lines.

Finally, we close this subsection by defining syntax and semantics of coalgebraic modal languages:

Definition 3.2.18: Coalgebraic Modal Language (syntax) [Pat03]

Given a cardinal κ and a set Λ of evaluation maps $ev\colon F2 \to 2$, we define a coalgebraic modal language $\mathcal{L}^\kappa(\Lambda)$ via the grammar:

$$\varphi ::= \bigwedge \Phi \mid \neg\varphi \mid [ev] \quad \text{where } \Phi \subseteq \mathcal{L}^\kappa(\Lambda) \text{ with } card(\Phi) < \kappa \text{ and } ev \in \Lambda.$$

Given a coalgebra $\alpha\colon X \to FX$ and a formula φ, the *semantics* of such a formula is given by a map $[\![\varphi]\!]_\alpha\colon X \to 2$ defined along this lines:

- $[\![\bigwedge \Phi]\!]_\alpha = \bigcap_{\varphi\in\Phi}[\![\varphi]\!]_\alpha$
- $[\![\neg\varphi]\!]_\alpha = X \setminus [\![\varphi]\!]_\alpha$
- $[\![[ev]\varphi]\!]_\alpha = ev \circ F[\![\varphi]\!]_\alpha \circ \alpha$

The last case describes the prefixing of a formula φ with a modality $[ev]$. For simplicity we will often write $[\![\varphi]\!]$ instead of $[\![\varphi]\!]_\alpha$. Furthermore for $x \in X$ we write $x \models \varphi$ whenever $[\![\varphi]\!]_\alpha(x) = 1$. In addition, we will use derived operators such as tt (empty conjunction), ff ($\neg tt$) and $\bigvee$ (disjunction).

In [Pat03] Pattinson has already isolated the property of a separating set of predicate liftings to ensure that logical and behavioural equivalence coincide, i.e., the Hennessy-Milner property holds with some further conditions on Λ. Besides, the result has already been improved in [Sch08], where the proof works for polyadic predicate liftings without additional assumptions on Λ and is closer to our setting.

Proposition 3.2.19: [Sch08; KM18]

The logic $\mathcal{L}^\kappa(\Lambda)$ is sound, that is given a coalgebra $\alpha\colon X \to FX$ and $x, y \in X$, $x \sim y$ implies that $[\![\varphi]\!]_\alpha(x) = [\![\varphi]\!]_\alpha(y)$ for all formulas φ.

Whenever F is κ-accessible and Λ is separating for F, the logic is also expressive: whenever $[\![\varphi]\!]_\alpha(x) = [\![\varphi]\!]_\alpha(y)$ for all formulas φ we have that $x \sim y$.

For completeness, we give a proof based on evaluation maps in the Appendix A.3.

Now, we have discussed two of the three coalgebraic criteria of the triad: behavioural equivalence and coalgebraic logic. Therefore, we can move to the third part and present the coalgebraic game.

3.3 Coalgebraic Games for the Classical Case

This section presents an original contribution presented in this thesis: a coalgebraic game is introduced, which fills the gap of the triad mentioned in the motivation of this chapter.

First, the game itself is presented followed by a discussion about a variation of the game characterization. And finally, we conclude with the connection between the spoiler strategy and a distinguishing formula for two non-bisimilar states.

3.3.1 Coalgebraic Games

We will now present the rules for the behavioural equivalence game. At the beginning of a game, there are two states x, y available for selection. The aim of the spoiler (**S**)

is to prove that $x \not\sim y$, the duplicator (**D**) attempts to show the opposite.

Definition 3.3.1: Coalgebraic Game

- **Initial situation:** A coalgebra $\alpha\colon X \to FX$ and two states $x, y \in X$.
- **Step 1: S** chooses $s \in \{x, y\}$ and a predicate $p_1\colon X \to 2$.
- **Step 2: D** takes $t \in \{x, y\}\backslash\{s\}$ if $x \neq y$ and $t = s$ otherwise and has to answer with a predicate $p_2\colon X \to 2$ satisfying

$$Fp_1(\alpha(s)) \leq^F Fp_2(\alpha(t))$$

- **Step 3: S** chooses p_i with $i \in \{1, 2\}$ and some state $x' \in X$ with $p_i(x') = 1$.
- **Step 4: D** chooses some state $y' \in X$ with $p_j(y') = 1$ where $i \neq j$.

After one round the game continues in Step 1 with states x' and y'. In case **S** chooses $i = 1$ the game proceeds with (x', y') otherwise with (y', x'). **D** wins the game if the game continues forever or if **S** has no move at Step 3. In the other cases, i.e. **D** has no move at Step 2 or Step 4, **S** wins.

In a sense such a game seems to mimic the evaluation of a modal formula, but note that the chosen predicates do not have to be bisimulation-invariant, as opposed to modal formulas.

Theorem 3.3.2

Assume that F preserves weak pullbacks and has a separating set of monotone evaluation maps. Then $x \sim y$ iff **D** has a winning strategy for the initial situation (x, y).

The proof of Theorem 3.3.2 splits into three parts. On the one hand, we need to show that the relation

$$\mathcal{W} = \{(x, y) \in X \times X \mid \text{there exists a winning strategy of } \mathbf{D} \text{ for } (x, y)\}$$

is an equivalence. Secondly, a part of the proof is to establish a winning strategy for **D** whenever $x \sim y$. And thirdly, a witness $f : X \to X/\mathcal{W}$ (i.e. coalgebra homomorphism) with $f(x) = f(y)$ for each $(x, y) \in \mathcal{W}$ and a coalgebra $\beta : X/\mathcal{W} \to F(X/\mathcal{W})$ have to be constructed.

Lemma 3.3.3

Given a coalgebra $\alpha\colon X \to FX$ for an endofunctor $F : \mathbf{Set} \to \mathbf{Set}$, then

$$\mathcal{W} = \{(x, y) \in X \times X \mid \text{there exists a winning strategy of } \mathbf{D} \text{ for } (x, y)\}$$

is an equivalence relation.

Proof:

- $\mathcal{W}$ *is reflexive:* $(x, x) \in \mathcal{W}$ for every $x \in X$.

  Assume that **S** chooses x and p_1, then **D** chooses x and p_1 as well, for which we clearly have $Fp_1(\alpha(x)) \leq^F Fp_1(\alpha(x))$. Then the next game situation is (x', x'), for which we can continue this strategy forever.

- $\mathcal{W}$ *is symmetric:* $(x, y) \in \mathcal{W}$ implies $(y, x) \in \mathcal{W}$.

  If there is a winning strategy for (x, y) there must always also be a winning strategy for (y, x), since **S** can choose either x or y.

- $\mathcal{W}$ *is transitive:* if $(x, y), (y, z) \in \mathcal{W}$, then $(x, z) \in \mathcal{W}$.

  Assume that in Step 1 **S** chooses x and p_1 (the case where **S** chooses y is analogous, taking into account that $\mathcal{W}$ is symmetric). We know by $(x, y) \in \mathcal{W}$ that **D** has an answer, hence he chooses p_2, for which $Fp_1(\alpha(x)) \leq^F Fp_2(\alpha(y))$.

  If **S** were to make the choice of p_2 and y, we know by $(y, z) \in \mathcal{W}$ that **D** has an answering move, by choosing p_3 such that $Fp_2(\alpha(y)) \leq^F Fp_3(\alpha(z))$.

  Hence **D** makes the choice of p_3 in Step 2. Now we have that $Fp_1(\alpha(x)) \leq^F Fp_2(\alpha(y)) \leq^F Fp_3(\alpha(z))$ and, by transitivity, $Fp_1(\alpha(x)) \leq^F Fp_3(\alpha(z))$. (Transitivity holds by Lemma 3.2.7.)

  Assume that in Step 3 **S** chooses p_1, x' with $p_1(x') = 1$. Again, by $(x, y) \in \mathcal{W}$, there is an answer of **D** who chooses y' with $p_2(y') = 1$ and $(x', y') \in \mathcal{W}$. From $(y, z) \in \mathcal{W}$ we know that if **S** chooses p_2, y', there is an answer by **D** who chooses z' with $p_3(z') = 1$ and $(y', z') \in \mathcal{W}$. This state z' is hence finally chosen by **D** in Step 4.

  If instead in Step 3 **S** chooses p_3, z', the choice propagates in the other direction.

  Since we now have $(x', y'), (y', z') \in \mathcal{W}$, we can continue this strategy for **D** forever.

□

Before we formally give the second and third part of the proof, we will quickly sketch a winning strategy for **D** when $x \sim y$: In Step 1 **S** plays p_1 which represents a set of states. One good strategy for **D** in Step 2 is to close this set under behavioural equivalence, i.e., to add all states which are behaviourally equivalent to a state in p_1, thus obtaining p_2. It can be shown that $Fp_1(\alpha(s)) \leq^F Fp_2(\alpha(t))$ always holds for this choice. Now, if **S** chooses x', p_1 in Step 3, **D** simply takes x' as well. On the other hand, if **S** chooses x', p_2, then either x' is already present in p_1 or a state y' with $x' \sim y'$. **D** simply chooses y' and the game continues.

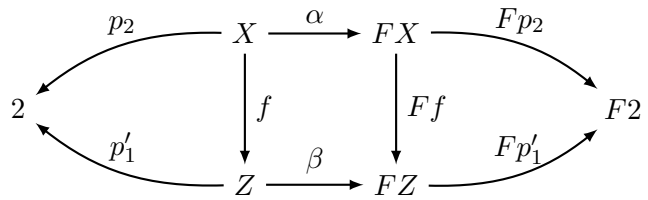
We now proceed with the soundness and completeness proof of Theorem 3.3.2.

Proof: (Theorem 3.3.2):

$x \sim y \Rightarrow$ **D** *has winning strategy:* We show that whenever $x \sim y$, then **D** can always answer the steps of **S** and we end up in a pair $x' \sim y'$, from which this strategy continues.

Whenever $x \sim y$, there exists a coalgebra $\beta\colon Z \to FZ$ and a coalgebra homomorphism $f\colon X \to Z$ such that $f(x) = f(y)$.

We assume that **S** chooses state x (the other case is analogous) and a predicate $p_1\colon X \to 2$. **D** has now to react with a predicate p_2. This is constructed by setting $p_2(x) = 1$ for $x \in X$ whenever there exists $x' \in X$ such that $f(x) = f(x')$ and $p_1(x') = 1$. In other words, we set $p_2 = p_1' \circ f$, where $p_1'\colon Z \to 2$ is the least predicate such that $p_1' \circ f \geq p_1$ (i.e., $p_1' \leq p$ for all p satisfying $p \circ f \geq p_1$).



Since $p_1 \leq p_2$, we know by Lemma 3.2.8 that $Fp_1 \leq^F Fp_2$ holds.

We obtain:

$$(Ff \circ \alpha)(x) = (\beta \circ f)(x) = (\beta \circ f)(y) = (Ff \circ \alpha)(y)$$

which implies

$$\begin{aligned}(Fp_1' \circ Ff \circ \alpha)(x) &= (Fp_1' \circ Ff \circ \alpha)(y)\\ \Rightarrow (Fp_2 \circ \alpha)(x) &= (Fp_2 \circ \alpha)(y)\\ \Rightarrow (Fp_1 \circ \alpha)(x) &\leq^F (Fp_2 \circ \alpha)(x) = (Fp_2 \circ \alpha)(y)\end{aligned}$$

S now chooses a predicate p_i and a state x' with the constraints described in Step 3, i.e., $p_i(x') = 1$. (If S can not give such a predicate and state, **D** wins automatically.)

If **S** chooses p_1 and x', **D** can simply pick $y' = x'$, since $p_2(y') \geq p_1(x') = 1$. In this case we end up in (x', x') with $x' \sim x'$.

If **D** chooses p_2 and x', by construction of p_2 **D** can find a state y' with $f(x') = f(y')$ and $p_1(y') = 1$. In this case $x' \sim y'$ holds.

In both cases the game can continue.

D *has a winning strategy* $\Rightarrow x \sim y$*:* We already by Lemma 3.3.3 know that

$$\mathcal{W} = \{(x, y) \in X \times X \mid \text{there exists a winning strategy of } \mathbf{D} \text{ for } (x, y)\}$$

is an equivalence. We define a function $f\colon X \to Y$ with $Y = X/\mathcal{W}$ and $f(x) = [x]_{\mathcal{W}}$.

$$\begin{array}{ccc} X & \xrightarrow{\alpha} & FX \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle Ff} \\ Y & \xrightarrow{\beta} & FY \end{array}$$

It suffices to show that $\beta(f(x)) := Ff(\alpha(x))$ is well defined, since then we have a coalgebra homomorphism that witnesses the behavioural equivalence of x, y. (Note that $f(x) = f(y)$, since both are contained in the same equivalence class.)

Assume that we have a winning strategy for (x, y) (i.e., $(x, y) \in \mathcal{W}$) or in other words $f(x) = f(y)$, but $Ff(\alpha(x)) \neq Ff(\alpha(y))$. Then we know, by the assumption that the functor F has a separating set of predicate liftings (respectively the equivalent condition in [Sch08]), that some $p\colon Y \to 2$ exists such that $Fp(Ff(\alpha(x))) \neq Fp(Ff(\alpha(y)))$.

We now show, by contradiction, that **D** does not have a winning strategy for (x, y): **S** chooses $p_1 = p \circ f$ and we obtain $Fp_1 \circ \alpha(x) \neq Fp_1 \circ \alpha(y)$, since $Fp_1 = Fp \circ Ff$. Since the preorder $\leq^F$ on $F2$ is antisymmetric due to Proposition 3.2.17 at least one of the following two overlapping cases will occur:

- $Fp_1 \circ \alpha(x) \not\leq^F Fp_1 \circ \alpha(y)$
- $Fp_1 \circ \alpha(y) \not\leq^F Fp_1 \circ \alpha(x)$

Here we only consider the first case, since for the second case the argument is analogous. **S** picks x and **D** can not play p_2 such that $p_2 \leq p_1$, since in this case we would get $Fp_2 \leq^F Fp_1$. Combining this with the condition of Step 2 we obtain $(Fp_1 \circ \alpha)(x) \leq^F (Fp_2 \circ \alpha)(y) \leq^F (Fp_1 \circ \alpha)(y)$ which, with transitivity, is a contradiction to the first case above.

Hence $p_2 \not\leq p_1$, which implies that some $x' \in X$ exists such that $p_2(x') = 1$ and $p_1(x') = 0$. So **S** picks p_2 and x'. **D** then picks some $y' \in X$ with $p_1(y') = 1$. If $(x', y') \in \mathcal{W}$ it follows from the construction of $p_1 = p \circ f$ that $p_1(x') = p_1(y')$, but this is again a contradiction to $p_1(x') = 0$. Hence $(x', y') \notin \mathcal{W}$ and **D** does not have a winning strategy. □

We now give an example that illustrates the differences between our generic game and the classical bisimulation game for labelled transition systems [Sti99].

Example 3.3.4

Consider the transition system in Figure 3.7, which depicts a coalgebra $\alpha\colon X \to FX$, where $F = \mathcal{P}_f(A \times \_)$ specifies finitely branching labelled transition systems. Clearly $x \nsim y$.

First consider the classical game where one possible winning strategy of the spoiler is as follows: he moves $x = 1 \xrightarrow{a} 4$, which must be answered by the duplicator via $y = 2 \xrightarrow{a} 5$. Now the spoiler switches and makes a move $5 \xrightarrow{a} 8$, which can not be answered by the duplicator.

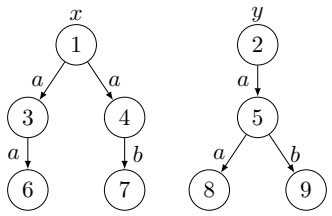


Figure 3.7: Spoiler has a winning strategy at (x, y).

In our case a corresponding game proceeds as follows: the spoiler chooses x and $p_1 = \chi_{\{4\}}$. Now the duplicator takes y and can for instance answer with $p_2 = \chi_{\{5\}}$, which leads to

$$Fp_1(\alpha(x)) = \{(a, 0), (a, 1)\} \leq^F \{(a, 1)\} = Fp_2(\alpha(y))$$

(Compare this with the visualization of the order $\leq^F$ on $F2$ in Figure 3.6.) Regardless of how **S** and **D** choose states, the next game configuration is $(4, 5)$. Now the spoiler is *not* forced to switch, but can choose 4 and can play basically any predicate p_1, which leads to either $Fp_1(\alpha(4)) = \{(b, 1)\}$ or $Fp_1(\alpha(4)) = \{(b, 0)\}$. **D** has no answering move, since $Fp_2(\alpha(5))$ will always contain tuples with *a and b*, which are not in $\leq^F$-relation with the move of **S** (see also Figure 3.6).

This game is inspired by the game for labelled Markov processes in [DLT08], where both players has to choose subsets of the state space. Therefore, we explain the connection in more detail.

We start with a discussion about the motivation to use predicates within the game and conclude with the comparison of both games.

At the level of coalgebras execution results in an element $\alpha(x) \in FX$. Thinking of probabilistic systems as described in Example 3.3.5, a single transition loses significance compared to one action in a labelled transition system.

Example 3.3.5

In Figure 3.8, both states x and y are initial states of systems with a probabilistic branching defined by the functor $F = (\mathcal{D}\_ + \{\bullet\})^{\{a,b\}}$. In addition, both states have the probability 1 to end in a state, which enables a *b-transition* with probability 1, after executing an *a-transition*. Now assume, that one starts from state 6 taking the *a-transition* with probability 1. Next, we require that this move has to be imitated with a single *a-transition* from state 1.

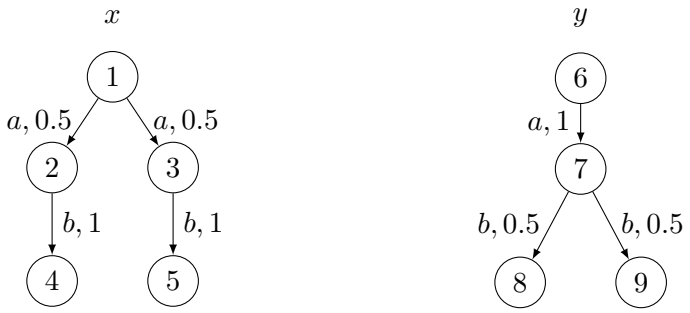


Figure 3.8: Two transition systems with the branching type characterized by the functor $F = (\mathcal{D}\_ + \{\bullet\})^{\{a,b\}}$. The left state x has the same behaviour as the state y to the right.

Obviously, this is not possible from state x and therefore we need to extend single transitions to moves based on predicates (compare this with the moves in the games over LTS given in Definition 3.2.1).

Therefore, in a generic game, a player chooses a predicate instead of a single transition and we get $Fp(\alpha(x)) \in F2$. This is why the representation of predicate liftings as evaluation maps $ev : F2 \to 2$ better fits into our setting.

Since this idea is inspired by a probabilistic game given in [DLT08], we want to compare our generic approach with the so-called ε-bisimulation game.

Comparison to [DLT08]: Let $h\colon A \times X \times \mathcal{P}X \to [0,1]$ be a labelled Markov process, which means that for $a \in A$, $x \in X$, $S \mapsto h(a,x,S)$ is a sub-probability measure. We simplify for explanatory purposes and assume that A is a singleton and that $h(a,x,X)$ is either 1 or 0 (either the probabilities sum up to 1 or to 0). Hence,

such a system corresponds to a coalgebra $\alpha\colon X \to \mathcal{D}X + 1$ (i.e. $1 = \{\bullet\}$).

In particular, [DLT08] introduces spoiler-duplicator games for ε-bisimulation. In the ε-bisimulation game the spoiler wants to disprove that the distance of two states is smaller or equal than some threshold $\varepsilon \geq 0$. Analogous to our game, the duplicator wants to demonstrate the opposite.

To compare the games, we assume that $\varepsilon = 0$, which means that we have exactly probabilistic bisimilarity as defined in [LS89] and in this case the rules of the game are as follows: consider two states $x \neq y$ (we omit the trivial case $x = y$).

- Step 1: **S** chooses a state $s \in \{x, y\}$. **S** will play on s whereas **D** will play on $t \in \{x, y\} \setminus \{s\}$. **S** chooses a label $a \in A$ and a set $E \subseteq X$.
- Step 2: **D** chooses a set $F \subseteq X$, such that $h(a, t, F) \geq h(a, s, E)$.
- Step 3: **S** chooses a state x' in E or F.
- Step 4: **D** chooses a state y' in the set not chosen by **S**. In this way we obtain one state in E and one state in F.

For the functor $F = \mathcal{D}\_ + 1$ both games are the same: for a non-terminating state s, $Fp_1(\alpha(s))$ corresponds to $h(a, s, E)$ whenever E is the set specified by p_1. Furthermore $\leq^F$ is in this case just the order on the reals (see Example 3.2.6). If however, s is terminating, i.e. $\alpha(s) = \bullet$, we have $Fp_1(\alpha(s)) = \bullet$ and $h(a, s, E) = 0$ for every E. Hence both games agree if both states are terminating or non-terminating.

If however x is terminating and y is not, it is necessary for **S** in the game of [DLT08] to choose $s = y$ and, for instance $E = X$, since **D** can not match this move. In our game **S** can choose either x or y, since $\bullet$ is not related to any real number via $\leq^F$ and the inequality can never be satisfied.

This can also be extended to functors of type $(\mathcal{D}\_+1)^L$, where L represents a finite set of labels or to sub-probability distributions. For sub-probability distributions, the same phenomenon as above appears: whenever there are two states whose sub-probability distributions do not sum up to the same value, **S** will always win in Step 2, regardless of his choice, where in the game of [DLT08], **S** has to pick the state with the larger value.

The branching type $(\mathcal{D}\_ + 1)^L$ permits only weights greater than or equal to zero, where the functor $F = (\mathbb{R}_0 \times (\mathbb{R}_0^-)^A_\omega$ in Example 3.2.16 allows also negative weights. As a result, the game fails to be complete, illustrated by the following counterexample. Although x, y are not behaviourally equivalent, the duplicator can mimic each move of the spoiler.

Example 3.3.6: D wins for $x \nsim y$ in case $\leq^F$ is not anti-symmetric

Again we consider a pair of non-bisimilar states x, y in a system of the branching type $F = \mathbb{R}_0 \times (\mathbb{R}_{\overline{0}})^A_\omega$ and explain the connection between the game and the lifted order $\leq^F$, which is not anti-symmetric. (Note, the termination behaviour is the same for every label and therefore omitted in the visual presentation.)

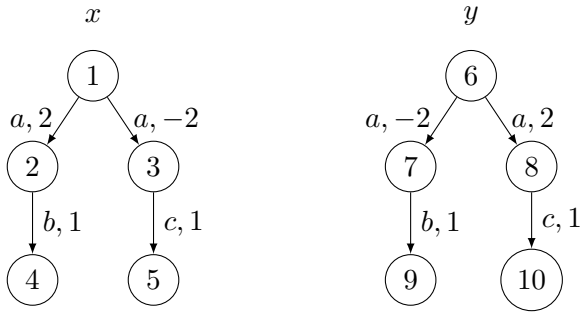


Figure 3.9: The inequivalent behaviour of x and y can not be distingusihed by the lifted order $\leq^F$ and therefore the duplicator can win although $x \nsim y$.

Already in Example 3.2.16, we pointed out that the states x, y are not behaviourally equivalent.

Now, assume **S** chooses x and a predicate p_1 which only includes state 2, **D** can answer with $\hat{p}_2 = \{7\}$ and obtains a valid move:

Considering the two values $t_1 = Fp_1(\alpha(x)), t_2 = Fp_2(\alpha(y))$, for simplicity we consider only the a-label since x, y behave the same for b and c, we get $\pi_2(t_1)(a) = [0 \to -2, 1 \to 2]$ and $\pi_2(t_2)(a) = [0 \to 2, 1 \to -2]$ and we have $t_1 \neq t_2$, but $t_1 \leq^F t_2$ by $[(0,0) \to -2, (0,1) \to 4, (1,1) \to -2]$. Thus, the condition in Step 2 of the game is satisfied. No matter what spoiler chooses in Step 3, the next round proceeds with $(2, 7)$.

Note, that for $(2, 7)$ only the states $4, 9$ play a major role and in case spoiler chooses $2, \{4\}$ $(7, \{9\})$ the duplicator has a valid move $7, \{9\}$ $(2, \{4\})$ in Step 2. Regardless of the moves in Steps 3 and 4 the last round is $(4, 9)$. Since the spoiler has no move at Step 1, the game terminates and the duplicator wins.

Note, in case **S** chooses $(x, \{3\})$,$(y, \{7\})$, or $(y, \{8\})$ for the initial pair (x, y) the situation is analogous. Furthermore, a move such $x, \{2, 3\}$ by **S** makes it even easier for **D** since the a-value evaluates to $[0 \mapsto 0, 1 \mapsto -2 + 2]$. (Note, that other moves for the spoiler do not play a significant role, because $4, 5, 9, 10$ are no successor states of state 1 or 6.)

One problem with the functor in the example above is that the lifted order is not anti-symmetric. The second problem is, that the functor is not weak pullback preserving and behaviour equivalence and F-bisimulation do not coincide (for further details see Section 2.3.3).

In this context, it has to be mentioned that a small modification of the game rules omits the preorder lifting and then weak pullback preservation is no longer necessary. The evaluation maps (i.e. modalities) can be integrated directly into Step 2, the game rule which forces the duplicator at least to mimic the move of the spoiler.

Moreover, this variation provides a small foretaste of the content presented in Chapter 6, where the level of abstraction goes beyond **Set**. However, it does not solve the problems presented in Example 3.3.6, because we still require Λ to be a separating set of monotone predicate liftings.

Game variant: By looking at the proof of Theorem 3.3.2 it can be easily seen that the game works as well if we replace the condition $Fp_1(\alpha(s)) \leq^F Fp_2(\alpha(t))$ in Step 2 by $ev(Fp_1(\alpha(s))) \leq ev(Fp_2(\alpha(t)))$ for all $ev \in \Lambda$, provided that Λ is a separating set of monotone evaluation maps. This variant is in some ways less desirable, since we have to find such a set Λ (instead of simply requiring that it exists), on the other hand in this case the proof does not require weak pullback preservation, since we do not any more require transitivity of $\leq^F$. This variant of the game is conceptually quite close to the Λ-bisimulations studied in [GS13]. In our notation, a relation $S \subseteq X \times X$ is a Λ-bisimulation, if whenever $x\, S\, y$, then for all $ev \in \Lambda$, $p\colon X \to 2$, $ev(Fp(\alpha(x))) \leq ev(Fq(\alpha(y)))$, where q corresponds to the image of p under S (and the same condition holds for S^{-1}). Λ-bisimulation is sound and complete for behavioural equivalence if F admits a separating set of monotone predicate liftings, which coincides with our condition.

We have already explained how one can extract the winning strategy of the duplicator from the bisimulation relation. In the next section, we show, that in the case of two non-bisimilar states $x \nsim y$ we can convert a modal logic formula $\varphi_{x,y}$ distinguishing x, y (i.e., $x \models \varphi$ and $y \not\models \varphi$) into a winning strategy for the spoiler.

3.3.2 The Coalgebraic Triad: Bisimulation, Modal Logics & Games

In bisimulation games the winning strategy for **D** can be derived from the bisimulation or, in our case, from the map f that witnesses the behavioural equivalence of two states x, y (see the description after Lemma 3.3.3).

Here we will show that the winning strategy for **S** can be derived from a modal

formula φ which distinguishes x, y, i.e., $x \models \varphi$ and $y \not\models \varphi$. We assume that all modalities are monotone (cf. Proposition 3.2.17).

The spoiler strategy is defined over the structure of φ:

- $\varphi = \bigwedge \Phi$: in this case **S** picks a formula $\psi \in \Phi$ with $y \not\models \psi$.
- $\varphi = \neg\psi$: in this case **S** takes ψ and reverses the roles of x, y.
- $\varphi = [ev]\psi$: in this case **S** chooses x and $p_1 = [\![\psi]\!]$ in Step 1. After **D** has chosen y and some predicate p_2 in Step 2, we can observe that $p_2 \not\leq [\![\psi]\!]$ (see proof of Theorem 3.3.7). Now in Step 3 **S** chooses p_2 and a state y' with $p_2(y') = 1$ and $y' \not\models \psi$. Then **D** must choose $[\![\psi]\!]$ and a state x' with $x' \models \psi$ in Step 4 and the game continues with x', y' and ψ.

Theorem 3.3.7

Assume that $\alpha\colon X \to FX$ is a coalgebra and F satisfies the requirements of Theorem 3.3.2. Let φ be a formula that contains only monotone modalities and let $x \models \varphi$ and $y \not\models \varphi$. Then the spoiler strategy described above is winning for **S**.

Proof: Each step described in the strategy yields a smaller formula by structural induction and a pair of states which is distinguished by the formula. Hence the game will eventually terminate.

We only have to consider the case $\varphi = [ev]\psi$ in more detail and show (by contradiction) that **S** can make a valid move in Step 3 by proving that the predicate p_2 chosen by **D** in Step 2 must satisfy $p_2 \not\leq [\![\psi]\!]$.

Hence, assume that $p_2 \leq [\![\psi]\!]$. From Lemma 3.2.8 and from the monotonicity of ev it follows that

$$(ev \circ Fp_2 \circ \alpha)(y) \leq (ev \circ F[\![\psi]\!] \circ \alpha)(y) = [\![\varphi]\!](y).$$

Since $y \not\models \varphi$ the right-hand side, as well as the left-hand side of the inequality must be 0. On the other hand $Fp_1 \circ \alpha(x) \leq^F Fp_2 \circ \alpha(y)$ and hence, again due to monotonicity of ev, we have

$$(ev \circ F[\![\psi]\!] \circ \alpha)(x) \leq (ev \circ Fp_2 \circ \alpha)(y).$$

Since $x \models \varphi$ the left-hand side of the inequality must be 1. But this is a contradiction, because then the right-hand side has to be equal to 1 as well.

Hence $p_2 \not\leq [\![\psi]\!]$, which means that there is a $y' \in X$ such that $p_2(y') = 1$ and $[\![\psi]\!](y') = 0$, i.e., $y' \not\models \psi$. □

Finally, we have completed the generic framework since we have filled the gap in the triad. However, we did not yet show how to directly derive the winning strategy of both players or how to construct a distinguishing formula φ (see Figure 3.10).

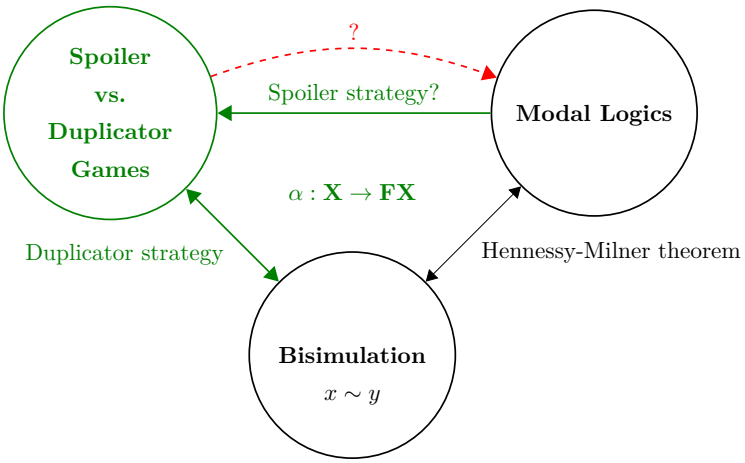


Figure 3.10: The coalgebraic triad capturing coalgebraic bisimulation, modal logics and games for weak pullback preserving endofunctors $F : \mathbf{Set} \to \mathbf{Set}$.

3.4 Explaining Non-Bisimilarity in a Coalgebraic Approach

The next contributions presented in this chapter focus on non-bisimilar state pairs and two ways to explain the non-bisimilarity of such state pairs. Regarding the game an algorithm which computes the winning strategy of the spoiler is introduced. In addition, a generic technique to derive distinguishing formulas is described for the logical view.

This results are inspired by the work over LTS in [Cle90], where the computation of the greatest bisimulation relation serves as a base for the recursive construction of distinguishing formulas. Therefore, this chapter includes the development of two algorithms at the level of coalgebras and their composition.

Thus, the first Subsection 3.4.1 offers an overview over generic partition refinement techniques. Afterwards, the contribution itself is discussed and a simple coalgebraic modal logic is introduced. The modalities are given by cones over $F2$ and therefore we call them *cone modalities*. Finally, this chapter concludes with the transformations between cone-based modalities and more common modalities [KMS20b].

3.4.1 An Introduction into Coalgebraic Partition Refinement Algorithms

As the name indicates, a partition refinement algorithm refines a partition. More precisely, in our context we deal with partitions over state spaces, which consists of blocks or classes and those blocks are divided into finer and disjoint entities. Before we address concrete instantiations of such algorithms and study the details behind the different runtime behaviours, we would like to start with the basic idea of such algorithms.

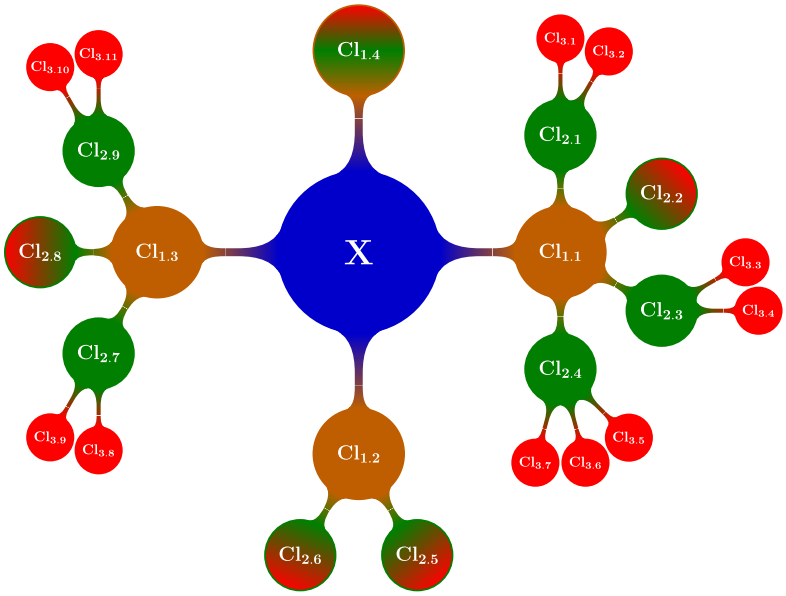


Figure 3.11: Concept of partition refinement, where the coarsest partition is the blue node including the class X, i.e. state space X. Towards the outside the classes get finer and each new partition is illustrated by a different colour. In case a node is filled with several colours, this indicates that the class remains the same for all these partitions.(Illustration based on [Kot15a; Kot15b])

For a given coalgebra $\alpha : X \to FX$ we start with a very coarse partition $\Pi_0 = \{X\}$ including just one class illustrated by the blue node in Figure 3.11. The next partition Π_1 (orange) is obtained from the initial partition Π_0 through a split of class X into finer and disjoint classes. This principle repeats for the finer classes until a partition is recognized as stable. One such criterion might be $\Pi_i = \Pi_{i+1}$, but this depends on the details of the underlying splitting techniques.

Indeed, Section 2.2.2 already introduces the foundations behind all the different partition refinement techniques [KS90; KK18; DM+17]. De facto, all the algorithms

compute the greatest fixpoint ($\nu\mathcal{F}$) of a monotone function

$$\mathcal{F}(\mathcal{R}) : \mathcal{P}(X \times X) \to \mathcal{P}(X \times X)$$

The definitions and explanations of this introduction are based on [AIS12; KK18; DM+17; Küp17] and partly originate from [KS90; PT87; Wor05].

To make life easier, we start with labelled transition systems. Given an LTS $(X, \Sigma, \to)$ we refer to Section 2.2.2, where the monotone function $\mathcal{F}$ is defined.

A first idea to evaluate if a pair of states satisfies the conditions described by $\mathcal{F}$ (see Equation 2.1), would be to compare the transition behaviour for every single class. This approach is described in an algorithm by Kanellakis and Smolka [KS90] and they already show that it suffices to work with the single classes given by the previous partition to compute $\nu\mathcal{F}$ and thus obtain the greatest bisimulation (i.e. bisimilarity) starting with $X \times X$.

Concerning the algorithm by Kanellakis and Smolka [KS90] for LTS, at the beginning the procedure separates two states if one of them has at least one a-labelled transition and the other state does not. Thus, at the beginning the states are classified by their outgoing transitions. In case, all states coincide in the labels – either each state has an a-transition for each $a \in \Sigma$ or no state has a transition – the algorithm terminates and all states are bisimilar. Otherwise, the partition Π_0 is refined and Π_1 contains k classes with $k > 1$ (and $k \leq |X|$). Here, the state space X acts as a so-called *splitter*.

A class C_i is a *splitter* for a class C_j with $i, j \in \{0, \ldots, k\}$ in case there exists some label $a \in \Sigma$, such that some states in C_j have a-transitions to C_i and others do not. Thus, a class C_j is refined in the following way [AIS12]:

$$C_{j,1} = \{s \mid s \in C_j \text{ and } s \xrightarrow{a} s', \text{ for some } s' \in C_i\}$$
$$C_{j,2} = C_j \setminus C_{j,1}$$

In general a partition Π_i gets refined to Π_{i+1} if at least one class C of Π_i is splitted into two classes by some class in $\{[x]_{\Pi_i} \mid x \in X\}$. In each iteration the algorithm checks for each class and all labels whether it can be refined based on the current partition (*refinement check*) and repeats this until no further refinements occur. The algorithm has a polynomial runtime $n \cdot m$ where n denotes the number of states and m the number of transitions [AIS12].

In addition, an optimized algorithm with runtime $m \cdot \log n$ works with the so-called *three-way-splitting* published by Paige and Tarjan [PT87]. In this algorithm, the authors work with two partitions Γ and Π, where one is coarser than the other. This

means that at least one class in Γ is the union of two or more classes in Π and the first step is to choose such a class $T \in \Gamma$. The phenomenon of the *three-way-splitting* is obtained through a refinement of the coarser partition Γ and the finer partition Π. This splits are based on $S \subset T$ with $S \in \Pi$ (i.e. $\Gamma_{i+1} = \Gamma_i \setminus T \cup \{S, T \setminus S\}$) followed by two refinement checks of Π based on S, $T \setminus S$, where T is a *compound* class (i.e. $S \subset T$) in the coarser partition which includes at least two classes of Π. The most interesting aspect of this technique is how the logarithmic part of the time performance is obtained. A class $T \in \Gamma \setminus \Pi$ is refined based on a class S in Π such that $|S| \leq \frac{|T|}{2}$ [AIS12]. However, as we are interested in the coalgebraic view of all these things, we refer the interested reader to [PT87; AIS12] for further details.

We need to emphasize that there are two fundamental approaches capturing the concepts described above. To compare the behaviour of two states, the first approach allows you to work with single classes, while the second approach deals with single classes in combination with a union of classes respecting compound classes.

To understand how the coalgebraic view is related with this two approaches we consider two different coalgebraic partition refinement techniques.

The algorithms in [KK14; KK18; Küp17] are based on the final chain construction (see Definition 2.3.22) to describe behavioural equivalence for coalgebras on concrete categories. An improved version of a final-chain based partition refinement algorithm for a wide range of endofunctors over **Set** is presented in [DM+17]. In the following we would like to give you a short overview of the technical key points presented in [KK18; DM+17].

First, we present the abstract version of a generic partition refinement, which splits with respect to all classes at once (compared with the single-class concept of Kanellakis and Smolka discussed previously). Afterwards we briefly explain the termination and runtime behaviour using the example of weighted automata.

Recall the final chain construction from Section 2.3.3 given in Definition 2.3.22. Figure 3.12 shows how the final chain enables to obtain partitions Π_i based on the i- step-behaviour in the following sense: the α^i-morphisms induce classes which only relate states with the same i-step-behaviour. The algorithm stops if an α_j is reached such that some morphism $\beta : F^j 1 \to F^{j+1} 1$ with $\beta \circ \alpha^j = F\alpha^j \circ \alpha$ is reached [KK18]. An optimization of the version described in Figure 3.12 starts with $\alpha^0 : X \to 1$ and checks for each iteration i if an arrow e^i of α^i with $\alpha^i = \alpha^i_e = m^i \circ e^i$ for some morphism m_i exists. This means that the arrow e^i is a more compact representative of α^i, i.e. it induces the same partition as α^i. Therefore, the greatest fixpoint of the sequence $\alpha^0 \dots \alpha^i \dots$ in Figure 3.12 is determined based on a surjection $e^i : X \to Y_i$

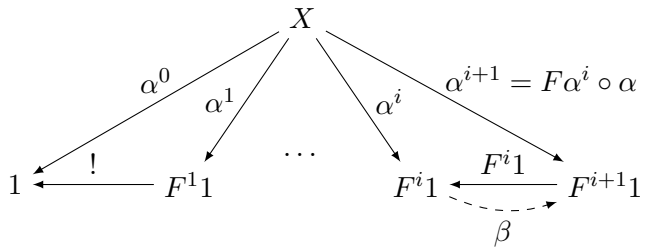


Figure 3.12: Final chain equipped with a sequence $\alpha^i : X \to F^i 1$ of morphisms. The approximative algorithm terminates if a morphism β with $\beta \circ \alpha^i = F\alpha^i \circ \alpha$ is found [KK18].

derived from $\alpha_e^i = Fe^{i-1} \circ \alpha$. Since such a compact representative e^i can be used to obtain the next morphism $\alpha_e^{i+1} = Fe^i \circ \alpha$. The algorithm terminates if $e^i \leq \alpha_e^{i+1}$, or in other words, there exists a coalgebra $\beta : F^i 1 \to F(F^i 1)$ and therefore e^i captures enough information to provide statements about the behavioural equivalence of each state pair (cf. [KK18]).

Given that, the current equivalence relation $R_i \subseteq X \times X$ is represented by the surjection e^i. Therefore, we separate x, y whenever $Fe^i(\alpha(x)) \neq Fe^i(\alpha(y))$, which intuitively means that we split with respect to all equivalence classes at once.

Now we want to talk about the termination and runtime behaviour of this algorithm, which become clearer if we use an example. First of all, working with representatives e^i instead of α^i minimizes the number of arithmetic operations.

Concerning weighted automata in general language equivalence is not decidable, therefore the runtime analysis in [KK18; KKM17] restricts to decidable cases, which strongly depends on the underlying mathematical structure (more precisely the semiring providing the weights for the transitions). In addition, the computations behind e^i are based on the solutions of linear systems of equations. So we conclude that in the case of weighted automata the runtime depends on the semiring [Küp17; KKM17].

The work presented in [DM+17; WD+20] lifts the idea by Paige and Tarjan to the coalgebraic setting of categories which have coequalizers and some additional requirements.

Just to give a rough idea, they combine the three-way-splitting based on two partitions P, Q, where Q is coarser than P, with the constructions over the final chain. The difference to a pure final chain algorithm lies in the filtering of the information used for the next splitting given by the previously computed partitions. More precisely, the information $q_{i+1} : X \to K_{i+1}$ is given by a composition of two

morphisms, where one is given by $X \twoheadrightarrow X/P_i$ (where $\twoheadrightarrow$ denotes a coequalizer). The second morphism $k_{i+1} : X/P_i \twoheadrightarrow K_{i+1}$ is determined by a *selection procedure* based on the input P_i and Q_i which respects *compound* classes (i.e. classes within the coarser partition consisting at least of two classes of the finer partition).

The *selection procedure* depends on the concrete instantiation, where one example is given by the *smaller half* strategy used in the algorithm by Paige and Tarjan [PT87].

Moreover, while there are no problems with the generalized version of the step, which refines the coarser partition Q_i into Q_{i+1}, the computation of P_{i+1} is based on the kernel of the F-lifted pair of morphisms $\langle \bar{q}_i, q_{i+1} \rangle$, where for each iteration i the morphism $\bar{q}_i$ represents a collection of the *selection information* obtained so far. Formally this means that $ker\, \bar{q}_i = Q_i$ (for kernel see Example 2.3.20).

Under a certain requirement denoted with 2-*zippability*, the computation of such F-lifted morphisms (i.e. $ker\, F\langle \bar{q}_i, q_{i+1} \rangle$) can be optimized, which enables an incremental implementation of P_{i+1} [DM+17; WD+20]. (Note, 2-zippability is a special case of m-zippability.)

Definition 3.4.1: m-zippability

A functor F is *m-zippable* if the map

$$F(A_1 + \cdots + A_m) \xrightarrow{\langle F(f_1),\ldots,F(f_m)\rangle} F(A_1 + 1) \times \cdots \times F(A_m + 1)$$

is injective for all sets $A_1, \ldots, A_m$, where $f_i = id_{A_i} + !\colon A_1 + \cdots + A_m \to A_i + 1$, with $!\colon A_1 + \cdots + A_{i-1} + A_{i+1} + \cdots + A_m \to 1$, is the function mapping all elements of A_i to themselves and all other elements to $\bullet$ (assuming that $1 = \{\bullet\}$).

The technical details of the algorithm and the *selection procedure* are quite extensive [DM+17], so we just summarize here the key points:

1. The *selection procedure* respects compound blocks and the functor F is 2-*zippable*: Necessary to enable an incremental computation of P_{i+1}, which means that this partition is derived from P_i and some information q_{i+1}, which is determined based on $X \twoheadrightarrow X/P_i$.

2. The functor F admits a suitable *refinement interface*: Necessary to enable efficient computations of the kernels where the coalgebra is encoded (i.e. a representation format in the sense of a (labelled) graph depending on F) [DM+19].

Among some typical restrictions on the functor, both conditions above enable the efficient coalgebraic partition refinement algorithm presented in [DM+17; WD+20], which runs in time $\mathcal{O}((m+n)\log n)$ (where m denotes the number of edges based on

the representation). Furthermore, a *standard final chain* algorithm is transformed into a *compound* class respecting one, which does not use all information from the previous partition as the algorithms presented in [KK18].

Partition Refinement is not sufficient to derive a Winning Strategy for the Spoiler Both algorithms described above compute behavioural equivalence relations and can therefore be used to determine the *splitting information*. But if we want to derive a winning strategy for the spoiler in our game or a distinguishing formula, it is not enough to recognize the difference (i.e. non-equivalence) of the behaviour (cf. [Section 5][WMS21]).

The next example serves as motivation to focus on the lifted $\leq^F$ during partition refinement.

Example 3.4.2

The states $(1, 2)$ in the LTS of Figure 3.13 are not bisimilar, because for all a-transitions 2 enables an a-transition and a b-transition. This is however not possible with state 1.
Therefore, the spoiler has a winning strategy for $(1, 2)$ given by $(1, \{3\})$, since $Fp(\alpha(1)) = \{(a, 0), (a, 1)\} \not\leq^F \{(a, 0)\} = Fp(\alpha(2))$. In addition 3 is separated from $4, 5$ since the values $\{(a, 1)\}$ and $\{(a, 1), (b, 1)\}$ based on $\{6, 7, 8, 9\}$ are not comparable via the lifted order $\leq^F$.

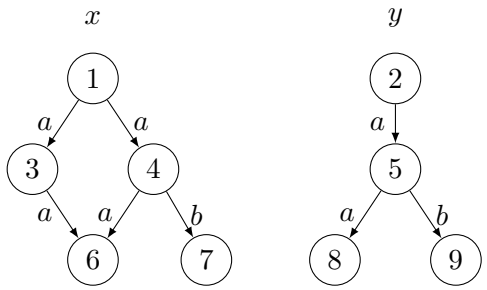


Figure 3.13: Two non-bisimilar states x and y in an LTS.

Next, we will see, that partition refinement alone, does not provide enough information to derive $(1, \{3\})$ or $(2, \{4, 5\})$ as suitable moves for the spoiler.
Assume that we are in a situation that $X/\Pi_i = \{\{6, 7, 8, 9\}, \{1, 2\}, \{3\}, \{4, 5\}\}$ and $\Pi_i \subseteq X \times X$ is the current partition derived by one of the previously described partition refinement techniques. The next step would be to split $(1, 2) \in \Pi_i$.
Now, an algorithm as the one by Kanellakis and Smolka or Paige and Tarjan

determines $S = \{3\}$ as splitting information. For the latter, such an information can be derived via the kernel computations as introduced previously.
However, the class $\{1, 2\}$ is separated due to the information given by S since state 1 has an a-successor in $\{3\}$ but state 2 not. Therefore, expressed from a coalgebraic point of view, we know that $\alpha(1)$ and $\alpha(2)$ behave differently with respect to the characteristic function χ_S (or via a *three-valued* approach presented in [DM+17]):

$$F\chi_S(\alpha(1)) = \{(a, 0), (a, 1)\} \neq \{(a, 0)\} = F\chi_S(\alpha(2))$$

Finally, both algorithms proceed and result in the partition Π which leads to $X/\Pi = \{\{1\}, \{2\}, \{3\}, \{6, 7, 8, 9\}, \{4, 5\}\}$. But to derive the winning strategy in our game, one needs to be able to compare the elements derived from the splitting information with respect to $\leq^F$ instead of $=$. More concretely, for LTS this means that we need to recognize

$$F\chi_S(\alpha(1)) = \{(a, 0), (a, 1)\} \not\leq^F \{(a, 0)\} = F\chi_S(\alpha(2))$$

If we want to generate a winning strategy for the spoiler working with the pure final chain algorithm [KK18], we have to find out which class p_c (or union of classes) causes the difference, and therefore one would have to derive all classes (or union of classes) and evaluate if $Fp_c(\alpha(x)) \leq^F Fp_c(\alpha(y))$ or $Fp_c(\alpha(y)) \leq^F Fp_c(\alpha(x))$ hold for each predicate p_c and each separated pair of states.

The approach by [DM+17] enables to save the splitting class or union of classes p_c on-the-fly, but it still remains open if $Fp_c(\alpha(x)) \leq^F Fp_c(\alpha(y))$ or $Fp_c(\alpha(y)) \leq^F Fp_c(\alpha(x))$ holds and how a modality can be extracted on-the-fly.

Our game is already defined by comparing two values $t_1, t_2 \in F2$. Therefore, the next section presents a coalgebraic partition refinement algorithm directly derived from the game, which solves the previously mentioned problem. Moreover, we also study the conditions under which such an algorithm admits a polynomial runtime and compare this with the coalgebraic *three way splitting* algorithm in [DM+17].

Remember, that a spoiler strategy exists iff there is a distinguishing formula $\varphi_{x,y}$. From Theorem 3.3.7 we know, that this formula can be transformed into a spoiler strategy. After deriving a spoiler strategy, we want to show the opposite, how such a winning strategy is used to construct a distinguishing formula.

3.4.2 Computation of Spoiler Winning Strategies

Since this is more practical, we fix X to be a finite set. To refresh the framework conditions introduced in Section 3.2, we will give a short revision. We will fix a coalgebra $\alpha : X \to FX$ for a weak pullback preserving endofunctor $F\colon \mathbf{Set} \to \mathbf{Set}$. Furthermore we assume that F has a separating set of *monotone* predicate liftings, which implies that $\leq^F$, the lifted order on 2, is anti-symmetric, hence a partial order.

We first present a simple but generic partition refinement algorithm to derive the winning strategy for the spoiler (**S**) for a given coalgebra $\alpha\colon X \to FX$. The case of the duplicator (**D**) is discussed in detail concerning Theorem 3.3.2.

In particular we consider the relation $\mathcal{W}_\alpha$, which – as we will show — is the greatest fixpoint of the following monotone function $\mathcal{F}_\alpha\colon Eq(X) \to Eq(X)$ on equivalence relations:

$$\begin{aligned} \mathcal{F}_\alpha(R) &= \{(x,y) \in R \mid \forall P \in E(R)\colon F\chi_P(\alpha(x)) = F\chi_P(\alpha(y))\} \\ \mathcal{W}_\alpha &= \{(x,y) \in X \times X \mid \text{there exists a winning strategy of } \mathbf{D} \text{ for } (x,y)\} \end{aligned}$$

In the following, we will prove that the greatest fixpoint of $\mathcal{F}_\alpha$ (i.e. $\nu\mathcal{F}_\alpha$) coincides with $\mathcal{W}_\alpha$ and hence gives us bisimilarity. Note that $\mathcal{F}_\alpha$ splits classes with respect to only a single equivalence class P as in [KS90]. This is done to avoid an exponential runtime, since the condition within $\mathcal{F}_\alpha$ can alternatively be defined based on all possible unions over $E(R)$. Hence we will need to impose extra requirements on the functor, spelled out below, in order to obtain this result.

One direction of the proof deals with deriving a winning strategy for **S** for each pair $(x,y) \notin \nu\mathcal{F}_\alpha$. In order to explicitly extract such a winning strategy for **S** – which will also be important later when we construct the distinguishing formula – we will slightly adapt the fixpoint iteration $\nu\mathcal{F}_\alpha$.

Before we come to this, we formally define and explain the strategy of **S**. A strategy for the spoiler is given by a pair of functions

$$I\colon X \times X \to \mathbb{N}_0 \cup \{\infty\} \text{ and } T\colon (X \times X)\backslash\nu\mathcal{F}_\alpha \to X \times \mathcal{P}(X).$$

Here, $I(x,y)$ denotes the first index where x, y are separated in the fixpoint iteration of $\mathcal{F}_\alpha$. The second component T tells the spoiler what to play in Step 1. In particular whenever $T(x,y) = (x,P)$, **S** will play x and $p_1 = \chi_P$.
In the case $I(x,y) < \infty$ such a winning strategy for **S** can be computed during fixpoint iteration, see Algorithm 3.1. Assume that the algorithm terminates after n steps and returns R_n. It is easy to see that R_n coincides with $\nu\mathcal{F}_\alpha$: as usual for partition refinement, we start with the coarsest relation $R_0 = X \times X$. Since $\leq^F$

Algorithm 3.1 Computation of $\nu\mathcal{F}_\alpha$ and the winning strategy of the spoiler

1: **procedure** COMPUTE $\nu\mathcal{F}_\alpha$ AND T, I FOR INPUT $\alpha : X \to FX$
2: **for all** $(x, y) \in X \times X$ **do**
3: $I(x, y) \leftarrow \infty$
4: **end for**
5: $i \leftarrow 0$, $R_0 \leftarrow X \times X$
6: **repeat**
7: $i \leftarrow i + 1$, $R_i \leftarrow R_{i-1}$
8: **for all** $(x, y) \in R_{i-1}$ **do**
9: **for all** $P \in E(R_{i-1})$ **do**
10: **if** $F\chi_P(\alpha(x)) \not\leq^F F\chi_P(\alpha(y))$ **then**
11: $T(x, y) \leftarrow (x, P)$, $I(x, y) \leftarrow i$, $R_i \leftarrow R_i \setminus \{(x, y)\}$
12: **else**
13: **if** $F\chi_P(\alpha(y)) \not\leq^F F\chi_P(\alpha(x))$ **then**
14: $T(x, y) \leftarrow (y, P)$, $I(x, y) \leftarrow i$, $R_i \leftarrow R_i \setminus \{(x, y)\}$
15: **end if**
16: **end if**
17: **end for**
18: **end for**
19: **until** $R_{i-1} = R_i$
20: **return** R_i, T, I
21: **end procedure**

is, by assumption, anti-symmetric $F\chi_P(\alpha(x)) \leq^F F\chi_P(\alpha(y))$ and $F\chi_P(\alpha(x)) \leq^F F\chi_P(\alpha(y))$ are equivalent to $F\chi_P(\alpha(x)) = F\chi_P(\alpha(y))$ and the algorithm removes a pair (x, y) from the relation iff this condition does not hold.

Every relation R_i is finer than its predecessor R_{i-1} and, since $\mathcal{F}_\alpha$ preserves equivalences, each is an equivalence relation. Since we are assuming a finite set X of states, the algorithm will eventually terminate.

In addition, $T(x, y)$ and $I(x, y)$ are updated, where we distinguish whether $Fp(\alpha(x)) \not\leq^F Fp(\alpha(y))$ or the other inequality hold.

We will now show that Algorithm 3.1 indeed computes a winning strategy for the spoiler.

Proposition 3.4.3

Assume that $R_n = \nu\mathcal{F}_\alpha, T, I$ have been computed by Algorithm 3.1. Furthermore let $(x, y) \notin R_n$, which means that $I(x, y) < \infty$ and $T(x, y)$ is defined. Then the following constitutes a winning strategy for the spoiler:

- Let $T(x, y) = (s, P_1)$. Then in Step 1 **S** plays a predicate $p_1 = \chi_{P_1}$ and $s \in \{x, y\}$.
- Assume that in Step 2 **D** answers with a state t and a predicate p_2 such that $Fp_1(\alpha(s)) \leq^F Fp_2(\alpha(t))$.
- Then, in Step 3 there exists a state $y' \in X$ such that $p_2(y') = 1$ and $I(x', y') < I(x, y)$ for all $x' \in X$ with $p_1(x') = 1$. **S** will hence select p_2 and this state y'.
- Next, in Step 4 **D** selects some x' with $p_1(x') = 1$ and the game continues with $(x', y') \notin R_n$.

Proof: We have to show that whenever we reach Step 3 there always exists a state $y' \in X$ such that $p_2(y') = 1$ and $I(x', y') < I(x, y)$ for all $x' \in X$ with $p_1(x') = 1$.

Let us first observe that $p_2 \not\leq p_1$. If this were the case, we would have $Fp_1(\alpha(s)) \leq^F Fp_2(\alpha(t)) \leq^F Fp_1(\alpha(t))$. But $\{x, y\} = \{s, t\}$ are separated at Step $I(x, y) = i$ precisely because this inequality does not hold for p_1 which represents one of the equivalence classes of R_{i-1}. Hence there exists an $y' \in X$ such that $p_2(y') = 1$ and $p_1(y') = 0$.

Since the equivalence relations R_i are subsequently refined by the algorithm, p_1 – being an equivalence class of R_{i-1} – is a union of equivalence classes of R_n. So, since y' is not contained in $P_1 = \hat{p_1}$, it is not in R_{i-1}-relation to any $x' \in P_1$, hence $I(x', y') \leq i - 1$ for all such x'.

Since the index $I(x, y)$ decreases after every round of the game, **D** will eventually not be able to find a suitable answer in Step 2 and will lose. □

Finally, we show that $\nu\mathcal{F}_\alpha$ coincides with $\mathcal{W}_\alpha$ and therefore also with behavioural equivalence $\sim$ (see [KM18]). For this purpose, we need one further requirement on the functor:

Definition 3.4.4

Let F: **Set** $\to$ **Set** be an endofunctor on **Set**. We say that F is *separable by singletons for a set* X if the following holds: for all $t_0 \neq t_1$ with $t_0, t_1 \in FX$,

there exists $p\colon X \to 2$ where $p(x) = 1$ for exactly one $x \in X$ (i.e., p is a singleton) and $Fp(t_0) \neq Fp(t_1)$. Moreover, F is *separable by singletons* if F is separable by singletons for all sets X.

A wide range of functors as introduced in Chapter 2.3 are separable by singletons.

Example 3.4.5

We show separability by singletons for two functors which are already introduced in Example 2.3.13. Let X be some non empty set.

1. Labelled transition systems with finitely branching given by the functor $F = \mathcal{P}_f(A \times \_)$: Let $t_0, t_1 \in \mathcal{P}_f(A \times X)$ with $t_0 \neq t_1$ be given. Therefore, at least for one $a \in A$ there exists some $x \in X$ such that either $(a, x) \in t_0$ and $(a, x) \notin t_1$ or $(a, x) \notin t_0$ and $(a, x) \in t_1$.

   Next, we define a predicate $p : X \to 2$ with $p(x) = 1$ and $p(x') = 0$ for $x' \in X \setminus \{x\}$. For the case $(a, x) \in t_0$ and $(a, x) \notin t_1$ (the other case is analogous) we get $Fp(t_0) \neq Fp(t_1)$ since $Fp(t_0)$ includes $(a, 1)$ and $Fp(t_1)$ not.

2. Probabilistic systems with termination (and finite support) given by the functor $F = (\mathcal{D}\_ + \{\bullet\})^A$: Let $t_0, t_1 \in (\mathcal{D}X + \{\bullet\})^A$ with $t_0 \neq t_1$ be given. Therefore, at least for one $a \in A$ we have $t_0(a) \neq t_1(a)$ and we have to distinguish two cases:

   - $t_0(a) = \bullet$ and $t_1(a) \neq \bullet$: Since $t_1(a) \in \mathcal{D}X$ at least one $x \in X$ exists such that $t_1(a)(x) > 0$ and we define a predicate $p : X \to 2$ with $p(x) = 1$ and $p(x') = 0$ for $x' \in X \setminus \{x\}$. Thus, we get $Fp(t_0) \neq Fp(t_1)$ since $Fp(t_0)(a) = \bullet$ and $Fp(t_1)(a) \in \mathcal{D}2$ with $Fp(t_1)(a)(1) > 0$.

     (The case $t_0(a) \neq \bullet$ and $t_1(a) = \bullet$ can be shown analogous.)

   - $t_0(a) \neq \bullet$ and $t_1(a) \neq \bullet$: Since $t_0(a) \neq t_1(a)$ for some $a \in A$ we have at least one $x \in X$ such that $t_0(a)(x) > t_1(a)(x)$ or $t_0(a)(x) < t_1(a)(x)$. The proof works analogous for both cases, so we just consider the first one. Again, we define a predicate $p : X \to 2$ with $p(x) = 1$ and $p(x') = 0$ for $x' \in X \setminus \{x\}$. Next, we obtain $Fp(t_0)(a) \in \mathcal{D}2$ such that $Fp(t_0)(a)(1) > Fp(t_1)(a)(1)$.

Besides, it is obvious that separability by singletons implies the existence of a separable set of predicate liftings, however the reverse implication does not hold as

the following example shows.

Example 3.4.6

A functor that does not have this property, but does have a separating set of predicate liftings, is the monotone neighbourhood functor $\mathcal{M}$ with

$$\mathcal{M}X = \{Y \in \mathcal{Q}\mathcal{Q}X \mid Y \text{ upwards-closed}\} \text{ (see e.g. [DM+18])},$$

where $\mathcal{Q}$ is the contravariant powerset functor.
Consider $X = \{a, b, c, d\}$ and two elements $t_0, t_1 \in \mathcal{M}X$ where

$$t_0 = \uparrow\{\{a, b\}, \{c, d\}\}, t_1 = \uparrow\{\{a, b, c\}, \{a, b, d\}, \{c, d\}$$

That is, the only difference is that t_0 contains the two-element set $\{a, b\}$ and t_1 does not. For any singleton predicate p the image of $\mathcal{Q}p\colon \mathcal{P}2 \to \mathcal{P}X$ does not contain a two-element set, hence $\mathcal{M}p(t_0) = \mathcal{M}p(t_1)$ – since they contain the same sets of sizes different from two – and t_0, t_1 cannot be distinguished.

We are now ready to prove the following theorem.

Theorem 3.4.7

Let $\alpha : X \to FX$ be a coalgebra where F is separable by singletons. Then $\nu\mathcal{F}_\alpha = \mathcal{W}_\alpha$, i.e., $\nu\mathcal{F}_\alpha$ contains exactly the pairs $(x, y) \in X \times X$ for which the duplicator has a winning strategy.

Proof:

"$\subseteq$" Assume that $(x, y) \in \nu\mathcal{F}_\alpha = R_n$. We show that $x \sim y$ and with [KM18] it follows that $(x, y) \in \mathcal{W}_\alpha$. We do this by constructing a coalgebra homomorphism f with $f(x) = f(y)$.

Let $Y = E(R_n)$, the set of equivalence classes of R_n and we define $f\colon X \to Y$, $f(x) = [x]_{R_n}$. In order to show that f is a coalgebra homomorphism, we have to construct a coalgebra $\beta\colon Y \to FY$ such that $\beta \circ f = Ff \circ \alpha$. We define $\beta([x]_{R_n}) = Ff(\alpha(x))$ and it suffices to show that β is well-defined.

So let $(x, y) \in R_n$ and assume by contradiction that $t_0 = Ff(\alpha(x)) \neq Ff(\alpha(y)) = t_1$.

Then, since F is separable by singletons, we have a singleton predicate p with $Fp(t_0) \neq Fp(t_1)$. By expanding the definition we get $F(p \circ f)(\alpha(x)) \neq F(p \circ f)(\alpha(y))$. By construction $p \circ f = \chi_P$, where P is an equivalence class

of R_n. This is a contradiction, since $\mathcal{F}_\alpha(R_{n-1}) = R_n = R_{n-1}$, which indicates that there can not be such a P.

"$\supseteq$" Whenever $(x, y) \notin \nu\mathcal{F}_\alpha = R_n$, we have shown in Proposition 3.4.3 that the spoiler has a winning strategy, which implies $(x, y) \notin \mathcal{W}_\alpha$. Hence $\mathcal{W}_\alpha \subseteq \nu\mathcal{F}_\alpha$.

□

Before we proceed to compare Algorithm 3.1 with the results in [DM+17], we first demonstrate the construction of a winning strategy for the spoiler over a simple example.

Example 3.4.8

We come back to Example 3.3.4 and explain the execution of Algorithm 3.1. In the first iteration we only have to consider one predicate χ_X and for all separated pairs of states (s, t) we set $I(s, t) = 1$ where the second component of $T(s, t)$ is X.

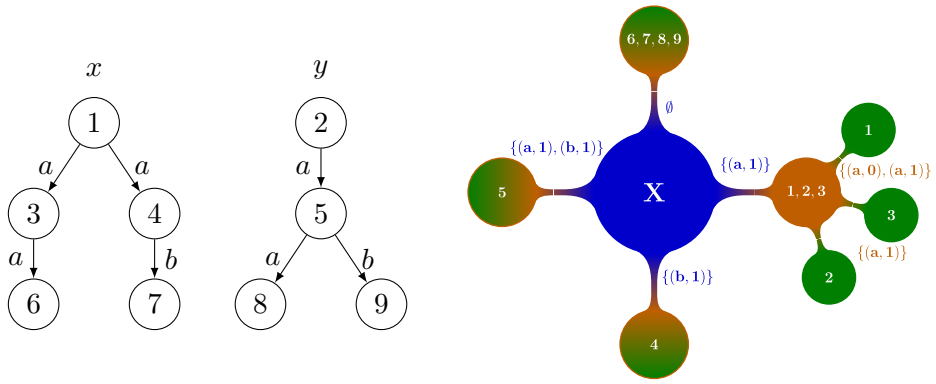


Figure 3.14: Three partition refinement steps starting from the blue partition $\Pi_0 = \{X\}$ followed by Π_1. The refinement terminates because of $\Pi_2 = \Pi_3$. A value $v \in F2$ (e.g. $\{(a, 1), (b, 1)\}$) written next to the branch in colour i (e.g. blue) indicates that only the states within the ball (e.g. $\{5\}$) according to that branch satisfy this value for a class (e.g. X) included in the i-coloured partition.

Therefore, in the first iteration, the states are simply divided into equivalence classes according to their outgoing transitions. More concretely, we obtain the separation of $\{1, 2, 3\}$ (with value $\{(a, 1)\}$) from $\{4\}$ (with value $\{(b, 1)\}$), $\{5\}$ (with value $\{(a, 1), (b, 1)\}$ and $\{6, 7, 8, 9\}$ (with value $\emptyset$) as illustrated in Figure 3.14.

In the second iteration the predicate $\chi_{\{4\}}$ is employed to separate $\{1\}$ (with

value $\{(a,0),(a,1)\}$) from $\{2\}$ (with value $\{(a,0)\}$) and we get $I(1,2) = 2$ with $T(1,2) = (1,\{4\})$, which also determines the strategy of the spoiler explained earlier. Similarly $\{3\}$ can be separated from both $\{1\}$ and $\{2\}$ with the predicate $\chi_{\{6,7,8,9\}}$. The other three classes obtained during the first iteration step do not change from then on.

The notion of separability by singletons is needed because the partition refinement algorithm we are using separates two states based on a *single* equivalence class of their successors, whereas other partition refinement algorithms (e.g. [KK18]) consider *all* equivalence classes. As shown in Example 3.4.6, this is indeed a restriction, however such additional assumptions seem necessary if we want to adapt efficient bisimulation checking algorithms such as the ones by Kanellakis/Smolka [KS90] or Paige/Tarjan [PT87] to the coalgebraic setting. In fact, the Paige/Tarjan algorithm already has a coalgebraic version [DM+17] which operates under the assumption that the functor is *zippable*. Here we show that the related notion of m-zippability is very similar to separability by singletons. The zippability of [DM+17] is in fact 2-zippability, which is strictly weaker than 3-zippability [Wiß]. Note, that the sufficiency of 2-zippability results from the basic idea in [DM+17] to work in one step with a coarser and a finer partition.

Lemma 3.4.9

Assume that $F\colon \mathbf{Set} \to \mathbf{Set}$ is a functor preserving injections. If F is separable by singletons, then F is m-zippable for all m. Conversely, if F is m-zippable, then F is separable by singletons for all sets X with $|X| \leq m$.

Proof:

- Suppose that F is separable by singletons. We need to show that

$$F(A_1 + \cdots + A_m) \xrightarrow{\langle F(f_1),\ldots,F(f_m)\rangle} F(A_1+1) \times \cdots \times F(A_m+1)$$

is injective. Hence let $t_0, t_1 \in F(A_1 + \cdots + A_m)$ with

$$\langle F(f_1),\ldots,F(f_m)\rangle(t_0) = \langle F(f_1),\ldots,F(f_m)\rangle(t_1)$$

be given. The situation is depicted in Figure 3.15 below.

Now let $x_i \in A_i$ and consider the singleton predicate $\chi_{\{x_i\}}\colon A_1 + \cdots + A_m \to 2$, which decomposes as $\chi_{\{x_i\}} = h_{x_i} \circ f_i$ where $h_{x_i}\colon A_i + 1 \to 2$ is the characteristic function of x_i on $A_i + 1$ (see Figure 3.16 below).

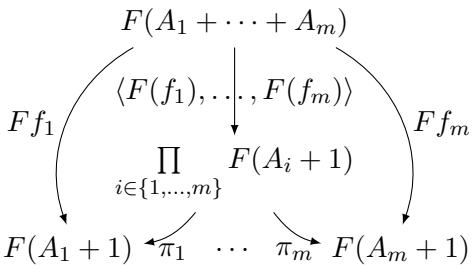


Figure 3.15

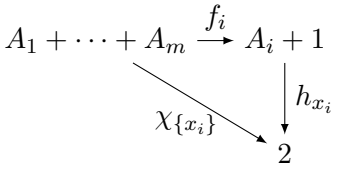


Figure 3.16

Now we can proceed as follows:

$$\begin{aligned} & \pi_i(\langle F(f_1), \ldots, F(f_m) \rangle(t_0)) = \pi_i(\langle F(f_1), \ldots, F(f_m) \rangle(t_1)) \\ \Rightarrow \quad & F(f_i)(t_0) = F(f_i)(t_1) \\ \Rightarrow \quad & Fh_{x_i}(F(f_i)(t_0)) = Fh_{x_i}(F(f_i)(t_1)) \\ \Rightarrow \quad & F\chi_{\{x_i\}}(t_0) = F\chi_{\{x_i\}}(t_1) \end{aligned}$$

Since this holds for all x_i in $A_1 + \cdots + A_m$, and F is separable by singletons, we can conclude that $t_0 = t_1$.

- We first observe that every functor that is m-zippable is also m'-zippable for $m' \leq m$ (just set $A_i = \emptyset$ for some i). Hence it is sufficient to prove that whenever F is m-zippable, then it is separable by singletons for all sets X with $|X| = m$. So we can assume without loss of generality that $X = \{x_1, \ldots, x_m\}$.

  We set $A_i = \{x_i\}$ and know from the premise that

  $$F(A_1 + \cdots + A_m) \xrightarrow{\langle F(f_1), \ldots, F(f_m) \rangle} F(A_1 + 1) \times \cdots \times F(A_m + 1)$$

  is injective (see Figure 3.15).

  Let $t_0, t_1 \in FX$ and $t_0 \neq t_1$ be given. Due to the injectivity of the map above, we know that there exists an index i such that $Ff_i(t_0) \neq Ff_i(t_1)$. Since $A_i + 1 \cong 2$, every f_i is itself a singleton predicate and hence we witness the inequality of t_0, t_1 via a singleton.

□

Runtime Analysis

We assume that X is finite and that the inequalities in Algorithm 3.1 (with respect to $\leq^F$) are decidable in polynomial time. Then our algorithm terminates and has polynomial runtime.

In fact, if $|X| = n$, the algorithm runs through at most n iterations, since there can be at most n splits of equivalence classes. In each iteration we consider up to n^2 pairs of states, and in order to decide whether a pair can be separated, we have to consider up to n equivalence classes, which results in $O(n^4)$ steps (not counting the steps required to decide the inequalities).

For a finite label set A, the inequalities are decidable in linear time for the functors in our examples ($F = \mathcal{P}_f(A \times \_)$ and $F = (\mathcal{D}\_ + 1)^A$). We expect that we can exploit optimizations based on [KS90; PT87]. In particular one could incorporate the generalization of the Paige-Tarjan algorithm to the coalgebraic setting [DM+17] which is presented in [WMS21].

To conclude, in general, the game yields a generic partition refinement algorithm, but to admit a polynomial runtime, we need to restrict to m-zippable functors. At the same time, the algorithm solves our main goal, namely to derive for all non-bisimilar state pairs the spoiler's winning strategies coalgebraically. To our knowledge, this is a new contribution to the field of coalgebraic partition refinement algorithms. The next section discusses how these strategies can be transformed into distinguishing formulas.

3.4.3 From Winning Strategies to Distinguishing Formulas

Using modal logic formulas, one can define the specifications a system has to satisfy. Therefore, a formula that indicates that an implementation x differs from the specification y serves as a witness for this inconsistency.

Example 3.4.10

Recall Example 2.2.2 where I_y represents a specification and I_x an implementation. The formula $\varphi = \Box_{register}(\Diamond_{pay} tt \ \wedge \ \Diamond_{add_{Hotel}} \Diamond_{pay} tt)$ distinguishes I_x, I_y since $I_y \vDash \varphi$ and $I_x \nvDash \varphi$.

Next we illustrate how to derive a distinguishing modal logic formula from the winning strategy of $\mathbf{S}$ computed by Algorithm 3.1. The other direction (obtaining the winning strategy from a distinguishing formula) has been covered in Theorem 3.3.7.

We focus on an on-the-fly extraction of relevant modalities, to our knowledge a new contribution, and discuss the connection to other – given – sets of separating predicate liftings.

One way of enabling the construction of formulas is to specify the separating set of predicate liftings Λ in advance. But this set might be infinite and hard to represent.

Instead here we generate the modalities while constructing the formula. We focus in particular on so-called *cone modalities*: given $v \in F2$ we take the upward-closure of v as a modality.

We also explain how logical formulas with cone modalities can be translated into other separating sets of modalities.

Definition 3.4.11: Cone Modalities

Let $v \in F2$. A cone modality $[\uparrow v]$ is given by the following evaluation map $\uparrow v : F2 \to 2$:

$$\uparrow v(u) = ev(u) = \begin{cases} 1, & \text{if } v \leq^F u \\ 0, & \text{otherwise} \end{cases}$$

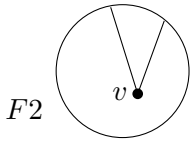


Figure 3.17: Cone of value $v \in F2$.

Obviously these evaluation maps yield a separating set of predicate liftings (provided that the functor has such a set): if $v_0 \neq v_1$ (for $v_0, v_1 \in F2$), either $v_0 \not\leq^F v_1$ or $v_1 \not\leq^F v_0$, since we require that the lifted order is anti-symmetric on $F2$. In the first case $\uparrow v_0(v_0) = 1$ and $\uparrow v_0(v_1) = 0$, in the second case $\uparrow v_1$ is the distinguishing evaluation map.

Example 3.4.12

We discuss modalities respectively evaluation maps in more detail for the functor $F = \mathcal{P}_f(A \times \_)$ (see also Example 3.3.4). In our example $A = \{a, b\}$. The set $F2$ with order $\leq^F$ is depicted as a Hasse diagram in Figure 3.18.

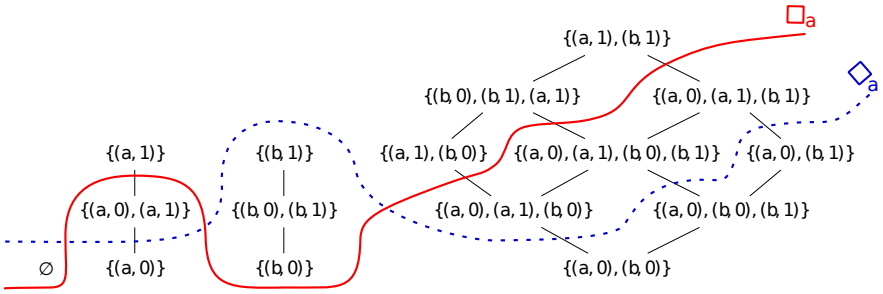


Figure 3.18: Set $F2 = \mathcal{P}_f(\{a, b\} \times 2)$ with order $\leq^F$ (for labelled transition systems). $\Box_a$ and $\Diamond_a$ are given by all values above the drawn (dashed) lines.

For every element there is a cone modality, 16 modalities in total. It is known from the Hennessy-Milner theorem [HM85] that two modalities are enough: either $\Box_a, \Box_b$ (box modalities) or $\Diamond_a, \Diamond_b$ (diamond modalities), where for $v \in F2$:

$$\Box_a(v) = \begin{cases} 1 & \text{if } v \cap \{(a,0)\} = \emptyset \\ 0 & \text{otherwise} \end{cases} \qquad \Diamond_a(v) = \begin{cases} 1 & \text{if } (a,1) \in v \\ 0 & \text{otherwise} \end{cases}$$

In Figure 3.18 $\Box_a$ respectively $\Diamond_a$ are represented by the elements above the two lines (solid respectively dashed).

Example 3.4.13

As a second example we discuss the functor $F = (\mathcal{D}\_+1)^A$, specifying probabilistic transition systems. The singleton set $1 = \{\bullet\}$ denotes termination. Again we set $A = \{a, b\}$.
Note that since $\mathcal{D}2$ is isomorphic to the interval $[0,1]$, we can simply represent any distribution $d\colon 2 \to [0,1]$ by $d(1)$. Hence $F2 \cong ([0,1]+1)^A$.

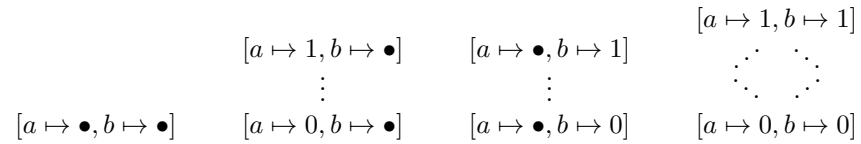


Figure 3.19: $F2 \cong ([0,1]+1)^A$ with order $\leq^F$ (for probabilistic transition systems).

The partial order is componentwise and is depicted in Figure 3.19: it decomposes into four disjoint partial orders, depending on whether both a, b, neither or one of them is mapped to $\bullet$. The right-hand part of this partial order consists of function $[0,1]^A$ with the pointwise order. We will also abbreviate a map $[a \mapsto p, b \mapsto q]$ by $\langle a_p, b_q \rangle$.

We will now show how a winning strategy of **S** can be transformed into a distinguishing formula, based on cone modalities, including some examples.

The basic idea behind the construction in Definition 3.4.14 is the following: Let (x, y) be a pair of states separated during iteration i of the partition refinement algorithm (Algorithm 3.1). This means that we have the following situation: $F\chi_P(\alpha(x)) \not\leq^F F\chi_P(\alpha(y))$ (or vice versa) for some equivalence class P of R_{i-1}. Based on $v = F\chi_P(\alpha(x))$ we define a cone modality $ev = \uparrow v$. Now, if we can

characterize P by some formula ψ, i.e., $[\![\psi]\!] = \chi_P$ (we will later show that this is always possible), we can define the formula $\varphi = [ev]\psi$. Then it holds that:

$$[\![\varphi]\!](x) = ev(F[\![\psi]\!](\alpha(x))) = \uparrow v(F\chi_P(\alpha(x))) = 1$$
$$[\![\varphi]\!](y) = ev(F[\![\psi]\!](\alpha(y))) = \uparrow v(F\chi_P(\alpha(y))) = 0$$

That is we have $x \models \varphi$ and $y \not\models \varphi$, which means that we have constructed a distinguishing formula for x, y.

First, we describe how a winning strategy for the spoiler for a pair (x, y) is converted into a formula and then prove that this formula distinguishes x, y.

Definition 3.4.14

Let $x \nsim y$ (equivalently $(x, y) \notin R_n$) and let (T, I) be the winning strategy for the spoiler computed by Algorithm 3.1. We construct a formula $\varphi_{x,y}$ as follows: assume that $T(x, y) = (s, P)$ where $s = x$. Then set $v = F\chi_P(\alpha(x))$, $ev = \uparrow v$ and define $\varphi_{x,y} = [ev]\varphi$, where φ is constructed in the following way:

- $I(x, y) = 1$: $\varphi = tt$
- $I(x, y) > 1$: $\varphi = \bigvee_{x' \in P} \big(\bigwedge_{y' \in X \setminus P} \varphi_{x',y'} \big)$

Whenever we have $s = y$ we instead define $v = F\chi_P(\alpha(y))$ and $\varphi_{x,y} = \neg[ev]\varphi$.

This encoding is well-defined, because it always holds that $I(x', y') < I(x, y)$ (since P is an equivalence class of R_{i-1} where $i = I(x, y)$).

Proposition 3.4.15

Let $\alpha : X \to FX$ be a coalgebra and assume that we have computed R_n, T, I with Algorithm 3.1. Then, given $(x, y) \notin R_n$, the construction in Definition 3.4.14 yields a formula $\varphi_{x,y} \in \mathcal{L}^\kappa(\Lambda)$ such that $x \vDash \varphi_{x,y}$ and $y \nvDash \varphi_{x,y}$.

Proof: We prove this by induction over $i = I(x, y)$:

$i = 1$: x, y have been separated at Step 1, since $F\chi_X(\alpha(x)) \not\leq^F F\chi_X(\alpha(x))$, where $T(x, y) = (x, X)$ (or vice versa), because X is the only equivalence class so far. Note also that $\varphi = tt$ and $[\![tt]\!] = X$.

We set $v = F\chi_X(\alpha(x))$, $ev = \uparrow v$ and we have

$$\begin{aligned} [\![\varphi_{x,y}]\!](x) &= ev(F[\![\varphi]\!](\alpha(x))) = ev(F\chi_X(\alpha(x))) = ev(v) = 1 \\ [\![\varphi_{x,y}]\!](y) &= ev(F[\![\varphi]\!](\alpha(y))) = ev(F\chi_X(\alpha(y))) = 0 \end{aligned}$$

Hence $x \models \varphi_{x,y}$ and $y \not\models \varphi_{x,y}$.

In the case where $T(x,y) = (y, X)$, we have $v = F\chi_X(\alpha(y))$, $ev = \uparrow v$ and we obtain

$$\begin{aligned} [\![[ev]\varphi]\!](x) &= ev(F[\![\varphi]\!](\alpha(x))) = ev(F\chi_X(\alpha(x))) = 0 \\ [\![[ev]\varphi]\!](y) &= ev(F[\![\varphi]\!](\alpha(y))) = ev(F\chi_X(\alpha(y))) = ev(v) = 1 \end{aligned}$$

Hence again $x \models \varphi_{x,y}$ and $y \not\models \varphi_{x,y}$.

$i \to i+1$: Due to the induction hypothesis we can assume that the $\varphi_{x',y'}$ are distinguishing formulas for (x', y') with $I(x', y') < i + 1$.

First, we show that $[\![\varphi]\!] = P$.

- Let $z \in P$. Then there exists an $x' \in P$ (namely $x' = z$) such that $z \models \varphi_{x',y'}$ for all $y' \notin P$. Furthermore, by construction of $\varphi_{x',y'}$ it holds that $y' \not\models \varphi_{x',y'}$. This means that $z \models \bigwedge_{y' \in X \setminus P} \varphi_{x',y'}$ and also $z \models \bigvee_{x' \in P} \bigwedge_{y' \in X \setminus P} \varphi_{x',y'} = \varphi$.
- Let $z \notin P$. Then for every $x' \in P$ there exists an $y' \notin P$ (namely $y' = z$) such that $z \not\models \varphi_{x',y'}$. Hence $z \not\models \bigwedge_{y' \in X \setminus P} \varphi_{x',y'}$. Since this is true for every such x' we also have $z \not\models \bigvee_{x' \in P} \bigwedge_{y' \in X \setminus P} \varphi_{x',y'} = \varphi$.

Assume that $T(x,y) = (x, P)$ (the case $T(x,y) = (y, P)$ can be handled analogously as for $i = 1$). Hence we know that $F\chi_P(\alpha(x)) \not\leq^F F\chi_P(\alpha(y))$.

We set $v = F\chi_P(\alpha(x))$, $ev = \uparrow v$ and we have

$$\begin{aligned} [\![\varphi_{x,y}]\!](x) &= ev(F[\![\varphi]\!](\alpha(x))) = ev(F\chi_P(\alpha(x))) = ev(v) = 1 \\ [\![\varphi_{x,y}]\!](y) &= ev(F[\![\varphi]\!](\alpha(y))) = ev(F\chi_P(\alpha(y))) = 0 \end{aligned}$$

Hence $x \models \varphi_{x,y}$ and $y \not\models \varphi_{x,y}$.

□

We next present an optimization of the construction in Definition 3.4.14, inspired by [Cle90]. In the case $I(x,y) > 1$ one can pick an arbitrary $x' \in P$ and keep only one element of the disjunction. In order to show that this simplification is permissible, we need the following lemma.

Lemma 3.4.16

Given two states $(x,y) \notin R_n$ and a distinguishing formula $\varphi_{x,y}$ based on Definition 3.4.14. Let (x', y') be given such that $I(x', y') > I(x,y)$. Then $x' \vDash \varphi_{x,y}$ if

and only if $y' \vDash \varphi_{x,y}$.

Proof: We have to distinguish two different cases for $I(x', y')$

$I(x', y') = 1$**:** this can not be true since we require $I(x', y') > I(x, y) \geq 1$.

$I(x', y') > 1$**:** For any (x, y) with $I(x, y) < I(x', y')$ we have $x \vDash \varphi_{x,y}$ and $y \nvDash \varphi_{x,y}$ where $\varphi_{x,y} = [ev]\varphi$, $ev = \uparrow F\chi_P(\alpha(x))$ and $T(x, y) = (x, P)$ (the case $T(x, y) = (y, P)$ is analogous). Furthermore, the semantics of φ is $[\![\varphi]\!] = \chi_P$ (for details we refer to the proof of Proposition 3.4.15). Now, assume without loss of generality that the following holds

$$1 = [\![\varphi_{x,y}]\!](x') = ev(F\chi_P(\alpha(x'))) \qquad 0 = \quad [\![\varphi_{x,y}]\!](y') = ev(F\chi_P(\alpha(y')))$$

Due to Proposition 3.2.17 ev is monotone. Therefore, the above assumption implies $F\chi_P(\alpha(x')) \not\leq^F F\chi_P(\alpha(y'))$. But this yields a contradiction, since then x', y' would have been separated in a Step $i \leq I(x, y) < I(x', y')$.

□

Now we can show that we can replace the formula φ from Definition 3.4.14 by a simpler formula φ'.

Lemma 3.4.17

Let $(x, y) \notin R_i$ and let P be an equivalence class of R_{i-1}. Furthermore let

$$\varphi' = \bigwedge_{y' \in X \setminus P} \varphi_{x',y'}$$

for some $x' \in P$. Then $[\![\varphi']\!] = \chi_P$.

Proof: Clearly $[\![\varphi']\!] \leq [\![\varphi]\!] = \chi_P$.

We now have to show that the other inequality holds as well, so let $z \in P$. Furthermore let y' be arbitrary such that $y' \notin P$. Since $z, x' \in P$ and $y' \notin P$, where P is an equivalence class, we know that $I(z, x') > I(x', y')$ (possibly even $I(z, x') = \infty$). Hence, by Lemma 3.4.16 we have that $z \models \varphi_{x',y'}$ if and only if $x' \models \varphi_{x',y'}$. And since the latter holds, we have $z \models \varphi_{x',y'}$.

Hence $z \models \bigwedge_{y' \in X \setminus P} \varphi_{x',y'} = \varphi'$. In summary, we get $\chi_P \leq [\![\varphi']\!]$. □

Finally, we can simplify our construction described in Definition 3.4.14 to only one inner conjunction.

Corollary 3.4.18

We use the construction of $\varphi_{x,y}$ as described in Definition 3.4.14 with the only modification that for $I(x,y) > 1$ the formula φ is replaced by

$$\varphi' = \bigwedge_{y' \in X \setminus P} \varphi_{x',y'}$$

for some $x' \in P$. Then this yields a formula $\varphi_{x,y}$ such that $x \vDash \varphi_{x,y}$ and $y \nvDash \varphi_{x,y}$.

A further optimization takes only one representative y' from every equivalence class different from P.

We now explore two slightly more complex examples to demonstrate Corollary 3.4.18 , where the second example also refers to Corollary A.3.1.

Example 3.4.19

Take the coalgebra for the functor $F = (\mathcal{D}\_ + 1)^A$ depicted in Figure 3.20. Note that $A = \{a, b\}$ and $X = \{1, \ldots, 5\}$. We have for instance $\alpha(3) = [a \mapsto \delta_3, b \mapsto \bullet]$ where δ_3 is the Dirac distribution. This is visualized by drawing an arrow labelled $a, 1$ from 3 to 3 and omitting b-labelled arrows.

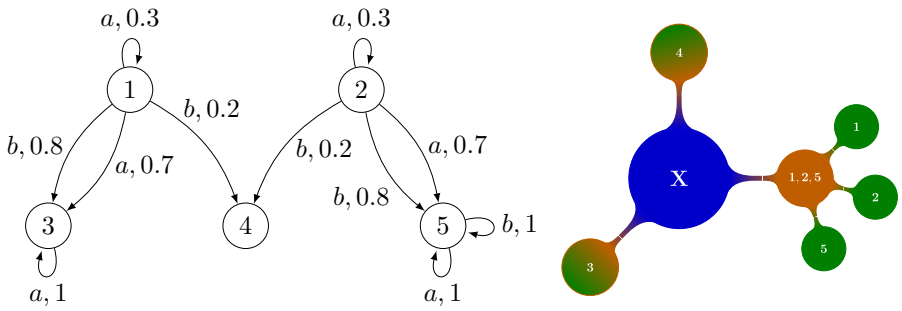


Figure 3.20: Probabilistic transition system: Three partition refinement steps starting from the blue partition $\Pi_0 = \{X\}$ followed by Π_1. The refinement terminates because of $\Pi_2 = \Pi_3$.

We explain only selected steps of the construction: in the first step the partition refinement algorithm (Algorithm 3.1) separates 1 from 3 (among other separations), where the spoiler strategy is given by $T(1,3) = (1, X)$. In order to obtain a distinguishing formula we determine $v = F\chi_X(\alpha(1)) = \langle a_1, b_1 \rangle$ (using the abbreviations explained in Example 3.4.13) and obtain $\varphi_{1,3} = [\uparrow \langle a_1, b_1 \rangle] tt$. In fact, this formula also distinguishes 1 from 4, hence $\varphi_{1,3} = \varphi_{1,4}$. If, on the other

hand, we would like to distinguish $3, 4$, we would obtain $\varphi_{3,4} = [\uparrow\langle a_1, b_\bullet\rangle]tt$.
After the first iteration, we obtain the partition $\{1, 2, 5\}, \{3\}, \{4\}$. Now we consider states $1, 2$ which can be separated by playing $T(2, 1) = (2, \{1, 2, 5\})$, since 5 behaves differently from 3. Again we compute $v = F\chi_P(\alpha(2)) = \langle a_1, b_{0.8}\rangle$ (for $P = \{1, 2, 5\}$) and obtain $\varphi_{2,1} = [\uparrow\langle a_1, b_{0.8}\rangle](\varphi_{1,3} \wedge \varphi_{1,4})$. Here we picked 1 as the representative of its equivalence class.
In summary we obtain $\varphi_{2,1} = [\uparrow\langle a_1, b_{0.8}\rangle][\uparrow\langle a_1, b_1\rangle]tt$ that is satisfied by 2, but not by 1.

Example 3.4.20

We will now give an example where conjunction is required to obtain the distinguishing formula. We work with a coalgebra for the functor $F = \mathcal{P}_f(A \times \_)$, which is depicted in Figure 3.21. Note that $A = \{a, b, c, d, e, f\}$ and $X = \{1, \ldots, 9\}$.

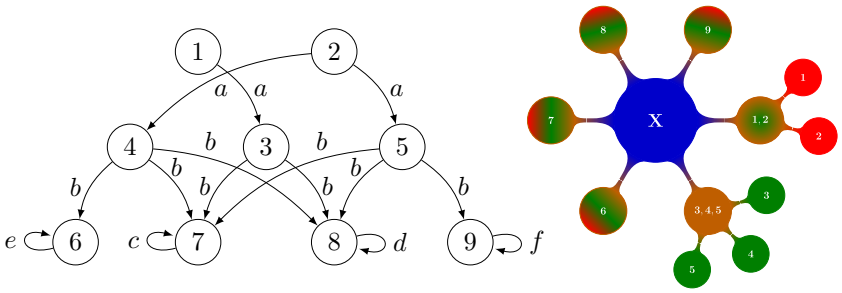


Figure 3.21: For the non-deterministic labelled transition system three partition refinement steps are necessary to obtain the distinguishing formula $\varphi_{1,2}$.

We explain only selected steps: in the first step the partition refinement separates 6 from 7 (among other separations), where the spoiler strategy is given by $T(6, 7) = (6, X)$. As explained above we determine $v = F\chi_X(\alpha(6)) = \{(e, 1)\}$ and obtain $\varphi_{6,7} = [\uparrow\{(e, 1)\}]tt$. In fact, this formula also distinguishes 6 from all other states, so we denote it by $\varphi_{6,*}$.
Next, we consider states $3, 4$, where the possible moves of 3 are a strict subset of the moves of 4. Hence the spoiler strategy is $T(3, 4) = (4, \{6\})$, i.e., the spoiler has to move to state 6 that is not reachable from 3. Again we compute $v = F\chi_P(\alpha(4)) = \{(b, 1), (b, 0)\}$ (for $P = \{6\}$) and obtain:

$$\varphi_{3,4} = \neg[\uparrow\{(b, 1), (b, 0)\}]\varphi_{6,*}$$

Note that this time we have to use negation, since the spoiler moves from the

second state in the pair.

Finally we regard states $1, 2$ where the spoiler strategy is $T(1,2) = (1, \{3\})$. We compute $v = F\chi_P(\alpha(1)) = \{(a,1)\}$ (for $P = \{3\}$) and derive:

$$\varphi_{1,2} = [\uparrow\{(a,1)\}](\bigwedge_{x\in\{1,2,4,\ldots,9\}} \varphi_{3,x})$$

In fact, here it is sufficient to consider $x = 4$ and $x = 5$, resulting in the following distinguishing formula:

$$[\uparrow\{(a,1)\}](\neg[\uparrow\{(b,0),(b,1)\}][\uparrow\{(e,1)\}]tt \wedge \neg[\uparrow\{(b,0),(b,1)\}][\uparrow\{(f,1)\}]tt).$$

Note, that for the conjunction of $\varphi_{1,2}$ it suffices to work with $\{4,5,3\} \setminus \{3\}$ instead of $X \setminus \{3\}$ (see Corollary A.3.1).

Regarding the two examples above, it seems that *cone modalities* capture similar intuitions as the standard modalities. Therefore, we analyze how the evaluation maps for *cone modalities* can be transformed into other modalities.

3.4.4 Recoding Modalities

Finally, we will show under which conditions one can encode cone modalities into generic modalities, given by a separating set of predicate liftings Λ, not necessarily monotone. We first need the notion of strong separation.

Definition 3.4.21

Let Λ be a separating set of predicate liftings of the form $ev\colon F2 \to 2$. We call Λ *strongly separating* if for every $t_0 \neq t_1$ with $t_0, t_1 \in F2$ there exists $ev \in \Lambda$ such that $ev(t_0) \neq ev(t_1)$.

We can generate a set of strongly separating predicate liftings from every separating set of predicate liftings.

Lemma 3.4.22

Let Λ be a separating set of predicate liftings. Furthermore we denote the four functions on 2 by id_2, *one* (constant 1-function), *zero* (constant 0-function) and *neg* ($neg(0) = 1$, $neg(1) = 0$).

Then

$$\Lambda' = \{ev = ev \circ Fid_2, ev \circ Fone, ev \circ Fzero, ev \circ Fneg \mid ev \in \Lambda\}$$

is a set of strongly separating predicate liftings.

Furthermore for every formula φ we have that

$$[ev \circ F one]\varphi \equiv [ev]tt \qquad [ev \circ F zero]\varphi \equiv [ev]ff \qquad [ev \circ F neg]\varphi \equiv [ev](\neg\varphi)$$

Proof:

Let $t_0, t_1 \in F2$ with $t_0 \neq t_1$. According to the definition of separation there must be a predicate $p\colon 2 \to 2$ such that $ev(Fp(t_0)) \neq ev(Fp(t_1))$. Since there are only four such functions, p must be one of id_2, *one*, *zero*, *neg* and we immediately obtain that Λ' is strongly separating.

In addition we have that, given a coalgebra $\alpha\colon X \to FX$:

$$\begin{aligned}
[\![[ev \circ F one]\varphi]\!] &= ev \circ F one \circ F[\![\varphi]\!] \circ \alpha = ev \circ F(one \circ [\![\varphi]\!]) \circ \alpha \\
&= ev \circ F[\![tt]\!] \circ \alpha = [\![[ev]tt]\!] \\
[\![[ev \circ F zero]\varphi]\!] &= ev \circ F zero \circ F[\![\varphi]\!] \circ \alpha = ev \circ F(zero \circ [\![\varphi]\!]) \circ \alpha \\
&= ev \circ F[\![ff]\!] \circ \alpha = [\![[ev]ff]\!] \\
[\![[ev \circ F neg]\varphi]\!] &= ev \circ F neg \circ F[\![\varphi]\!] \circ \alpha = ev \circ F(neg \circ [\![\varphi]\!]) \circ \alpha \\
&= ev \circ F[\![\neg\varphi]\!] \circ \alpha = [\![[ev]\neg\varphi]\!]
\end{aligned}$$

□

This means that we can still express the new modalities with the previous ones. Λ' is just an auxiliary construct that helps us to state the following proposition. The construction of Λ' from Λ was already considered in [Sch08, Definition 24], where it is called *closure*.

Proposition 3.4.23

Suppose that $F2$ is finite, and let Λ be a strongly separating set of predicate liftings. Moreover, let $v \in F2$, and let φ be a formula. For $u \in F2$, we write $\Lambda_u = \{ev \in \Lambda \mid ev(u) = 1\}$. Then

$$[\uparrow v]\varphi \equiv \bigvee_{v \leq^F u} \Big(\bigwedge_{ev \in \Lambda_u} [ev]\varphi \wedge \bigwedge_{ev \notin \Lambda_u} \neg[ev]\varphi \Big).$$

Proof:

First observe that since Λ is strongly separating, every $u \in F2$ is characterized uniquely by Λ_u.

Let $\alpha\colon X \to FX$ be a coalgebra. We set $\psi_u = \bigwedge_{ev \in \Lambda_u} [ev]\varphi \wedge \bigwedge_{ev \notin \Lambda_u} \neg[ev]\varphi$ and we first show that

$$x \models \psi_u \iff u = F[\![\varphi]\!](\alpha(x))$$

$\Rightarrow$: Assume that $x \models \psi_u$. This means that for every $ev \in \Lambda_u$ we have that $ev(F[\![\varphi]\!](\alpha(x))) = 1$ and for every $ev \notin \Lambda_u$ we have that $ev(F[\![\varphi]\!](\alpha(x))) = 0$. This means that u and $F[\![\varphi]\!](\alpha(x))$ are both characterized by Λ_u and from the strong separation property it follows that they are equal, i.e., $u = F[\![\varphi]\!](\alpha(x))$.

$\Leftarrow$: Assume that $u = F[\![\varphi]\!](\alpha(x))$. Then for every $ev \in \Lambda_u$ we have that $[\![[ev]\varphi]\!](x) = ev(F[\![\varphi]\!](\alpha(x))) = ev(u) = 1$. For every $ev \notin \Lambda_u$ we obtain $[\![[ev]\varphi]\!](x) = 0$. Everything combined, we have $[\![\psi_u]\!](x) = 1$ and hence $x \models \psi_u$.

We can conclude the proof by observing that

$$x \models [\uparrow v]\varphi \iff v \leq^F F[\![\varphi]\!](\alpha(x)) \iff \exists u\colon \Big(v \leq^F u \wedge u = F[\![\varphi]\!](\alpha(x))\Big)$$
$$\iff \exists u\colon \Big(v \leq^F u \wedge x \models \psi_u\Big) \iff x \models \bigvee_{v \leq^F u} \psi_u$$

□

By performing this encoding inductively, we can transform a formula with cone modalities into a formula with modalities in Λ. The encoding preserves negation and conjunction, only the modalities are transformed.

Example 3.4.24

We come back to labelled transition systems and the functor $F = \mathcal{P}_f(A \times \_)$ with $A = \{a, b\}$. In this case the set $\{\Box_a, \Box_b, \Diamond_a, \Diamond_b\}$ of predicate liftings is strongly separating.
Now let $v = \{(a, 0), (b, 1)\} \in \mathcal{P}_f(A \times 2)$ and we show how to encode the corresponding cone modality using only box and diamond:

$$\begin{aligned}[\uparrow v]\varphi \quad \equiv \quad & (\neg\Box_a\varphi \wedge \Box_b\varphi \wedge \neg\Diamond_a\varphi \wedge \Diamond_b\varphi) \vee (\neg\Box_a\varphi \wedge \Box_b\varphi \wedge \Diamond_a\varphi \wedge \Diamond_b\varphi) \\ & \vee (\Box_a\varphi \wedge \Box_b\varphi \wedge \Diamond_a\varphi \wedge \Diamond_b\varphi)\end{aligned}$$

The first term describes $\{(a, 0), (b, 1)\}$, the second $\{(a, 0), (a, 1), (b, 1)\}$ and the third $\{(a, 1), (b, 1)\}$.

Given a cone modality $[\uparrow v]$ with $v \in F2$ the size of the formula depends on the number of disjunctions (i.e. $|\{u \in F2 \mid v \leq^F u\}|$) and the size of each conjunction is determined by the size of Λ. Note that we cannot directly generalize Proposition 3.4.23 to the case where $F2$ is infinite. The reason for this is that the disjunction over all $u \in F2$ such that $v \leq^F u$ might violate the cardinality constraints of the logic. Hence we will consider an alternative, where the re-coding works only under certain assumptions. We will start with the following example.

Example 3.4.25

Consider the functor $F = (\mathcal{D}\_ + 1)^A$ (see also Example 3.4.13) and the corresponding (countable) separating set of (monotone) predicate liftings

$$\Lambda = \{ev_{(a,q)} \colon F2 \to 2 \mid a \in A, q \in [0,1] \cap \mathbb{Q}_0\} \cup \{ev_{(a,\bullet)} \mid a \in A\}$$

where $ev_{(a,q)}(v) = 1$ if $v(a) \in \mathbb{R}_0$ and $v(a) \geq q$ and $ev_{(a,\bullet)} = 1$ if $v(a) = \bullet$. Here, a modality $[ev_{(a,q)}]$ indicates that the probability of making an a-transition is greater than or equal to q, and a modality $[ev_{(a,\bullet)}]$ tells us that we terminate with a.

The disjunction $\bigvee_{v \leq^F u}$ in the construction of $[\uparrow v]\varphi$ in Proposition 3.4.23 is in general uncountable and may hence fail to satisfy the cardinality constraints of the logic.

However, we can exploit certain properties of this set of predicate liftings, in order to re-code modalities.

Lemma 3.4.26

Let F be the functor with $F = (\mathcal{D}\_ + 1)^A$ and let Λ be the separating set of predicate liftings from Example 3.4.25. Furthermore let $v \in F2$. Then it holds that:

$$\uparrow v = \bigcap_{ev \in \Lambda, ev(v)=1} \hat{ev}$$

Proof:

"$\subseteq$" Let $u \in F2$ with $u \in \uparrow v$, i.e., $v \leq^F u$. Whenever $ev(v) = 1$ we also have $ev(u) = 1$ due to the monotonicity of the predicate liftings (cf. Proposition 3.2.15) and hence $u \in \hat{ev}$. Since this holds for all such ev, we can conclude that $u \in \bigcap_{ev \in \Lambda, ev(v)=1} \hat{ev}$.

"$\supseteq$" Now suppose by contradiction that we have $u \in F2$ with $v \not\leq^F u$ and $u \in \bigcap_{ev \in \Lambda, ev(v)=1} \hat{ev}$.

There are three cases which may cause $v \not\leq^F u$, in particular they are distinguished by a specific $a \in A$:

- $v(a), u(a) \in \mathbb{R}_0$, but $v(a) \not\leq u(a)$, which implies $u(a) < v(a)$. However, there exists $q \in [0,1] \cap \mathbb{Q}_0$ with $u(a) < q \leq v(a)$ and for the corresponding

modality $ev_{(a,q)} \in \Lambda$ we have $ev_{(a,q)}(u) = 0$, $ev_{(a,q)}(v) = 1$ and hence $u \notin \bigcap_{ev\in\Lambda, ev(v)=1} \hat{ev}$.

- $v(a) \in \mathbb{R}_0$, $u(a) = \bullet$: Now take any $q \in [0,1] \cap \mathbb{Q}_0$ with $q \leq v(a)$. We use the modality $ev_{(a,q)}$, for which we have $ev_{(a,q)}(u) = 0$, $ev_{(a,q)}(v) = 1$ and the proof proceeds as before.
- $v(a) = \bullet$, $u(a) \in \mathbb{R}_0$: Now we take the modality $ev_{(a,\bullet)}$, for which we have $ev_{(a,\bullet)}(u) = 0$, $ev_{(a,\bullet)}(v) = 1$ and again the proof proceeds as before.

□

Note that this property does not hold for the $\Box$ and $\Diamond$ modalities for the functor $F = \mathcal{P}_f(A \times \_)$. This can be seen via Figure 3.18, where the upward closure of $\{(b,0)\}$ contains three elements. However, $\{(b,0)\}$ is only contained in the modality $\Box_a$ (and no other modality), which does not coincide with the upward-closure of $\{(b,0)\}$.

Next we show the following proposition, which gives us a recipe for transforming cone modalities that satisfy the properties of Lemma 3.4.26 into the given modalities.

Proposition 3.4.27

Given a set $\Lambda' \subseteq \Lambda$ of predicate liftings we have

$$[\bigcap_{ev\in\Lambda'} ev]\varphi \equiv \bigwedge_{ev\in\Lambda'} [ev]\varphi.$$

Proof:

"$\subseteq$" Let $x \vDash [\bigcap_{ev\in\Lambda'} ev]\varphi$, which implies that $(\bigcap_{ev\in\Lambda'} ev)(F[\![\varphi]\!](\alpha(x))) = 1$. From this we conclude that $ev(F[\![\varphi]\!](\alpha(x))) = 1$ for all $ev \in \Lambda'$, $x \models [ev]\varphi$. And finally we have $x \models \bigwedge_{ev\in\Lambda'}[ev]\varphi$.

"$\supseteq$" Let $x \vDash \bigwedge_{ev\in\Lambda'}[ev]\varphi$, which means that $x \vDash [ev]\varphi$ for all $ev \in \Lambda'$. This implies that $ev(F[\![\varphi]\!](\alpha(x))) = 1$. Hence we obtain $(\bigcap_{ev\in\Lambda'} ev)(F[\![\varphi]\!](\alpha(x))) = 1$ and finally $x \vDash [\bigcap_{ev\in\Lambda'} ev]\varphi$.

□

Note that this construction might again violate the cardinality constraints of the logic. In particular, for the probabilistic case (Example 3.4.13) we have finite formulas, but countably many modalities. However, if we assume that the set of labels A is finite and restrict the coefficients in the coalgebra to rational numbers,

every cone modality can be represented as the intersection of only finitely many minimal given modalities and so the encoding preserves finiteness.

3.5 Conclusion and Discussion

At the beginning of this chapter, we noted that, in contrast to labelled transition systems, there was no game-based view in the sense of predicate liftings, which forms the base for coalgebraic modal logic. In summary, the contribution of our work rests on two areas:

▷ First of all, we present a coalgebraic game characterization for bisimulation. Our contribution generalizes the games of [DLT08; Sti99] and allows us to derive the corresponding game for each branching type defined by a weak pullback preserving functor over the category **Set**. In addition, the game is strongly linked to the existence of monotone and separating modalities, which are implied by an anti-symmetric lifted order $\leq^F$. It is quite natural to consider monotone modalities as they are widely used in most common case studies [LS89; BRS08].

▷ Secondly, we give concrete recipes for explaining non-bisimilarity in a coalgebraic setting. This involves the computation of the spoiler winning strategies based on a partition refinement algorithm as well as the generation of distinguishing formulas according to the ideas of [Cle90]. Therefore, we defined so-called *cone modalities* to avoid the specification of a separating set of monotone predicate liftings. Additionally, we provide encoding techniques into the corresponding standard modalities.

Regarding the first item, we are mainly aware of the work by Baltag [Bal00], which describes a coalgebraic game based on the bisimulation relation, which differs from the games studied in this paper and is associated with another variant of logic, namely Moss' coalgebraic logics [Mos99]. A variant of Baltag's game was used in [Kup07] for terminal sequence induction via games. Moreover, we are aware of games from a fibrational perspective in [KK+19] which is linked to our results about coalgebraic metric games [KM18] presented in Chapter 5. (There are more contributions on evaluation games which describe the evaluation of a modal formula on a transition system, see for instance [FLV10].) Incidentally, we expect that the bisimulation game can be extended to polyadic predicate liftings.

To our knowledge, a generic way that automatically derives explanations for non-bisimilar state pairs is new. The difference between a coalgebraic partition refinement algorithm [PT87; DM+17] that lifts the idea of Paige/Tarjan [PT87] to coalgebras is already discussed detailed in Section 3.4.1. The main difference results

from the so-called *three way splitting*, which selects equivalence classes for splitting in a clever way. Our results obtained from the game-based partition refinement algorithm point to the fact that a combination of our techniques to derive significant modalities with the *three way splitting* will improve the polynomial runtime behaviour and slacken the requirements (cf. [WMS21]).

For the generation of distinguishing formulas an option would be to fix the modalities a priori and to use them in the game, similar to the notion of λ-bisimulation [GS13; KM18]. However, there might be infinitely many modalities and the partition refinement algorithm can not iterate over all of them. A possible solution would be to find a way to check the conditions symbolically in order to obtain suitable modalities. Furthermore, we present optimization techniques wrt. the size of the distinguishing formula (cf. Appendix A.3.1) and we would like to compare our coalgebraic results with Cleaveland's definition of a minimal formula [Cle90].

Of course we are also interested in whether we can lift the extra assumptions that were necessary in order to re-code modalities in Section 3.4.4.

An interesting further idea is to translate the coalgebra into multi-neighbourhood frames [Han03; KW99], based on the predicate liftings, and to derive a λ-bisimulation game as in [KM18; GS13] from there. (The λ-bisimulation game does not require weak pullback preservation and extends the class of admissible functors, but requires us to fix the modalities rather than generate them.) One could go on and translate these multi-neighbourhood frames into Kripke frames, but this step unfortunately does not preserve bisimilarity.

4 Tools and Case Studies

The game and the generation of the distinguishing formulas introduced in Chapter 3 have been implemented in a tool called T-Beg. A second tool referred to as Paws (Program for the Analysis of Weighted Systems) has been mainly developed during my master thesis together with Sebastian Küpper, which primarily supports a fixed type of a weighted functor, where the user can change the semiring. The work presented in [KKM17] relies on the further development of Paws, including several extensive runtime tests.

Section 4.1 starts with a short introduction to T-Beg. The software design of the tool is presented in Section 4.1.1 and an interface which refers to the idea to represent transition systems of *branching type* F (i.e functors) via F-coalgebras is described in Section 4.1.2.

4.1 T-BEG: A Generic Tool for Games and the Construction of Distinguishing Formulas

A tool for playing bisimulation games is useful for teaching, for illustrating examples in talks, for case studies and in general for interaction with the user. There are already available tools, providing visual feedback to help the user understand why two states are (not) bisimilar, such as The Bisimulation Game Game[1] or Bisimulation Games Tools[2] (see [FKW17]). These games are designed for labelled transition systems and [FKW17] also covers branching bisimulation.

Our tool T-Beg goes beyond labelled transition system and allows to treat coalgebras in general (under the restrictions that we impose), that is, we exploit the categorical view to create a generic tool. As shown earlier in Sections 3.4.2 and 3.4.3, the coalgebraic game defined in Definition 3.3.1 provides us with a generic algorithm to compute the winning strategies and distinguishing formulas.

The user can either take on the role of the spoiler or of the duplicator, playing on some coalgebra against the computer. The tool computes the winning strategy (if

[1] `http://www.brics.dk/bisim/`

[2] `https://www.jeroenkeiren.nl/2017/02/23/on-games-and-simulations.html`

any) and follows this winning strategy if possible. We have also implemented the construction of the distinguishing formula for two non-bisimilar states.

The genericity over the functor is in practice achieved as follows: The user either selects an existing functor F (e.g. the running examples of [KMS20b]), or implements his/her own functor by providing the code of one class with nine methods (explained below). Everything else, such as embedding the functor into the game and the visualization are automatically handled by T-Beg.

Then, he/she enters or loads a coalgebra $\alpha : X \to FX$ (with X finite), stored as *csv* (comma separated value) file. Now the user can switch to the game view and start the game by choosing one of the two roles (spoiler or duplicator) and selecting a pair of states (x, y), based on the visual graph representation.

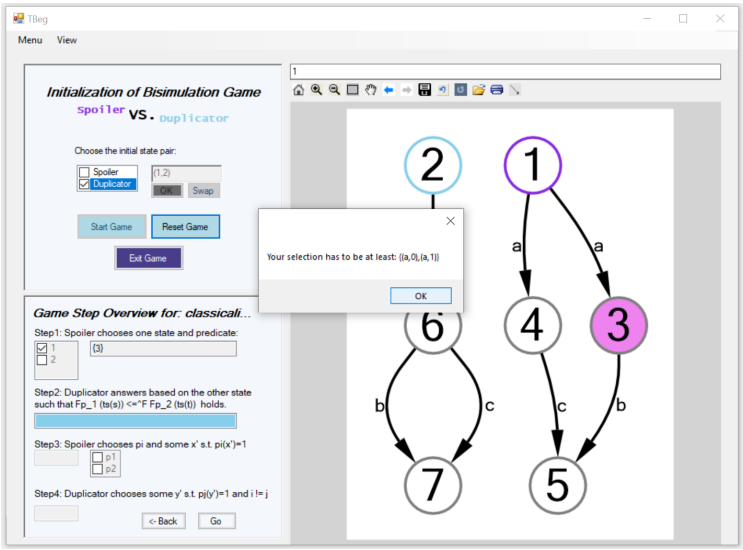


Figure 4.1: Screenshot of the graphical user interface with a game being played.

Next, the computer takes over the remaining role and the game starts: In the game overview, the user is guided through the steps by using two colors to indicate whether it is spoiler's (violet) or duplicator's (cyan) turn (see Figure 4.1).

In the case of two non-bisimular states, the tool will display a distinguishing formula at the end of the game.

4.1.1 Design

We now give an overview over the design and the relevant methods within the tool. We will also explain what has to be done in order to integrate a new functor.

T-Beg is a Windows tool offering a complete graphical interface, developed in Microsoft's Visual Studio using *C#*, especially *Generics*. It uses a graph library[3], which in turn provides a *GraphEditor* that allows for storing graphs as *MSAGL* files or as *png* and *jpg* files.

The program is divided into five components: Model, View, Controller, Game and Functor. We have chosen *MVC (Model View Controller)* as a modular pattern, so modules can be exchanged. Here we have several *Model*$\langle T \rangle$ managed by the *Controller*, where the functor in the sense of a *Functor* class, which always implements the *Functor Interface*, is indicated by the parameter $\langle T \rangle$.

While the tool supports more general functors, there is specific support for functors F with $F = V^{G(-)}$ where V specifies a semiring and G preserves finite sets. That is, F describes the branching type of a weighted transition system, where for instance $G = A \times (\_) + 1$ (introducing finitely many labels and termination). Coalgebras are of the form $X \to V^{GX}$ or – via currying – of the form $X \times GX \to V$, which means that they can be represented by $X \times GX$-matrices (matrices with index sets X, GX). In the implementation V is the generic data type of the matrix entries. In the case of the powerset functor we simply have $V = 2$ and $G = ID$.

If the branching type of the system can not simply be modelled as a matrix, there is an optional field that can be used to specify the system, since *Model*$\langle T \rangle$ calls the user-implemented method to initialize the F-coalgebra instance. The implementation of Algorithm 3.1 can be found in *Game*$\langle T, V \rangle$, representing the core of the tool's architecture, whose correctness is only guaranteed for functors that meet our requirements, such as the functors used in the paper [KMS20b].

4.1.2 Functor Interface

As mentioned previously, the user has to provide nine methods in order to implement the functor in the context of T-Beg: two are needed for the computation, two for rendering the coalgebra as a graph, one for creating modal formulas, another two for loading and saving, and two more for customizing the visual matrix representation.

We would like to emphasize here that the user is free to formally implement the functor in the sense of the categorical definition as long as the nine methods needed for the game are provided. In particular, we do not need the application of the functor to arrows since we only need to lift predicates $p : X \to 2$.

Within *MyFunctor*, which implements the interface *Functor*$\langle F, V \rangle$, the user defines the data structure F for the branching type of the transition system (e.g.,

[3] `https://www.nuget.org/packages/Microsoft.Msagl.GraphViewerGDI`

a list or bit vector for the powerset functor, or the corresponding function type in the case of the distribution functor). Further, the user specifies the type V that is needed to define the entries of $X \times GX$ (e.g. a double value for a weight or $0, 1$ to indicate the existence of a transition).

Then the following nine methods have to be provided:

Matrix$\langle F, V \rangle$*InitMatrix*(...)**:** This method initializes the transition system with the string-based input of the user. The information about the states and the alphabet is provided via an input mask in the form of a matrix.

bool CheckDuplicatorsConditionStep2(...): given two states x, y and two predicates p_1, p_2, this method checks whether

$$Fp_1(\alpha(x)) \leq^F Fp_2(\alpha(y)).$$

This method is used when playing the game (in Step 2) and in the partition refinement algorithm (Algorithm 3.1) for the case $p_1 = p_2$.

TSToGraph(...): This method handles the implementation of the graph-based visualization of the transition system. For weighted systems the user can rely on the default implementation included within the *Model*. In this case, arrows between states and their labels are generated automatically.

GraphToTS(...): This method is used for the other direction, i.e. to derive the transition system from a directed graph given by *Graph*.

string GetModalityToString(...): This method is essential for the automatic generation of the modal logical formulas distinguishing two non-bisimilar states as described in Definition 3.4.14. In each call, the cone modality that results from $F\chi_P(\alpha(s))$ with $T(x, y) = (s, P)$ is converted into a string.

SaveTransitionSystem(...): In order to store a transition system in a *csv* file.

LoadTransitionSystem(...): In order to load a transition system from a *csv* file.

GetRowHeadings(...): T-Beg can visualize a transition system $\alpha : X \to FX$ as a $X \times GX$ matrix within a *DataGrid*. For this purpose, the user needs to specify how the *RowHeaders* can be generated automatically.

ReturnRowCount(...): This method returns the number of rows of the matrix representing the coalgebra.

4.2 Case Study on Mealy Machines

Most of the state-based models used for explanations of the theories presented in this thesis are rather abstract in nature (see Examples in Section 2.2.1). This is due to the fact, that transition systems derived from real implementations are highly complex and therefore usually less demonstrative. (We refer to [DI02] which covers the translation of *C*-programs into LTS and the modeling of *Java*-programs is treated in [DH+01].)

Nevertheless, in order to build a bridge to the practical world we reconsider Mealy machines (see Definition 2.2.6). As already discussed in Section 2.2.1, Mealy machines are commonly used for the design of logical circuits and therefore the automatic generation of Mealy machines from given specifications is of particular interest. A logic which enables such a synthesis procedure is presented in [BRS08] and consists of a coalgebraic modal fragment introduced in Chapter 3 extended by the fixpoint operator ν. This extension suffices to formulate the behavior of a state via a finite formula, which additionally can be converted into a Mealy machine satisfying this specific formula. The transition system obtained through this process is not necessarily minimal and therefore bisimulation offers a useful technique to optimize the synthesis outcome [BRS08].

In order to exemplify the connection to our theory introduced in Chapter 3, we consider *sequence detection* as a concrete application, which plays a significant role in digital communication channels where bit sequences are used to identify the beginning and ending of messages [KRM17]. Another interesting scenario with pattern search involving Mealy machines with the input alphabet $\Sigma = \{T, A, G, C\}$ is used for the analysis of DNA sequences [MK16].

Since automatically generated and manually obtained models may not be minimal, our game-theoretical approach developed within T-Beg is not only suitable for recognizing all bisimilar state pairs; at the same time, T-Beg offers an intuitive explanation for non-bisimilar states.

Therefore, we design a simple sequence detector (extending Example 2.2.7) and apply our game to verify if our model is minimal (i.e., all distinct states are non-bisimilar).

The non-overlapping Mealy machine $M = (\{x_0, \dots, x_5\}, \{0, 1\}, \{0, 1\}, x_0, \delta)$ in Figure 4.2 outputs 1 in case it detects the pattern 101 or 000 (see Example 2.3.13 for the details of the coalgebraic view, where we also ignore the initial state). Non-overlapping simply means that only the first occurrence of the pattern 101 is identified

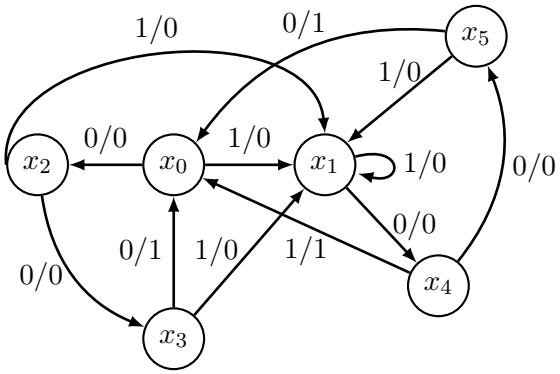


Figure 4.2: A simple sequence detector identifying 101 and 000 (with non-overlapping) and initial state x_0.

in 10101, where overlapping allows to reuse the last bit of a successful detection and an overlapping model would detect 101 two times.

To play our game we need to introduce the lifted order $\leq^F$ for $F = (\Sigma \times \_)^\Gamma$ with $\Sigma = \Gamma = \{0, 1\}$. Recall, that $t_1 \leq^F t_2$ with $t_1, t_2 \in (\{0, 1\} \times \{0, 1\})^{\{0,1\}}$ if there exists a $t \in (\{0, 1\} \times \{(0, 0), (0, 1), (1, 1)\})^{\{0,1\}}$ such that $F\pi_i(t) = t_i$ for $i \in \{1, 2\}$ and π_i are the usual projections $\pi_i : \{(0, 0), (0, 1), (1, 1)\} \to \{0, 1\}$.

The transitive partial order over $(\{0, 1\} \times \{0, 1\})^{\{0,1\}}$ indicated by Figure 4.3 shows that the output behaviour has to be equal i.e. $\pi'_1(t_1(i)) = \pi'_1(t_2(i))$ for each input $i \in \Sigma$ where $\pi'_1 : \{0, 1\} \times \{0, 1\} \to \{0, 1\}$ maps to the first component. In addition, we have $\pi'_2 : \{0, 1\} \times \{0, 1\} \to \{0, 1\}$ which maps to the second component and $\pi'_2(t_1(i)) \leq \pi'_2(t_2(i))$ has to hold. Therefore, one can easily prove that the lifted order is anti-symmetric.

$$
\begin{array}{ccc}
 & (\{0, 1\} \times \{(0, 0), (0, 1), (1, 1)\})^{\{0,1\}} & \\
F\pi_1 \swarrow & & \searrow F\pi_2 \\
(\{0, 1\} \times \{0, 1\})^{\{0,1\}} & \leq^F & (\{0, 1\} \times \{0, 1\})^{\{0,1\}}
\end{array}
$$

Figure 4.3: The preorder lifting yields a transitive order since the functor is weak-pullback preserving.

Corollary 4.2.1

The preorder lifting $\leq^F$ for $F = (\Sigma \times \_)^\Sigma$ yields an anti-symmetric partial order.

Proof: Reflexivity is given by the definition of the lifting [BK11] and transitivity is

implied by weak-pullback preservation of F [BK11]. We prove the anti-symmetry of $\leq^F$ via contradiction. Given $t_1, t_2 \in (\{0,1\} \times \{0,1\})^{\{0,1\}}$ with $t_1 \leq^F t_2$ and $t_2 \leq^F t_1$. Assume that $t_1 \neq t_2$ holds which implies that t_1, t_2 should differ in the second component since the lifting shown in Figure 4.3 requires that the output behaviour is equal for each input symbol. Therefore and by the fact that we just consider the output values $0, 1$, either $\pi_2'(t_1(i)) < \pi_2'(t_2(i))$ or $\pi_2'(t_2(i)) < \pi_2'(t_1(i))$ holds for some input $i \in \{0,1\}$. The proof works analogously for both cases and we consider the first case:

$$\pi_2'(t_1(i)) = 0 < 1 = \pi_2'(t_2(i))$$

which yields a contradiction to $t_2 \leq^F t_1$ since $(1,0) \notin \leq$. □

Therefore, we get *cone modalities* (see Section 3.4.3) which address the output behaviour and the successor state reached after consuming an input i. Intuitively, non-bisimilar states are distinguishable by their output behavior for at least one input word.

By Corollary 4.2.1 and Proposition 3.2.17 we are now ready to play the game and the initial pair is (x_3, x_5) where $\rightsquigarrow$ denotes the evaluation of $Fp(\alpha(x))$ for x, p where $p : X \to \{0,1\}$ is a predicate. We write $[(\_,\_),(\_,\_)]$ for an element $Fp(\alpha(x)) = t \in (\{0,1\} \times \{0,1\})^{\{0,1\}}$ where the first component of $[(\_,\_),(\_,\_)]$ corresponds to the input 0 and the second to the input 1.

The first component of $(\{0,1\} \times \{0,1\})$ denotes the output and the second one indicates if the successor state is captured by p.

Note, that besides the states x_0, x_1 no state has an impact on the value $Fp(\alpha(x_3))$ and the game proceeds as follows:

$$(x_3, x_5) \xrightarrow{\mathbf{S}} x_3, \{x_0, x_1\} \rightsquigarrow [(1,1),(0,1)], x_5$$

Next, starting from x_5, **D** has to provide at least the value $[(1,1),(0,1)]$ where the smallest set satisfying this is given by $\{x_0, x_1\}$:

$$x_5, [(1,1),(0,1)] \xrightarrow{\mathbf{D}} x_5, \{x_0, x_1\} \rightsquigarrow [(1,1),(0,1)]$$

Obviously, **D** can now choose the same states as **S** and therefore the game will never terminate, assuming that **D** follows her winning strategy implied by bisimilar state pairs.

Summarizing, due to the existence of a winning strategy for **D**, a merge of x_3, x_5 results in a simplification of the model in Figure 4.2.

In case two state are non-bisimilar, the winning strategy of **S** corresponds to an input sequence. Consider for example (x_0, x_1) which produce different outputs for the word 01. Due to Algorithm 3.1 the states x_2 and x_4 will be separated in the first iteration since they differ in their outputs for input 1. Based on that separation, x_0 and x_1 will be classified as non-bisimilar in the next iteration, since x_2 is the 0-successor of x_0 while x_1 has x_4 as 0-successor.

4.3 Conclusion

The didactical benefits of a tool such as T-Beg are discussed in the previous section and demonstrated via several examples in Chapter 3.

At this point, we want to emphasize that the coalgebraic framework facilitates the architecture design. The fact that our Algorithm 3.1 expects only two parameters, namely a functor F and a concrete instance of type $X \to FX$, already points to the encapsulation of the algorithm in its own class and a separate interface for the functors. Based on this observation, we used the namespace *Generic* of the .NET environment to develop a generic MVC. The modularity implied by such a generic approach also simplified the testing and debugging phase[4]. Optimizations of the polynomial runtime behaviour are already mentioned in Section 3.5 and we refer to the work presented in [WD+20].

The implementation costs arising on the user side can be improved by employing a separate module called *CoPaR* that automatically generates functors (see [DM+19]). But it is not clear whether the lifting of the preorder can be obtained automatically. Given two functors F, G such that $\leq^F, \leq^G$ are partial orders. According to the results for the neighborhood functor presented in Appendix A.1 the question arises whether the composition $F \circ G$ always implies that $\leq^{F \circ G}$ is a partial order too.

Furthermore, in [KKM17] we introduced Paws, a tool to analyse the behaviour of weighted automata and conditional transition systems. At its core, Paws is based on a generic implementation of the coalgebraic partition refinement algorithms for language equivalence in case of weighted automata and bisimulation for conditional transition systems [Küp17]. The architecture of Paws allows to use arbitrary user-defined semirings. New semirings can be generated during run-time and the user can rely on numerous automatisms to create new semiring structures for Paws.

Nevertheless, a combination of the work in [DM+19] (including *CoPaR*) and Paws [KKM17] featuring T-Beg would result in a powerful coalgebraic tool framework.

[4] A tutorial, small demo and testing report are available on the T-Beg website at `https://www.uni-due.de/theoinf/research/tools_tbeg.php`.

5

Behavioural Distances: Modal Logic and Games over Set

The second dimension of the triple bisimulation, logics and games is to move from a qualitative to a quantitative notion of behavioural equivalence and to provide logical and game-theoretical semantics in a quantitative manner (compare Figure 3.1 with Figure 5.1). That is, we refrain from classifying systems as either equivalent or non-equivalent, which is often too strict, but rather measure their behavioural distance. This makes sense in probabilistic systems, systems with time or real-valued output.

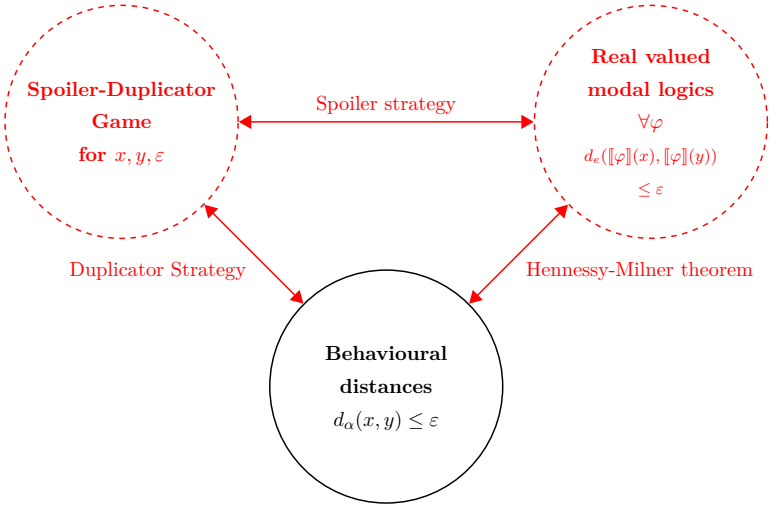


Figure 5.1: The quantitative coalgebraic triad of behavioural distance d_α, games, and modal logics for $\alpha : X \to FX$ over endofunctors $F : \mathbf{Set} \to \mathbf{Set}$, where F specifies the branching type of a system α. (The *red* coloring and the *dashed* circles indicate the open research questions.)

5.1 Introduction

In practice it is often only possible to get implementations that are (very) close to their specifications but not exactly the same. Therefore, it is necessary to move from

a qualitative to a quantitative notion of behavioural equivalence [DLT08; AFS09].

For instance, consider the response property in reactive systems, where systems are preferred that handle requests quickly and guarantee a large ratio of served to unserved requests [CHR10]. Given a specification s, which requires a response time of 5 *seconds* and two implementations i_1, i_2, we might obtain the result, that the running time of i_1 differs from the specification by 10 seconds, but i_2 differs by 20 seconds. Behavioural equivalence identifies (s, i_1) and (s, i_2) as non-equivalent, which is often too strict. Obviously, i_1 is closer to s compared to i_2 and therefore it is more appropriate to measure behavioural distance than working with behavioural equivalence.

Such an approach makes sense in probabilistic systems, systems with time or real-valued output. Therefore, quantitative notions are for instance useful in the area of conformance testing [KM15] and differential privacy [CG+14; CCP18; Dwo06]. On the other hand, two states are behaviourally equivalent in the classical sense if and only if they have distance 0.

Behavioural metrics have been studied in different variants, for instance in probabilistic settings [Des99; DG+04; CGT16] as well as in the setting of metric transition systems [AFS09; FLT11], which are non-deterministic transition systems with quantitative information. The groundwork for the treatment of coalgebras in metric spaces was laid by Turi and Rutten [TR98].

Due to the fact, that in the quantitative context the state space of an underlying system is equipped with a *(pseudo-)metric*, the work in [BB+14; BB+18] shows how to characterize behavioural metrics in coalgebras by studying various possibilities to lift functors from **Set** to the category of *(pseudo-)metric* spaces. Different from [TR98; BW05] we do not assume that the coalgebra is given a priori in the category of *pseudometric spaces*, that is we have to first choose a lifting of the behaviour functor in order to specify the behavioural metric. But such liftings are not unique as illustrated by the product bifunctor $F(X, Y) = X \times Y$ since there are several suitable liftings: we can e.g. use the maximum or the sum metric. While the maximum metric is canonically induced by the categorical product, the sum metric is also fairly natural.

In particular [BB+14; BB+18] introduces the *Kantorovich* and the *Wasserstein liftings*, which generalize well-known liftings for the probabilistic case and also capture the Hausdorff metric. Here we use the *Kantorovich lifting*, since this lifting integrates better with coalgebraic logic. Our results are parameterized over the lifting, in particular the behavioural metrics, the game and the logics are dependent on a set Γ

of evaluation functions.

In the metric setting it is natural to generalize from classical two-valued logics to real-valued modal logics and to state a corresponding Hennessy-Milner theorem that compares the behavioural distance of two states with the logical distance, i.e., the supremum of the differences of values, obtained by the evaluation of all formulas. Such a Hennessy-Milner theorem for probabilistic transition systems was shown in [DG+04] and also studied in a coalgebraic setting [BW05; BW06]. Similar results were obtained in [WS+18a] for fuzzy logics, on the way to proving a van Benthem theorem. Fuzzy logics were also studied in [SP11] in a general coalgebraic setting, but without stating a Hennessy-Milner theorem.

Again, work on games is scarce: [DLT08] presents a game (see Section 3.3.1) which characterizes behavioural distances, but pairs it with a classical logic. Games derived from a categorical framework are presented in [KK+19] where a fibrational approach is partly inspired by the work presented in this chapter. Moreover, the authors introduce a way that captures common bisimilarity notions and games, but do not incorporate modal logics.
The contributions in this chapter are [KM17a]:

- We extend the Kantorovich lifting in [BB+14; BB+18] to a set Γ of evaluation maps.
- We present a real-valued coalgebraic modal logic and give a Hennessy-Milner theorem for the general coalgebraic setting as a new contribution. Our proof strategy follows the one for the probabilistic case in [BW05]. We need several concepts from real analysis, such as *non-expansiveness* and *total boundedness* in order to show that the behavioural distance (characterized via a fixpoint) and the logical distance coincide.
- Furthermore we give a game characterization of this behavioural metric in a game where we aim to show that $d_\alpha(x, y) \leq \varepsilon$, i.e., the behavioural distance of two states x, y is bounded by ε.
- Furthermore, we work out the strategies for the duplicator and spoiler: while the strategy of the duplicator is based on the knowledge of the behavioural metric, the strategy of the spoiler can be derived from a logical formula that distinguishes both states. Therefore, we conclude by explaining how the strategy for the spoiler can be derived from a logical formula distinguishing two states.

The chapter is organized as follows: the development in the metric case is more complex, but in several respects mimics the classical case. Hence, in order to

emphasize the similarities, we will use the same structure as in Chapter 3. After this introduction (Section 5.1) we start with foundations (Section 5.2), followed by the introduction of modal logics and the proof of the Hennessy-Milner theorem in Section 5.3. In Section 5.4 we will introduce the game with a proof of its soundness and completeness. Finally we will show how the strategy for the spoiler can be derived from a logical formula. In the end we wrap everything up in the conclusion (Section 5.5).

5.2 Foundations

For the beginning we restrict to the same setting as in Chapter 3, where a system is given by a coalgebra $\alpha : X \to FX$ in **Set**. For further details regarding category theory we refer to Section 2.3.

Note that this chapter contains several results which are new with respect to [BB+18], in particular the extension of the Kantorovich lifting to several evaluation maps and Propositions 5.2.19, 5.2.20, 5.2.22, 5.2.23 and 5.2.26.

In the classical case discussed in Chapter 3, we consider behavioural equivalence characterized by relations over $X \times X$. Such relations can be interpreted as functions of type $X \times X \to \{0, 1\}$, where $(x, y) \mapsto 1$ means that states x, y have the same observable behaviour, and otherwise a state pair is recognized as non-equivalent.

To talk about distances, we consider functions of type $X \times X \to \mathbb{R}_0$ ($0 \in \mathbb{R}_0$) and assume that $\top$ is an element of $\mathbb{R}_0$, it denotes the upper bound of our distances. Dual to Chapter 3, $(x, y) \mapsto 0$ means that the distance of x and y is zero, i.e. they are behavioural equivalent. A value $v > 0$ indicates how similar the behaviour of x and y is, whereby a value closer to $\top$ signifies that two states behave very differently. Such functions are given by the standard notion of a pseudometric space.

Definition 5.2.1: Pseudometric, Pseudometric Space

Let X be a set and $d\colon X \times X \to [0, \top]$ a real-valued function, we call d a pseudometric if it satisfies

1. Reflexivity: $d(x, x) = 0$ (d is a metric if in addition $d(x, y) = 0$ implies $x = y$.)
2. Symmetry: $d(x, y) = d(y, x)$
3. Triangle inequality: $d(x, z) \leq d(x, y) + d(y, z)$

for all $x, y, z \in X$. If d satisfies only Condition 1 and 3, it is a directed

pseudometric.

A (directed) pseudometric space is a pair (X, d) where X is a set and d is a (directed) pseudometric on X.

Example 5.2.2

We will consider the following (directed) metrics on $[0, \top]$: the metric given by the Euclidean distance $d_e : [0, \top] \times [0, \top] \to [0, \top]$ with $d_e(a, b) = |a - b|$ and the directed metric defined by truncated subtraction with $d_\ominus(a, b) = a \ominus b = \max\{a - b, 0\}$. Note that $d_e(a, b) = \max\{d_\ominus(a, b), d_\ominus(b, a)\}$.

Maps between pseudometric spaces are given by non-expansive functions, which guarantee that mapping two elements either preserves or decreases their distance. Functions that decrease and do not increase the distances play an important role in *transportation theory* [Vil09], but the intuition can also be transferred to system behaviour analysis [BB+18; Ker16].

Definition 5.2.3: Non-expansive Function

Let (X, d_X), (Y, d_Y) be pseudometric spaces. A function $f\colon X \to Y$ is called non-expansive if $d_X(x, y) \geq d_Y(f(x), f(y))$ for all $x, y \in X$. In this case we write $f\colon (X, d_X) \xrightarrow{1} (Y, d_Y)$.

On some occasions we need to transform an arbitrary function into a non-expansive function, which can be done as follows.

Lemma 5.2.4

Let d be a pseudometric on X and let $f\colon X \to [0, \top]$ be any function. Then we define a non-expansive function $h\colon (X, d) \to ([0, \top], d_e)$ via $h(z) = \sup\{f(u) - d(u, z) \mid u \in X\}$ which satisfies $f \leq h$.

Analogously we define the function $g\colon (X, d) \to ([0, \top], d_e)$ via $g(z) = \inf\{f(u) + d(u, z) \mid u \in X\}$ which is non-expansive and satisfies $g \leq f$.

Proof: The first obvious fact is that, $f \leq h$ holds, since $f(z) = f(z) - d(z, z) \leq \sup\{f(u) - d(u, z) \mid u \in X\} = h(z)$.

Next, we show that h is non-expansive. Let $z, z' \in X$. Due to the definition of h there exists for all $\delta > 0$ a $u \in X$ with $h(z) \leq f(u) - d(u, z) + \delta$ and $f(u) - d(u, z') \leq h(z')$. Combined, we have

$$h(z) - h(z') \leq (f(u) - d(u, z) + \delta) - f(u) + d(u, z') = d(u, z') - d(u, z) + \delta.$$

Since this holds for every $\delta > 0$ we have $h(z) - h(z') \leq d(u, z') - d(u, z)$. Due to the triangle inequality $d(u, z') \leq d(u, z) + d(z, z')$, hence

$$h(z) - h(z') \leq d(u, z') - d(u, z) \leq d(z, z').$$

Analogously, we can show $h(z') - h(z) \leq d(z', z) = d(z', z)$ (due to symmetry), hence $d_e(h(z), h(z')) \leq d(z, z')$, which means that h is non-expansive.

For the function g the proof is analogous. □

Next, we want to explain, how the lifting of the behaviour functor enables the construction of a pseudometric $d^{\uparrow}$ over the set FX from a behavioural pseudometric d over the state space X for a given coalgebra $\alpha : X \to FX$.

To better understand the categorical view on such lifting techniques presented in [BB+14; BB+18] and supplemented by the work in this thesis, we first explore this concept on two concrete examples. Moreover, these two transition systems serve as our *running* motivation for *quantitative system analysis*. Therefore, the next subsection introduces the background information of two different branching-types: metric transition systems (MTS) and probabilistic systems (PB).

5.2.1 Two Quantitative Models

There are many *quantitative* application scenarios when it comes to the development and implementation of software systems. Software itself exhibits a rather discrete behaviour based on events, inputs or outputs. In contrast to this, several processes in the environment show continuous behaviour. It is not uncommon that components with continuous behaviour interact with discrete systems. For instance embedded systems are present everywhere: in electrical, mechanical or chemical systems [CRH02]. Such systems are called *hybrid systems* or *cyber-physical* systems [KM15].

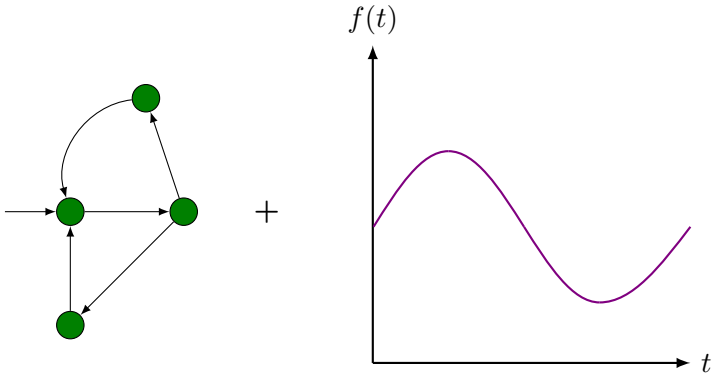


Figure 5.2: Combining discrete systems with continuous systems [CRH02].

Moreover, quantitative systems often deal with *costs*, which are in the context of software systems energy costs, processor costs (power and time) or memory costs [BF16; BHR14; AFS04]. Such phenomena (costs, rewards, or other observations with respect to hybrid systems) are captured by a set of *continuous variables*, where a *continuous variable* can take on uncountably many values.

In the literature one finds a lot of theoretical models that are expressive enough to capture continuous behaviour. For the interested reader we refer to (priced) timed automata (PTA) [BF16] or hybrid labelled transition systems (HLTS) [KM15]. Moreover, one should notice, that some of the models enable discrete-time samplings of continuous systems, where so-called *propositions* are real-valued discrete snapshots of continuous variables [AFS04].

There is no doubt that verifying such systems is of particular interest. One technique is known under the term of *conformance-testing*, where test cases are generated from models to verify how good an implementation approximates a specification [KM15]. As already indicated, there are several ways to model hybrid systems, and by the work in [KM15] we know, that some of them (e.g. HLTS) can be transformed into metric transition systems (MTS).

At this point, we want to introduce the notion of MTS presented in [AFS04], where we omit a set of labels compared to the version given in [KM15]. MTS offer a rather high level of abstraction and therefore the following model mainly serves as an illustrative example. A scenario of a real world application modelled via an MTS can be found in [GZ12].

We denote with $\Sigma = \{r_1, \cdots, r_n\}$ a finite set of propositions and each $r \in \Sigma$ is associated with a pseudometric space (M_r, d_r). Besides, we must have a finite $\top \in]0, \infty[$ such that $d_r : M_r^2 \to [0, \top]$ for all $r \in \Sigma$. A *valuation* of Σ is a function $v : \Sigma \to \bigcup_{r \in \Sigma} M_r$. For each proposition $r \in \Sigma$ the observation $v(r)$ is given by a value in M_r. We denote the set of all these valuations by $\mathcal{U}[\Sigma]$ [AFS09].

Definition 5.2.5: Metric Transitions System (MTS)

A metric transition system is a quadruple $M = (S, \tau, \Sigma, [\cdot])$, where S is a set of states, $\tau \subseteq S \times S$ is a transition relation, and Σ is a finite set of propositions. Besides, every state s is assigned a valuation $[s] \in \mathcal{U}[\Sigma]$ given by the function $[\cdot] : S \to \mathcal{U}[\Sigma]$. We define $\tau(s) := \{s' \in S \mid (s, s') \in \tau\}$.

Now, one can compute the distance $d(s, t)$ between two states $s, t \in S$ based on the so-called *Hausdorff-Distance* [AFS09].

Definition 5.2.6: Hausdorff Metric

We lift a metric space (X, d) to $(\mathcal{P}_f X, d')$: for $X_1, X_2 \subseteq X$ as follows:

$$d^H(X_1, X_2) = \max\{\max_{x \in X_1} \min_{y \in X_2} d(x, y), \max_{y \in X_2} \min_{x \in X_1} d(y, x)\}$$

It has already been observed that the Hausdorff metric is a monotone lifting and additionally the space of pseudometrics forms a complete lattice [BB+18]. Therefore and based on the Knaster-Tarski theorem, the distance $d : S \times S \to X$ between two states in a MTS is obtained via the least fixpoint of the following fixpoint equation

$$d(x, y) = \max\{pd([x], [y]), d^H(\tau(x), \tau(y))\}$$

where $pd([x], [y]) = |[x] - [y]|$ for $|\Sigma| = 1$. For the general case $|\Sigma| > 1$ we get the following *propositional distance* $pd : \mathcal{U}[\Sigma]^2 \to [0, \top]$ for all valuations $u, v \in \mathcal{U}[\Sigma]^2$ [AFS09, Definition 10]:

$$pd(u, v) = \max_{r \in \Sigma} d_r(u(r), v(r))$$

Next, given the sets of successors X_1, X_2 respectively of x and y, the distance is derived in such a way that for each element $s_x \in X_1$ and $s_y \in X_2$ take the closest element $y' \in X_2$ and $x' \in X_1$ and measure the distances $d(s_x, y')$, $d(s_y, x')$. Finally, take the maximum of all such distances.

Example 5.2.7

Figure 5.3 shows a metric system $M = (S, \tau, \Sigma, [\cdot])$ where Σ is a singleton representing some quantitative proposition. We explain the calculation of the distance $d(s, t)$ between the states $s, t \in S$.

Figure 5.3: Each of the states is equipped with a value that represents a quantitative observation [AFS09].

Therefore, we consider the sets of successor states $X_1 = \tau(s) = \{3, 4\}$ and $X_2 = \tau(t) = \{5, 6\}$. Note, that all these elements have no successor states and are distinguished only by their values $[\cdot]$.
According to Definition 5.2.6 we take for each element $x \in X_1$ the closest

element $y \in X_2$ and determinine the maximum. For an $x \in X_1$ the closest element is computed based on the distance $d(x, y)$ for each $y \in X_2$, where $d(x, y) = \max\{d_r([x], [y]), d^H(\tau(x), \tau(y))\}$.
Analogously, we take for each element $y \in X_2$ the closest element $x \in X_1$ and again determinine the maximum. Finally, we take the maximum of the two maxima.
Thus, we get $d(s, t) = max\{d_r(0, 0), d^H(\{3, 4\}, \{5, 6\})\} = \max\{0, 0.3\} = 0.3$ with $d^H(\{3, 4\}, \{5, 6\}) = \max\{\max\{0.2, 0.1\}, \max\{0.1, 0.3\}\} = \max\{0.2, 0.3\}$.

Next, we motivate a quantitative view on probabilistic systems (see Definition 2.2.4) compared to probabilistic bisimulation (see Definition 2.2.10). Naturally, probabilistic branching enables a more appropriate analysis of system behaviour. Therefore we proceed with probabilistic systems already introduced in Section 2, but for simplification we restrict to systems without labels.

Here, we want to underline that bisimulation is an extremely strict concept. Especially when you consider that a transition can fail as it happens in server based applications, which handle an enormous number of requests daily [CHR10].

To integrate these specific problems into the design of an application, a specification requires that a system handles such exceptions with a probability of 100%. Now, a *valid* implementation behaves in a stable way in 98% of the exceptions. Obviously, bisimulation classifies an *unsuitable* solution (handling only 70% of the exceptions) in the same way as the first version.

Example 5.2.8

Assume, that the systems given in Figure 5.4 represent a subsystem in some larger system. Each transition is equipped with a probability modelling the exception handling rate.
The states E_s, E_1, E_2 are reached if an exception occurs. The states F_1, F_2 represent the failure states, reached in case an exception is not handled. Otherwise the states H_s, H_1, H_2 indicate that an exception is handled by the system.

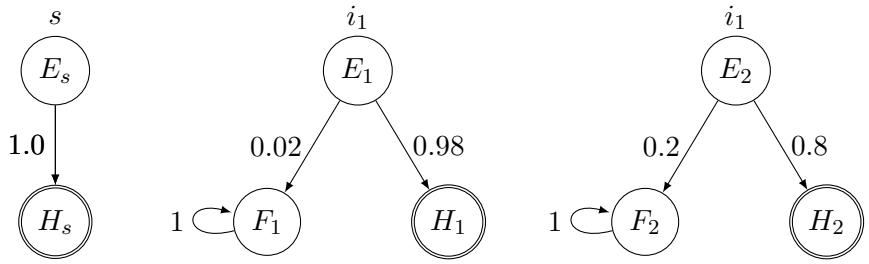


Figure 5.4: A probabilistic system with termination, where $d(s, i_1) = 0.02 \neq 0$ and $d(s, i_2) = 0.20 \neq 0$ (cf. [Hop20]).

In general measuring the distance of two states in a probabilistic system instead of expecting the states to admit equivalent behaviour seems to be more adequate.

Example 5.2.9

Each transition in Figure 5.5 is equipped with a probability to provide quantitative transition information.
The tolerance threshold for the behaviour of the state pair $(1, 2)$ is described by $\varepsilon \in [0, 1]$. Assume, that ε is a tiny value, then state 2 has a slightly greater probability to end up in a terminal state than state 1.

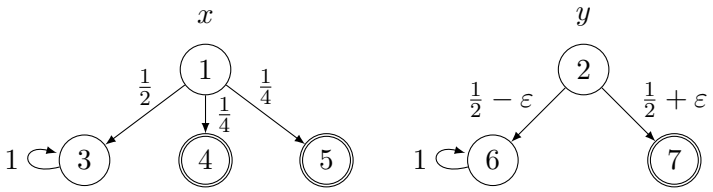


Figure 5.5: A probabilistic system with termination, where $d(x, y) = \varepsilon$ (inspired by [BW06]).

As shown in Example 5.2.9 the distance of two states depends on the probabilities of the transitions. Therefore, we proceed with the computation of the distance between two probability distributions p_x, p_y on a metric space to obtain the distance between two states in a probabilistic system (i.e. lifting the distance to probability distributions). Note, that $\top = 1$ determines the distance between a terminating state and a non-terminating state.

Definition 5.2.10: Probabilistic Distance[BW06]

Compute the smallest fixed-point of

$$d(x,y) = \begin{cases} 1 & \text{if } x \in T, y \notin T \text{ or } x \notin T, y \in T \\ 0 & \text{if } x \in T, y \in T \\ d^P(p_x, p_y) & \text{otherwise} \end{cases}$$

The distance d^P between two probability distributions is given by the so-called Kantorovich lifting [Ver06; Vil09]:

Definition 5.2.11: Kantorovich Lifting for Probability Distributions

Let $p, q : X \to [0,1]$ be two probability distributions and $d : X \times X \to [0,1]$ a metric. We lift d to the space of probability distributions on X as follows:
$d^{\uparrow}(p,q) = \sup\{|\sum_{x\in X} f(x)\cdot p(x) - \sum_{x\in X} f(x)\cdot q(x)| \mid \ f\colon (X,d) \xrightarrow{1} ([0,1], d_e)\}$

To summarize this section, two different models and their corresponding methods to lift a metric have been discussed. The next section presents a generalized view on how to lift metrics taken from [BB+18; Ker16].

5.2.2 Behavioural Distance Coalgebraically

From the previous section it can be deduced, that given a transition system $\alpha : X \to FX \in \mathbf{Set}$, the behavioural distance $d : X \times X \to [0,1]$ over the state space X strongly depends on the distance between $\alpha(x)$ and $\alpha(y)$ for two given states x, y. Thus, given an endofunctor F on **Set**, our goal is to define a pseudometric over X based on a pseudometric over FX.

Therefore, the next pages introduce a coalgebraic view on lifting techniques for metrics and most of the definitions and results are taken from [BB+18; Ker16].

Since our objects of interest are pseudometric spaces, we start by introducing a suitable category:

Definition 5.2.12: Category of Pseudometric Spaces

Let $\top \in]0, \infty]$ be a fixed maximal element. The category **PMet** has as objects all pseudometric spaces whose pseudometrics have codomain $[0, \top]$. The arrows are the non-expansive functions between these spaces. The identities are the (isometric) identity functions and composition of arrows is function composition.

We will now define the Kantorovich lifting for **Set**-functors, introduced in [BB+14;

Ker16]. Given a functor F we lift it to a functor $\bar{F}\colon \mathbf{PMet} \to \mathbf{PMet}$ such that $UF = \bar{F}U$, where $U : \mathbf{PMet} \to \mathbf{Set}$ is the forgetful functor, discarding the pseudometric. The Kantorovich lifting is parameterized over a finite set Γ of evaluation maps $\gamma\colon F[0,\top] \to [0,\top]$, the analogue to the evaluation maps for modalities in the classical case. This is an extension of the lifting in which only a single evaluation map is considered [BB+14]. The new version allows to generalize the liftings introduced earlier and to capture additional examples, without going via the somewhat cumbersome multifunctor lifting described in [BB+14].

Definition 5.2.13: Kantorovich Lifting

Let F be an endofunctor on **Set** and let Γ be a finite set of evaluation maps $\gamma\colon F[0,\top] \to [0,\top]$. For every pseudometric space (X,d) the Kantorovich pseudometric on FX is the function $d^{\uparrow\Gamma}\colon FX \times FX \to [0,\top]$, where for $t_1, t_2 \in FX$:

$$d^{\uparrow\Gamma}(t_1,t_2) := \sup\{d_e(\gamma(Ff(t_1)),\gamma(Ff(t_2))) \mid f\colon (X,d) \xrightarrow{1} ([0,\top],d_e), \gamma \in \Gamma\}$$

We define $\bar{F}_\Gamma(X,d) = (FX, d^{\uparrow\Gamma})$ on objects, while $\bar{F}_\Gamma$ is the identity on arrows.

We will abbreviate $\tilde{F}_\gamma f = \gamma \circ Ff$. Note that $\tilde{F}_\gamma$ is a functor on the slice category $\mathbf{Set}/[0,\top]$, which lifts real-valued predicates $p\colon X \to [0,\top]$ to real-valued predicates $\tilde{F}_\gamma p\colon FX \to [0,\top]$.

It still has to be shown that $\bar{F}$ is well-defined. The proofs are a straighforward adaptation of the proofs in [BB+14].

Lemma 5.2.14

The Kantorovich lifting for pseudometrics (Definition 5.2.13) is well-defined, in particular it preserves pseudometrics and maps non-expansive functions to non-expansive functions.

Proof: $\bar{F}$ preserves identities and composition of arrows, since F does. Hence we only have to show the following.

Preservation of pseudometrics: we show that reflexivity and triangle inequality are preserved. Assume that $t, t_1, t_2, t_3 \in FX$ and let $d\colon X \times X \to [0,\top]$ be a pseudometric.

- *Reflexivity* holds since d_e is reflexive:

$$d^{\uparrow\Gamma}(t,t) = \sup\{d_e(\tilde{F}_\gamma f(t), \tilde{F}_\gamma f(t)) \mid f : (X,d) \xrightarrow{1} ([0,\top],d_e), \gamma \in \Gamma\} = 0$$

- *Symmetry* holds since d_e is symmetric:

$$\begin{aligned} d^{\uparrow\Gamma}(t_1,t_2) &= \sup\{d_e(\tilde{F}_\gamma f(t_1),\tilde{F}_\gamma f(t_2)) \mid f:(X,d)\xrightarrow{1}([0,\top],d_e),\gamma\in\Gamma\} \\ &= \sup\{d_e(\tilde{F}_\gamma f(t_2),\tilde{F}_\gamma f(t_1)) \mid f:(X,d)\xrightarrow{1}([0,\top],d_e),\gamma\in\Gamma\} \\ &= d^{\uparrow\Gamma}(t_2,t_1) \end{aligned}$$

- *Triangle inequality:* since d_e is a pseudometric it satisfies the triangle inequality, in particular:

$$d_e(\tilde{F}_\gamma f(t_1),\tilde{F}_\gamma f(t_3)) \leq d_e(\tilde{F}_\gamma f(t_1),\tilde{F}_\gamma f(t_2)) + d_e(\tilde{F}_\gamma f(t_2),\tilde{F}_\gamma f(t_3))$$

  for all $f\colon (X,d)\xrightarrow{1}([0,\top],d_e)$. Therefore we take the supremum on both sides and obtain

$$\begin{aligned} d^{\uparrow\Gamma}(t_1,t_3) &= \sup_{f,\gamma} d_e(\tilde{F}_\gamma f(t_1),\tilde{F}_\gamma f(t_3)) \\ &\leq \sup_{f,\gamma} \big(d_e(\tilde{F}_\gamma f(t_1),\tilde{F}_\gamma f(t_2)) + d_e(\tilde{F}_\gamma f(t_2),\tilde{F}_\gamma f(t_3))\big) \\ &\leq \sup_{f,\gamma} d_e(\tilde{F}_\gamma f(t_1),\tilde{F}_\gamma f(t_2)) + \sup_{f,\gamma} d_e(\tilde{F}_\gamma f(t_2),\tilde{F}_\gamma f(t_3)) \\ &= d^{\uparrow\Gamma}(t_1,t_2) + d^{\uparrow\Gamma}(t_2,t_3). \end{aligned}$$

Non-expansive functions: $\bar{F}_\gamma$ preserves non-expansive functions:
Let $f\colon (X,d_X)\xrightarrow{1}(Y,d_Y)$ be non-expansive and $t_1,t_2\in FX$, then

$$\begin{aligned} d_Y^{\uparrow\Gamma}(Ff(t_1),Ff(t_2)) &= \sup_{\gamma\in\Gamma} \sup_{g\colon (Y,d_Y)\xrightarrow{1}([0,\top],d_e)} d_e\Big(\tilde{F}_\gamma(g\circ f)(t_1),\tilde{F}_\gamma(g\circ f)(t_2)\Big) \\ &\leq \sup_{\gamma\in\Gamma} \sup_{h\colon (X,d_X)\xrightarrow{1}([0,\top],d_e)} d_e\Big(\tilde{F}_\gamma(h)(t_1),\tilde{F}_\gamma(h)(t_2)\Big) \\ &= d_X^{\uparrow\Gamma}(t_1,t_2) \end{aligned}$$

due to the fact that since both f and g are non-expansive, also their composition $(g\circ f)\colon (X,d_X)\xrightarrow{1}([0,\top],d_e)$ is non-expansive.

□

Example 5.2.15

In this example we want to study the difference between the extended Kantorovich lifting $d^{\uparrow\Gamma}$ in Definition 5.2.13 with the parametrization over Γ and the version $d^{\uparrow F}$ based on a single evaluation map introduced in [BB+14].
Therefore, we consider the *input functor* $I = \_^A$ for a finite set A. This functor

maps a set X to X^A (i.e. the set of all functions $A \to X$) and a function $f : X \to Y$ to $f^A : X^A \to Y^A$ with $f^A(g) = f \circ g$.
Furthermore, we consider the following metric $d_m : X^A \times X^A \to [0,1]$ where the distance of two functions $g, h \in X^A$ is measured as follows:

$$d_m(g,h) = \max_{a \in A} d(g(a), h(a))$$

for a given pseudometric space (X,d) with $d : X \times X \to [0,1]$.
In [Ker16] it is shown, that d_m can not be characterized by the Kantorovich lifting $d^{\uparrow F}$. Here we show, that the Kantorovich lifing parametrized over Γ enables this. Therefore, we define $\Gamma = \{\gamma_a \mid a \in A\}$ with $\gamma_a : [0,1]^A \to [0,1]$ and $\gamma_a(g) = g(a)$ where $g \in [0,1]^A$.
Given $g \neq h$ (with $g, h \in X^A$) there must exist at least one $a \in A$ such that $g(a) \neq h(a)$. Assume, that the distance $d_m(g,h)$ is based on some $a_i \in A$ such that $d_m(g,h) = d(g(a_i), h(a_i))$ with $x_i = g(a_i) \neq y_i = h(a_i)$ and $d(g(a), h(a)) \leq d(g(a_i), h(a_i))$ for all $a \in A$ and $d(g(a_i), h(a_i)) > 0$. (The case where $d_m(g,h) = 0$ is trivial.)
Next, we need to show, that there exists a non-expansive function $(X,d) \to ([0,1], d_e)$ witnessing that $d^{\uparrow\Gamma}(g,h) = d(x_i, y_i)$. Note, that a non-expansive function which yields $|\gamma_a \circ f^A(g) - \gamma_a \circ f^A(h)| > d(x_i, y_i)$ for some $\gamma_a \in \Gamma$ can not exist:
Assume there is some function $u : X \to [0,1]$ such that for some $\gamma_a \in \Gamma$ we have that $|u(g(a)) - u(h(a))| = |u(x_j) - u(y_j)| > d(x_i, y_i)$ where $g(a) = x_j, h(a) = y_j$ holds. But, this is a contradiction to

$$|u(x_j) - u(y_j)| \leq d(x_j, y_j) \leq d(x_i, y_i) = \max_{a \in A} d(g(a), h(a))$$

We define a function $f : X \to [0,1]$ as a witness for $d^{\uparrow\Gamma}(g,h) = d(x_i, y_i)$ as follows:

$$f(z) = d(z, y_i)$$

First of all, we show that f yields $|\gamma_{a_i}(Ff(g)) - \gamma_{a_i}(Ff(h))| = d(x_i, y_i)$:

$$\begin{aligned} \gamma_{a_i}(Ff(g)) = f(g(a_i)) = f(x_i) &= d(x_i, y_i) \\ \gamma_{a_i}(Ff(h)) = f(h(a_i)) = f(y_i) &= 0 \end{aligned}$$

That f is non-expansive can be derived from [BB+18, Lemma 3.9].

As a sanity check we observe that all evaluation maps $\gamma \in \Gamma$ are non-expansive for the Kantorovich lifting of d_e. In fact, the Kantorovich lifting is the least lifting that makes the evaluation maps non-expansive. This also means that a non-expansive function $f\colon (X,d) \xrightarrow{1} ([0,1],d_e)$ is always mapped to a non-expansive $\tilde{F}_\gamma f\colon (FX, d^{\uparrow\Gamma}) \xrightarrow{1} ([0,1],d_e)$.

Lemma 5.2.16

Let Γ be a set of evaluation maps. Every evaluation map $\gamma \in \Gamma$ is non-expansive for the Kantorovich lifting, i.e. $\gamma\colon (F[0,\top], d_e^{\uparrow\Gamma}) \xrightarrow{1} ([0,\top], d_e)$.

Proof: We have

$$d_e^{\uparrow\Gamma}(t_1,t_2) = \sup\{d_e(\tilde{F}_\gamma f(t_1), \tilde{F}_\gamma f(t_2)) \mid f\colon (X,d) \xrightarrow{1} ([0,\top],d_e), \gamma \in \Gamma\}$$

and we substitute the identity function id for f. Thus we obtain the following inequality $d_e^{\uparrow\Gamma}(t_1,t_2) \geq d_e(\tilde{F}_\gamma id(t_1), \tilde{F}_\gamma id(t_2)) = d_e(\gamma(t_1),\gamma(t_2))$, because $\tilde{F}_\gamma id = \gamma \circ Fid = \gamma$. □

For the following definitions we need the supremum metric on functions, since we will be working with functions between pseudometric spaces more often. Especially, we need a way to relate the distance between the lifted versions of two pseudometrics with the distance of the original two pseudometrics.

Definition 5.2.17: Supremum Metric

Let (Y,d) be a pseudometric space. Then the set of all functions $f\colon X \to Y$ is equipped with a pseudometric d^∞, the supremum metric, defined as

$$d^\infty(f,g) = \sup_{x\in X} d(f(x),g(x)) \text{ for } f,g\colon X \to Y.$$

We consider the following restrictions on evaluation maps respectively predicate liftings, which are needed in order to prove the results.

Definition 5.2.18: Properties of Evaluation Maps

Let $\gamma\colon F[0,\top] \to [0,\top]$ be an evaluation map for a functor $F\colon Set \to Set$.

- The predicate lifting $\tilde{F}_\gamma$ induced by γ is non-expansive with respect to the supremum metric whenever $d_e^\infty(\tilde{F}_\gamma f, \tilde{F}_\gamma g) \leq d_e^\infty(f,g)$ for all $f,g\colon X \to [0,\top]$ and the same holds if we replace d_e by $d_\ominus$.
- The predicate lifting $\tilde{F}_\gamma$ is contractive with respect to the supremum metric whenever $d_e^\infty(\tilde{F}_\gamma f, \tilde{F}_\gamma g) \leq c \cdot d_e^\infty(f,g)$ for some c with $0 < c < 1$.

- The predicate lifting $\tilde{F}_\gamma$ is ω-continuous, whenever for an ascending chain of functions f_i (with $f_i \leq f_{i+1}$) we have that $\tilde{F}_\gamma(\sup_{i<\omega} f_i) = \sup_{i<\omega}(\tilde{F}_\gamma f_i)$.

The first property results from the fact, that the distance $d_e^\infty(f,g)$ between two functions $f, g : X \to [0, \top]$ given by the supremum metric serves as an upper bound for the distance of the lifted functions $d_e^\infty(\tilde{F}_\gamma f, \tilde{F}_\gamma g)$. Behind this lies the idea, that the evaluation maps can not increase the distance after lifting.

The second item includes only the contractive version of an upper bound with respect to the supremum metric. A non-expansive map is just a special version of a contractive map, where the so-called *Lipschitz constant c* is set to 1.

The third requirement is simply, that the supremum is preserved by $\tilde{F}$. Such restrictions with respect to the supremum are not unusual in the field of computer science or as soon as it comes to approximation. (A similar property is known under the term *Scott-continuity* from *domain theory*.)

It can be shown that the first property is equivalent to a property of the lifted functor, called local non-expansiveness, studied in [TR98]. Besides, every locally non-expansive functor on the category of complete metric spaces can be transformed into a locally contractive one and such endofunctors have a unique fixpoint [TR98, Theorem 7.2].

Proposition 5.2.19: Local Non-expansiveness

Let Γ be a set of evaluation maps and let $\bar{F}$ be the Kantorovich lifting of a functor F via Γ. It holds that

$$(d_Y^F)^\infty(\bar{F}f, \bar{F}g) \leq (d_Y)^\infty(f,g)$$

for all non-expansive functions $f, g\colon (X, d_X) \to (Y, d_Y)$ (where $\bar{F}(Y, d_Y) = (FY, d_Y^F)$) if and only if

$$d_e^\infty(\tilde{F}_\gamma f, \tilde{F}_\gamma g) \leq d_e^\infty(f,g)$$

for all non-expansive functions $f, g\colon (X, d_X) \to ([0, \top], d_e)$ and all $\gamma \in \Gamma$.

Proof:

"⇒" Let $t \in FX$. By choosing the non-expansive identity function in the Kantorovich lifting we obtain $d_e^\infty(\tilde{F}_\gamma f(t), \tilde{F}_\gamma g(t)) = d_e^\infty(\tilde{F}_\gamma id(Ff(t)), \tilde{F}_\gamma id(Fg(t))) \leq d_e^{\uparrow\Gamma}(Ff(t), Fg(t)) \leq d_e^\infty(f,g)$. Note that in this setting $d_Y = d_e$ and $d_Y^F = d_e^{\uparrow\Gamma}$.

"⇐" Let again $t \in FX$.

$$\begin{aligned}
& (d_Y^F)^\infty(\bar{F}f(t), \bar{F}g(t)) \\
= \; & (d_Y)^{\uparrow\Gamma}(Ff(t), Fg(t)) \\
= \; & \sup\{d_e(\tilde{F}_\gamma h(Ff(t)), \tilde{F}_\gamma h(Fg(t))) \mid h\colon (Y, d_Y) \xrightarrow{1} ([0,\top], d_e), \gamma \in \Gamma\} \\
= \; & \sup\{d_e(\tilde{F}_\gamma(h \circ f)(t), \tilde{F}_\gamma(h \circ g)(t)) \mid h\colon (Y, d_Y) \xrightarrow{1} ([0,\top], d_e), \gamma \in \Gamma\}
\end{aligned}$$

We know that $d_e(\tilde{F}_\gamma(h \circ f)(t), \tilde{F}_\gamma(h \circ g)(t)) \leq d_e^\infty(h \circ f, h \circ g)$ and it is left to show that $d_e^\infty(h \circ f, h \circ g) \leq (d_Y)^\infty(f, g)$ for every such h. So let $x \in X$ and we obtain $d_e(h(f(x)), h(g(x))) \leq d_Y(f(x), g(x))$ since h is non-expansive. Taking the supremum on both sides we obtain the desired result. □

Assumption: *In the following we will always assume the first property in Definition 5.2.18 for every evaluation map γ, i.e., the predicate lifting $\tilde{F}_\gamma$ is non-expansive with respect to the supremum metric.*

Under this assumption it can be shown that the Kantorovich lifting itself is non-expansive (respectively contractive). This time the distance of two pseudometrics given by the supremum metric serves as an upper bound for the distance between the *Kantorovich-lifted* pseudometrics.

Proposition 5.2.20

Let Γ be a set of evaluation maps and let $d_1, d_2\colon X \times X \to [0,\top]$ be two pseudometrics. Then $d_e^\infty(d_1^{\uparrow\Gamma}, d_2^{\uparrow\Gamma}) \leq d_e^\infty(d_1, d_2)$, that is, the Kantorovich lifting of metrics is non-expansive for the supremum metric.
If, in addition, every predicate lifting $\tilde{F}_\gamma$ for $\gamma \in \Gamma$ is contractive (cf. Definition 5.2.18), we have that $d_e^\infty(d_1^{\uparrow\Gamma}, d_2^{\uparrow\Gamma}) \leq c \cdot d_e^\infty(d_1, d_2)$ for some c with $0 < c < 1$, that is, the Kantorovich lifting of metrics is contractive.

Proof: We set $\delta = d_e^\infty(d_1, d_2)$. Let f be a function which is non-expansive for d_1 and d_e, i.e., $f\colon (X, d_1) \xrightarrow{1} ([0,\top], d_e)$. We define another function $h\colon X \to [0,\top]$ via $h(z) = \sup\{f(u) - d_2(u, z) \mid u \in X\}$ as in Lemma 5.2.4. We know that $f \leq h$ and

$f\colon (X,d_2) \xrightarrow{1} ([0,\top],d_e)$. Now define $g = h - \frac{\delta}{2}$. We have for every $z \in X$:

$$\begin{aligned}
f(z) - g(z) &= f(z) - h(z) + \frac{\delta}{2} \leq \frac{\delta}{2} \\
g(z) - f(z) &= h(z) - \frac{\delta}{2} - f(z) = \sup\{f(u) - d_2(u,z) \mid u \in X\} - \frac{\delta}{2} - f(z) \\
&= \sup\{f(u) - f(z) - d_2(u,z) \mid u \in X\} - \frac{\delta}{2} \\
&\leq \sup\{d_1(u,z) - d_2(u,z) \mid u \in X\} - \frac{\delta}{2} \\
&\leq \delta - \frac{\delta}{2} = \frac{\delta}{2}
\end{aligned}$$

Hence $d_e^\infty(f,g) \leq \frac{\delta}{2}$. Non-expansiveness of the predicate lifting with respect to the supremum metric (cf. Definition 5.2.18) and $g < h$ implies $d_e(\tilde{F}_\gamma f(t), \tilde{F}_\gamma h(t)) \leq \frac{\delta}{2}$ for all $t \in FX$.

Given $t_1, t_2 \in FX$ we can infer with the triangle inequality:

$$\begin{aligned}
& d_e(\tilde{F}_\gamma f(t_1), \tilde{F}_\gamma f(t_2)) \\
\leq\ & d_e(\tilde{F}_\gamma f(t_1), \tilde{F}_\gamma h(t_1)) + d_e(\tilde{F}_\gamma h(t_1), \tilde{F}_\gamma h(t_2)) + d_e(\tilde{F}_\gamma h(t_2), \tilde{F}_\gamma f(t_2)) \\
\leq\ & d_e(\tilde{F}_\gamma h(t_1), \tilde{F}_\gamma h(t_2)) + \delta
\end{aligned}$$

Finally:

$$\begin{aligned}
& d_1^{\uparrow\Gamma}(t_1,t_2) \\
=\ & \sup\{d_e(\tilde{F}_\gamma f(t_1), \tilde{F}_\gamma f(t_2)) \mid f\colon (X,d_1) \xrightarrow{1} ([0,\top],d_e), \gamma \in \Gamma\} \\
=\ & \sup\{d_e(\tilde{F}_\gamma f(t_1), \tilde{F}_\gamma f(t_2)) \mid f\colon (X,d_1) \xrightarrow{1} ([0,\top],d_e), \gamma \in \Gamma, \\
& \qquad d_e^\infty(f,g) \leq \frac{\delta}{2} \text{ for some } g\colon (X,d_2) \xrightarrow{1} ([0,\top],d_e)\} \\
\leq\ & \sup\{d_e(\tilde{F}_\gamma g(t_1), \tilde{F}_\gamma g(t_2)) + \delta \mid g\colon (X,d_2) \xrightarrow{1} ([0,\top],d_e), \gamma \in \Gamma\} \\
=\ & \sup\{d_e(\tilde{F}_\gamma g(t_1), \tilde{F}_\gamma g(t_2)) \mid g\colon (X,d_2) \xrightarrow{1} ([0,\top],d_e), \gamma \in \Gamma\} + \delta \\
=\ & d_2^{\uparrow\Gamma}(t_1,t_2) + \delta
\end{aligned}$$

Analogously we can show that $d_2^{\uparrow\Gamma}(t_1,t_2) \leq d_1^{\uparrow\Gamma}(t_1,t_2) + \delta$ and this implies

$$d_e^\infty(d_1^{\uparrow\Gamma}, d_2^{\uparrow\Gamma}) \leq \delta.$$

In the contractive case the proof is analogous. □

We will now see that for the functors studied in this chapter, we have evaluation maps that satisfy the required conditions. To prove this, we first list some properties of our underlying pseudometrics.

Lemma 5.2.21

For all $a, b, a_i, b_i, q \in [0, \top]$ it holds that

1. $d_e(a \ominus q, b \ominus q) \leq d_e(a, b)$.
2. $d_e(\sup_{i \in I} a_i, \sup_{i \in I} b_i) \leq \sup_{i \in I} d_e(a_i, b_i)$.
3. $d_e(\inf_{i \in I} a_i, \inf_{i \in I} b_i) \leq \inf_{i \in I} d_e(a_i, b_i)$.
4. $d_\ominus(\sup_{i \in I} a_i, \sup_{i \in I} b_i) \leq \sup_{i \in I} d_\ominus(a_i, b_i)$.

Proof:

1. For $d_e(a \ominus q, b \ominus q) = |\max(a - q, 0) - \max(b - q, 0)|$ we have to distinguish four cases:
   (a) $\max(a - q, 0) = \max(b - q, 0) = 0$: $|0 - 0| \leq d_e(a, b)$
   (b) $\max(a - q, 0) = a - q$ and $\max(b - q, 0) = 0$:

$$\begin{aligned} &= |a - q - 0| \\ &\leq |a - q - (b - q)| && (b - q) \leq 0 \\ &\leq |a - q - b + q| \\ &= d_e(a, b) \end{aligned}$$

   (c) Similarly to (1b) for $\max(a - q, 0) = 0$ and $\max(b - q, 0) = b - q$.
   (d) $\max(a - q, 0) = a - q$ and $\max(b - q, 0) = b - q$

$$\begin{aligned} &|a - q - (b - q)| && (b - q) > 0 \\ &= |a - q - b + q| \\ &= d_e(a, b) \end{aligned}$$

2. For $d_e(\sup_{i \in I} a_i, \sup_{i \in I} b_i) \leq \sup_{i \in I} d_e(a_i, b_i)$: Here, we set $A = \sup_{i \in I} a_i$ and $B = \sup_{i \in I} b_i$. Next, we consider three cases:
   (a) $A = B$: Obviously, $d_e(A, B) = 0 \leq \sup_{i \in I} d_e(a_i, b_i)$
   (b) $A < B$: We know, that for all $\delta > 0$ there exists some b_i such that $B \leq b_i + \delta$ and each $a_i \leq A$:

$$\begin{aligned} B - A &\leq b_i + \delta - a_i \\ &= b_i - a_i + \delta \end{aligned}$$

   And since, this holds for all $\delta > 0$ we conclude $B - A \leq \sup_{i \in I} d_e(a_i, b_i)$.

(c) $B < A$: We know, that for all $\delta > 0$ there exists some a_i such that $A \leq a_i + \delta$ and each $b_i \leq B$:

$$\begin{aligned} A - B &\leq a_i + \delta - b_i \\ &= a_i - b_i + \delta \end{aligned}$$

And since, this holds for all $\delta > 0$ we conclude $A - B \leq \sup_{i \in I} d_e(a_i, b_i)$.

The proofs for 3 and 4 can be derived similarly.

□

Proposition 5.2.22

The following evaluation maps induce predicate liftings which are non-expansive with respect to the supremum metric and ω-continuous.

- The evaluation map $\gamma_{\mathcal{P}}$ for the (finite or general) powerset functor $\mathcal{P}$ with $\gamma \colon \mathcal{P}[0, \top] \to [0, \top]$ where $\gamma_{\mathcal{P}}(R) = \sup R$.
- The evaluation map $\gamma_{\mathcal{D}}$ for the (finitely or countably supported) probability distribution functor $\mathcal{D}$ (for its definition see Example 2.3.5) with $\gamma_{\mathcal{D}} \colon \mathcal{D}[0,1] \to [0,1]$ where $\gamma_{\mathcal{D}}(p) = \sum_{r \in [0,1]} r \cdot p(r)$. Note that $\gamma_{\mathcal{D}}$ corresponds to the expectation of the identity random variable.
- The evaluation map $\gamma_{\mathcal{M}}$ for the constant functor $\mathcal{M}X = [0, \top]$ with $\gamma_{\mathcal{M}} \colon [0, \top] \to [0, \top]$ and $\gamma_{\mathcal{M}}(r) = r$.
- The evaluation map $\gamma_{\mathcal{S}}$ for the constant functor $\mathcal{S}X = 1 = \{\bullet\}$ with $\gamma_{\mathcal{S}} \colon 1 \to [0, \top]$ and $\gamma_{\mathcal{S}}(r) = \top$.
- The evaluation maps $\gamma_{\mathcal{I}}$ for the input functor $\mathcal{I}X = X^A$ defined in Example 5.2.15 with $\gamma_a : [0,1]^A \to [0,1]$ and $\gamma_a(g) = g(a)$ for $g \in [0,1]^A$.

Proof: In the following let $f, g \colon X \to [0, \top]$.

- Observe that $\tilde{\mathcal{P}}_{\gamma_{\mathcal{P}}}(f) \colon \mathcal{P}X \to [0, \top]$ satisfies $\tilde{\mathcal{P}}_{\gamma_{\mathcal{P}}}(f)(Z) = \sup f[Z]$ where $Z \subseteq X$.

  We have

$$\begin{aligned} & d_e(\tilde{\mathcal{P}}_{\gamma_{\mathcal{P}}}(f)(Z), \tilde{\mathcal{P}}_{\gamma_{\mathcal{P}}}(g)(Z)) = |\sup f[Z] - \sup g[Z]| \leq \sup_{z \in Z} |f(z) - g(z)| \\ \leq \; & \sup_{x \in X} |f(x) - f(y)| = d_e^{\infty}(f, g) \end{aligned}$$

(cf. Lemma 5.2.21(2)). The same holds with Lemma 5.2.21(4) if we replace d_e by $d_\ominus$ and this implies non-expansiveness.

Furthermore we obtain ω-continuity via

$$\begin{aligned}\tilde{\mathcal{P}}_{\gamma_\mathcal{P}}(\sup_{i<\omega} f_i)(Z) &= \sup(\sup_{i<\omega} f_i)[Z] = \sup_{z\in Z}\sup_{i<\omega} f_i(z) = \sup_{i<\omega}\sup_{z\in Z} f_i(z)\\ &= \sup_{i<\omega} f_i[Z]\\ &= \tilde{\mathcal{P}}_{\gamma_\mathcal{P}}(f_i)(Z).\end{aligned}$$

- Observe that $\tilde{\mathcal{D}}_{\gamma_\mathcal{D}}(f)\colon \mathcal{D}X \to [0,1]$ satisfies $\tilde{\mathcal{D}}_{\gamma_\mathcal{D}}(f)(p) = \sum_{x\in X} f(x)\cdot p(x)$ where $p \in \mathcal{D}X$ is a probability distribution on X.

  We have $d_e(\tilde{\mathcal{D}}_{\gamma_\mathcal{D}}(f)(p), \tilde{\mathcal{D}}_{\gamma_\mathcal{D}}(g)(p)) = |\sum_{x\in X} f(x)\cdot p(x) - \sum_{x\in X} g(x)\cdot p(x)| \leq \sum_{x\in X} p(x)\cdot|f(x)-g(x)| \leq 1\cdot d_e^\infty(f,g)$. The same holds if we replace d_e by $d_\ominus$ and this implies non-expansiveness.

  Furthermore we obtain ω-continuity via

$$\begin{aligned}\tilde{\mathcal{D}}_{\gamma_\mathcal{D}}(\sup_{i<\omega} f_i)(x) &= \sum_{x\in X}(\sup_{i<\omega} f_i)(x)\cdot p(x)\\ &= \sup_{i<\omega}\sum_{x\in X} f_i(x)\cdot p(x)\end{aligned}$$

  For the last equality we have to exchange the sum and the supremum, for which we use the fact that

$$\sum_{x\in X} f(x) = \sup_{\substack{X'\subseteq X\\ X' \text{ finite}}}\sum_{x\in X'} f(x).$$

- Observe that $\tilde{\mathcal{M}}_{\gamma_\mathcal{M}}(f)\colon [0,\top] \to [0,\top]$ satisfies $\tilde{\mathcal{M}}_{\gamma_\mathcal{M}}(f)(r) = r$. Hence $\tilde{\mathcal{M}}_{\gamma_\mathcal{M}}$ maps every function to the identity, which is clearly non-expansive and ω-continuous.

- Observe that $\tilde{\mathcal{S}}_{\gamma_\mathcal{S}}(f)\colon \{\bullet\} \to [0,\top]$ satisfies $\tilde{\mathcal{S}}_{\gamma_\mathcal{S}}(f)(\bullet) = \top$. Hence $\tilde{\mathcal{S}}_{\gamma_\mathcal{S}}$ maps every function to the constant $\top$-function, which is again clearly non-expansive and ω-continuous.

- Observe that $\tilde{I}_{\gamma_a}(f)\colon X^A \to [0,\top]$ satisfies $\tilde{I}_{\gamma_a}(f)(h) = f^A(h)(a) = f(h(a))$. We have $|f(h(a)) - g(h(a))| \leq d_e^\infty(f,g)$ since $h(a) \in X$.

Furthermore for each $\gamma_a \in \Gamma$ we have $\tilde{\mathcal{I}}_{\gamma_a} f = f(h(a))$ with $h \in X^A$ and therefore we obtain ω-continuity, since $f_i \leq f_{i+1} \leq \cdots \leq \sup_{i<\omega} f_i$ implies

$$\begin{aligned}\tilde{\mathcal{I}}_{\gamma_a} f_i(h) = f_i(h(a)) &\leq f_{i+1}(h(a)) = \tilde{\mathcal{I}}_{\gamma_a} f_{i+1}(h) \\ &\leq \tilde{\mathcal{I}}_{\gamma_a}(\sup_{i<\omega} f_i)(h) \qquad\qquad = (\sup_{i<\omega} f_i)(h(a))\end{aligned}$$

for all $h \in X^A$ and therefore

$$\sup_{i<\omega}(\mathcal{I}_{\gamma_a} f_i(h)) = \sup_{i<\omega}(f_i(h(a))) = (\sup_{i<\omega} f_i)(h(a)) = \tilde{\mathcal{I}}_{\gamma_a}(\sup_{i<\omega} f_i)(h).$$

□

As shown in [BB+14] the evaluation map $\gamma_{\mathcal{P}}$ induces the Hausdorff lifting on metrics and the evaluation map $\gamma_{\mathcal{D}}$ the classical Kantorovich lifting for probability distributions [Vil09] introduced in the previous section.

Contractivity can be typically obtained by using a predicate lifting which is non-expansive and multiplying with a discount factor $0 < c < 1$, for instance by using $\gamma_{\mathcal{P}}(R) = c \cdot \sup R$ in the first item of Proposition 5.2.22 above.

It can be shown that the properties of evaluation maps are preserved under various forms of composition.

Proposition 5.2.23: Composition of Evaluation Maps

The following constructions on evaluation maps preserve non-expansiveness for the supremum metric and ω-continuity for the induced predicate liftings. Let $\gamma_F \colon F[0,\top] \to [0,\top]$, $\gamma_G \colon G[0,\top] \to [0,\top]$ be evaluation maps for functors F, G.

- $\gamma \colon F[0,\top] \times G[0,\top] \to [0,\top]$ with $\gamma = \gamma_F \circ \pi_1$, as an evaluation map for $F \times G$.
- $\gamma \colon F[0,\top] + G[0,\top] \to [0,\top]$ with $\gamma(t) = \gamma_F(t)$ if $t \in F[0,\top]$ and $\gamma(t) = 0$ otherwise, as an evaluation map for $F + G$.
- $\gamma \colon FG[0,\top] \to [0,\top]$ with $\gamma = \gamma_F \circ F\gamma_G$, as an evaluation map for FG.

Proof: In the following let $f, g \colon X \to [0,\top]$.

- Note that $(\widetilde{F \times G})_\gamma(f) = \tilde{F}_{\gamma_F}(f) \circ \pi_1$.

  Hence we obtain non-expansiveness via the supremum metric with

$$\begin{aligned}& d_e^\infty((\widetilde{F \times G})_\gamma(f), (\widetilde{F \times G})_\gamma(g)) = d_e^\infty(\tilde{F}_{\gamma_F}(f) \circ \pi_1, \tilde{F}_{\gamma_F}(g) \circ \pi_1) \\ = \;& d_e^\infty(\tilde{F}_{\gamma_F}(f), \tilde{F}_{\gamma_F}(g)) \leq d_e^\infty(f, g).\end{aligned}$$

The same holds if we replace d_e by $d_\ominus$. Furthermore ω-continuity can be shown as follows:

$$\begin{aligned}(\widetilde{F \times G})_\gamma(\sup_{i<\omega} f_i) &= \tilde{F}_{\gamma_F}(\sup_{i<\omega} f_i) \circ \pi_1 \\ &= \sup_{i<\omega} \tilde{F}_{\gamma_F}(f_i) \circ \pi_1 = \sup_{i<\omega} (\widetilde{F \times G})_\gamma(f_i).\end{aligned}$$

- Note that $(\widetilde{F+G})_\gamma(f)(t) = \tilde{F}_{\gamma_F}(f)(t)$ if $t \in FX$ and $(\widetilde{F+G})_\gamma(f)(t) = 0$ if $t \in GX$. Hence we can deduce non-expansiveness via the supremum metric by observing that $d_e((\widetilde{F \times G})_\gamma(f)(t), (\widetilde{F \times G})_\gamma(g)(t))$ is either

$$d_e(\tilde{F}_{\gamma_F}(f)(t), \tilde{F}_{\gamma_F}(g)(t)) \leq d_e^\infty(f, g),$$

if $t \in FX$, or the value equals 0, if $t \in GX$. The same holds if we replace d_e by $d_\ominus$.

Furthermore ω-continuity can be shown as follows: whenever $t \in FX$ we have

$$\begin{aligned}(\widetilde{F+G})_\gamma(\sup_{i<\omega} f_i))(t) &= \tilde{F}_{\gamma_F}(\sup_{i<\omega} f_i)(t) = (\sup_{i<\omega} \tilde{F}_{\gamma_F}(f_i))(t) \\ &= (\sup_{i<\omega} (\widetilde{F+G})_\gamma(f_i))(t).\end{aligned}$$

And if $t \in GX$ both values are equal to 0.

- Note that we have

$$\widetilde{FG}_\gamma(f) = \gamma \circ FGf = \gamma_F \circ F\gamma_G \circ FGf = \gamma_F \circ F(\gamma_G \circ Gf) = \tilde{F}_{\gamma_F}(\tilde{G}_{\gamma_G}(f)).$$

Hence we obtain non-expansiveness via the supremum metric via

$$\begin{aligned}& d_e^\infty(\widetilde{FG}_\gamma(f), \widetilde{FG}_\gamma(g)) = d_e^\infty(\tilde{F}_{\gamma_F}(\tilde{G}_{\gamma_G}(f)), \tilde{F}_{\gamma_F}(\tilde{G}_{\gamma_G}(g))) \\ \leq\ & d_e^\infty(\tilde{G}_{\gamma_G}(f), \tilde{G}_{\gamma_G}(g)) \\ \leq\ & d_e^\infty(f, g).\end{aligned}$$

The same holds if we replace d_e by $d_\ominus$. Furthermore it is easy to show ω-continuity:

$$\begin{aligned}\widetilde{FG}_\gamma(\sup_{i<\omega} f_i) &= \tilde{F}_{\gamma_F}(\tilde{G}_{\gamma_G}(\sup_{i<\omega} f_i)) = \tilde{F}_{\gamma_F}(\sup_{i<\omega} \tilde{G}_{\gamma_G}(f_i)) \\ &= \sup_{i<\omega} \tilde{F}_{\gamma_F}(\tilde{G}_{\gamma_G}(f_i)) = \sup_{i<\omega} \widetilde{FG}_\gamma(f_i).\end{aligned}$$

□

Now we can define behavioural distance on a coalgebra, using the Kantorovich lifting. Note that the behavioural distance is parameterized over Γ, since, if we are given a coalgebra in **Set**, the notion of behaviour in the metric case is dependent on the chosen functor lifting.

Definition 5.2.24: Behavioural Distance

Let the coalgebra $\alpha\colon X \to FX$ and a set of evaluation maps Γ for F be given. We define the pseudometric $d_\alpha\colon X \times X \to [0, \top]$ as the smallest fixpoint of

$$d_\alpha = d_\alpha^{\uparrow\Gamma} \circ (\alpha \times \alpha)$$

Note that every lifting of metrics is necessarily monotone (since it turns the identity into a non-expansive function, cf. [BB+14]). Since in addition the space of pseudometrics forms a complete lattice (where sup is taken pointwise), the smallest fixpoint exists by Knaster-Tarski.

It has been shown in [BB+18] that whenever the Kantorovich lifting preserves metrics (which is the case for our examples) and the final chain converges, then d_α characterizes behavioural equivalence, i.e., $d_\alpha(x, y) = 0$ iff $x \sim y$.

Example 5.2.25

Using the building blocks introduced above we consider the following coalgebras with their corresponding behavioural metrics, generalizing notions from the literature. In both cases we are interested in the smallest fixpoint.

- *Metric transition systems* [AFS09]: $FX = [0, \top] \times \mathcal{P}X$ with two evaluation maps $\gamma_i\colon [0, \top] \times \mathcal{P}[0, \top] \to [0, \top]$, $i \in \{1, 2\}$ with $\gamma_1(r, R) = r$, $\gamma_2(r, R) = \sup R$.

  This gives us the following fixpoint equation, where d^H is the Hausdorff lifting of a metric d. Let $\alpha(x) = (r_x, S_x)$, $\alpha(y) = (r_y, S_y)$, then

  $$d(x, y) = \max\{|r_x - r_y|, d^H(S_x, S_y)\}$$

  Analogously to the argumentation in Example 5.2.15 it suffices to find one non-expansive function as a witness that $|\gamma_2 \circ Ff(\alpha(x)) - \gamma_2 \circ Ff(\alpha(y))| = d^H(S_x, S_y)$. This function is already defined in [Ker16] but for completeness we introduce it here once again:

  $$f(x) = \min_{x_2 \in S_y} d(x, x_2)$$

In addition we know from [Ker16] that no non-expansive function exists, which increases the distance based on γ_2, since the Wasserstein lifting serves here as an upper bound for the Kantorovich lifting. Regarding γ_1 the definition of f has no impact since $Ff(r, S)$ is mapped to $(r, f[S]) \in F[0, 1]$ and therefore the max is taken over the values given by $|r_x - r_y|$ and $d^H(S_x, S_y)$.

- *Probabilistic transition systems:* $GX = \mathcal{D}X + 1$ with two evaluation maps $\bar{\gamma}_{\mathcal{D}}, \gamma_\bullet \colon \mathcal{D}[0, 1] + 1 \to [0, 1]$ with $\bar{\gamma}_{\mathcal{D}}(p) = \gamma_{\mathcal{D}}(p)$, $\gamma_\bullet(p) = 0$ where $p \in \mathcal{D}[0, 1]$, $\bar{\gamma}_{\mathcal{D}}(\bullet) = 0$, $\gamma_\bullet(\bullet) = 1$.

  This gives us the following fixpoint equation, where d^K is the (probabilistic) Kantorovich lifting of a metric d. Let $T = \{x \mid \alpha(x) = \bullet\}$ and let $p_x = \alpha(x) \neq \bullet$.

$$d(x, y) = \begin{cases} 1 & \text{if } x \in T, y \notin T \text{ or } x \notin T, y \in T \\ 0 & \text{if } x, y \in T \\ d^K(p_x, p_y) & \text{otherwise} \end{cases}$$

Some of the results on (real-valued) modal logics in Section 5.3 will require that the fixpoint iteration terminates in ω steps. This is related to the fact that the original Hennessy-Milner theorem requires finite branching.

Proposition 5.2.26

Let Γ be a set of evaluation maps and let $\alpha \colon X \to FX$ be a coalgebra. We define an ascending sequence of metrics $d_i \colon X \times X \to [0, \top]$ as follows: d_0 is the constant 0-function and $d_{i+1} = d_i^{\uparrow\Gamma} \circ \alpha \times \alpha$. Furthermore $d_\omega = \sup_{i<\omega} d_i$.

- If for all evaluation maps $\gamma \in \Gamma$ the induced predicate liftings are ω-continuous (see Definition 5.2.18) and F is ω-accessible, the fixpoint d_α equals d_ω.
- If for all evaluation maps $\gamma \in \Gamma$ the induced predicate liftings are contractive with respect to the supremum metric (see Definition 5.2.18), the fixpoint d_α equals d_ω.

Proof: First note that d_ω as the pointwise supremum of pseudometrics, is again a pseudometric.

- We assume that every $\tilde{F}_\gamma$ is ω-continuous and F is ω-accessible. Obviously

$d_\omega \leq d_\alpha$ because $d_\omega = \sup_{i<\omega} d_i \leq d_\alpha$. Next, we need to show $d_\alpha \leq d_\omega$. Since d_α is the smallest fixpoint of $d_\alpha = d_\alpha^{\uparrow\Gamma} \circ (\alpha \times \alpha)$ we need to show, that d_ω is a prefix point. Given $x, y \in FX$, we have to prove that:

$$\begin{aligned} & d_\omega^{\uparrow\Gamma}(\alpha(x), \alpha(y)) \\ = \; & \sup\{d_e(\tilde{F}_\gamma f(\alpha(x)), \tilde{F}_\gamma f(\alpha(y))) \mid f\colon (X, d_\omega) \xrightarrow{1} ([0,\top], d_e), \gamma \in \Gamma\} \\ \leq \; & d_\omega(x, y) \end{aligned}$$

So we have to show that $d_e(\tilde{F}_\gamma f(\alpha(x)), \tilde{F}_\gamma f(\alpha(y)) \leq d_\omega(x, y)$ for every non-expansive function $f\colon (X, d_\omega) \xrightarrow{1} ([0,\top], d_e)$.

Since F is ω-accessible, there is a finite set $Z \subseteq X$ with $\alpha(x), \alpha(y) \in FZ$. We can assume that Z is non-empty. Now define $g\colon X \to [0,\top]$ as $g(z) = f(z)$ if $z \in Z$ and $g(z) = \top$ otherwise. Note that Ff and Fg agree on $\alpha(x), \alpha(y)$.

We approximate the function g on Z with an ascending chain of functions $g_0 \leq g_1 \leq g_2 \leq \ldots$ with $g_i\colon (Z, d_i) \xrightarrow{1} ([0,\top], d_e)$, i.e., each g_i is non-expansive for d_i. We define $g_i(z) = \inf\{g(u) + d_i(u, z) \mid u \in X\}$ as in Lemma 5.2.4, which means that $g_i \leq g$ and every g_i is non-expansive as desired. Since $d_i \leq d_{i+1}$ it also follows that $g_i \leq g_{i+1}$.

Since $g(u) = \top$ if $u \notin Z$, we know that for $z \in Z$ the infimum is reached for $u \in Z$ and hence $g_i(z) = \inf\{g(u) + d_i(u, z) \mid u \in Z\}$. Furthermore whenever $z \in Z$

$$\begin{aligned} g(z) - g_i(z) &= g(z) - \inf\{g(u) + d_i(u, z) \mid u \in Z\} \\ &= \sup\{g(z) - g(u) - d_i(u, z) \mid u \in Z\} \\ &= \sup\{f(z) - f(u) - d_i(u, z) \mid u \in Z\} \\ &\leq \sup\{d_\omega(u, z) - d_i(u, z) \mid u \in Z\} \end{aligned}$$

Since Z is finite this value converges to 0 when i approaches ω and hence $\sup_{i<\omega} g_i(z) = g(z)$ whenever $z \in Z$. This means that $\sup_{i<\omega} g_i$, g agree on $\alpha(x), \alpha(y) \in FZ$.

Hence, using the fact that $\tilde{F}_\gamma$ is ω-continuous and Lemma 5.2.21(2):

$$\begin{aligned} d_e(\tilde{F}_\gamma f(\alpha(x)), \tilde{F}_\gamma f(\alpha(y)) &= d_e(\tilde{F}_\gamma g(\alpha(x)), \tilde{F}_\gamma g(\alpha(y))) \\ &= d_e(\tilde{F}_\gamma(\sup_{i<\omega} g_i)(\alpha(x)), \tilde{F}_\gamma(\sup_{i<\omega} g_i)(\alpha(y))) \\ &= d_e(\sup_{i<\omega} \tilde{F}_\gamma g_i(\alpha(x)), \sup_{i<\omega} \tilde{F}_\gamma g_i(\alpha(y))) \\ &\leq \sup_{i<\omega} d_e(\tilde{F}_\gamma g_i(\alpha(x)), \tilde{F}_\gamma g_i(\alpha(y))) \\ &\leq \sup_{i<\omega} d_i^{\uparrow\Gamma}(\alpha(x), \alpha(y)) \\ &= \sup_{i<\omega} d_{i+1}(\alpha(x), \alpha(y)) \\ &= d_\omega(x, y) \end{aligned}$$

The inequality is due to the fact that non-expansiveness of g_i for d_i implies non-expansiveness of $\tilde{F}_\gamma(g_i)$ for $d_i^{\uparrow\Gamma}$.

- We assume that every $\tilde{F}_\gamma$ is contractive with respect to the supremum metric (for some constant c). Due to Proposition 5.2.20 we have that for two metrics $d_1, d_2 \colon X \times X \to [0, \top]$ it holds that

  $$d_e^\infty(d_1^{\uparrow\Gamma}, d_2^{\uparrow\Gamma}) \leq c \cdot d_e^\infty(d_1, d_2)$$

  where $0 < c < 1$.

  Note that $\top \neq \infty$, which means that the set of all such real-valued bounded functions forms a complete metric space with respect to the supremum metric. Hence we obtain the fixpoint in ω steps via the Banach fixpoint theorem.

□

Hence, if we are working with the finite powerset functor or the finitely supported distribution functor, the first case applies, whereas in the case of contractiveness, these restrictions are unnecessary (compare this with the result of [TR98] which guarantees the existence of a final coalgebra for a class of locally contractive functors).

To close this section we summarize, that behavioural distance d_α via the Kantorovich lifting given in Definition 5.2.13 characterizes behaviour equivalence under the restrictions in Definition 5.2.18 and whenever the Kantorovich lifting preserves metrics.

Having laid the foundation for our evaluation maps we proceed in the next section with the work on modal logics and concentrate on a quantitative Hennessy-Milner theorem as illustrated in Figure 5.6.

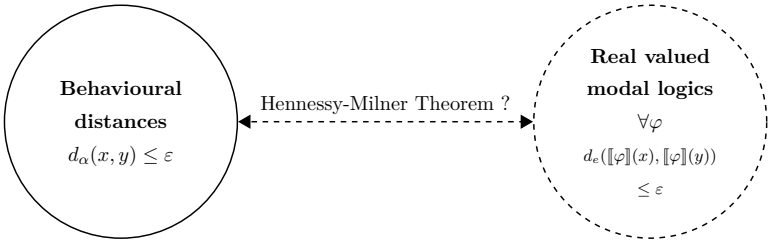


Figure 5.6: A quantitative version of the Hennessy-Milner theorem

5.3 Modal Logics for the Metric Case

We now define a coalgebraic modal logic $\mathcal{M}(\Gamma)$, which is inspired by [BW05]. Assume that Γ is a set of real-valued evaluation maps.

Definition 5.3.1: Coalgebraic Modal Logic (syntax)

Given a set Γ of real-valued evaluation maps. The syntax of the logic is defined by the following grammar:

$$\varphi ::= \top \mid [\gamma]\psi \mid \min(\psi, \psi') \mid \neg\psi \mid \psi \ominus q$$

where $\gamma \in \Gamma$.

Given a coalgebra $\alpha\colon X \to FX$ and a formula φ, the semantics of such a formula is given by a real-valued predicate $[\![\varphi]\!]_\alpha\colon X \to [0, \top]$, defined inductively, where $\gamma \in \Gamma$, $q \in \mathbb{Q}_0 \cap [0, \top]$:

$$\begin{array}{lll} [\![\top]\!]_\alpha & := \lambda x.\top & [\![[\gamma]\psi]\!]_\alpha := \gamma \circ F[\![\psi]\!]_\alpha \circ \alpha \\ [\![\min(\psi,\psi')]\!]_\alpha & := \min\{[\![\psi]\!]_\alpha, [\![\psi']\!]_\alpha\} & [\![\neg\psi]\!]_\alpha := \top - [\![\psi]\!]_\alpha \\ [\![\psi \ominus q]\!]_\alpha & := [\![\psi]\!]_\alpha \ominus q & \end{array}$$

Again we will occasionally omit the subscript α and an overview of the modal depth $md(\varphi)$ is given in Table 5.1.

φ:	$\top$	$[\gamma]\psi$	$\min(\psi,\psi')$	$\neg\psi$	$\psi \ominus q$
$md(\varphi)$:	0	$md(\psi)+1$	$\max\{md(\psi), md(\psi')\}$	$md(\psi)$	$md(\psi)$

Table 5.1: Overview of the modal logic formula and their modal depths $md(\varphi)$.

Note that, given a state x and a logical formula φ, we do not just obtain a true value (true, false) dependent on whether x satisfies the formula or not. Instead we obtain a value in the interval $[0, 1]$ that measures the degree or weight according to which x satisfies φ.

Definition 5.3.2: Logical Distance

Let $\alpha\colon X \to FX$ be a coalgebra and let $x, y \in X$. We define the logical distance of x, y as

$$d_\alpha^L(x,y) = \sup\{d_e([\![\varphi]\!]_\alpha(x), [\![\varphi]\!]_\alpha(y)) \mid \varphi \in \mathcal{M}(\Gamma)\}.$$

We also define the logical distance up to modal depth i.

$$d_i^L(x,y) = \sup\{d_e([\![\varphi]\!]_\alpha(x), [\![\varphi]\!]_\alpha(y)) \mid \varphi \in \mathcal{M}(\Gamma), md(\varphi) \leq i\}.$$

Example 5.3.3

We are considering probabilistic transition systems with evaluation maps as defined in Example 5.2.9.
The formula $\varphi = [\bar{\gamma}_{\mathcal{D}}][\gamma_\bullet]\top$ distinguishes the states x, y in Figure 5.5. The formula $\psi = [\gamma_\bullet]\top$ evaluates to a predicate $[\![\psi]\!]$ that assigns 1 to terminating states and 0 to non-terminating states. Now x makes a transition to a terminating state with probability $\frac{1}{2}$, which means that $[\![\varphi]\!](x) = \bar{\gamma}_{\mathcal{D}}(\mathcal{D}[\![\psi]\!](\alpha(x))) = \frac{1}{2}$. Similarly $[\![\varphi]\!](y) = \frac{1}{2} + \varepsilon$. Hence $d_\alpha^L(x,y) \geq d_e([\![\varphi]\!](x), [\![\varphi]\!](y)) = \varepsilon$. (In fact, the logical distance equals ε.)

We will now show that the logical distance d_α^L and the behavioural distance d_α coincide, i.e. a quantitative version of the Hennessy-Milner theorem, by generalizing the proof from [BW05]. Note that in some respects we simplify with respect to [BW05] by not working in **Meas**, the category of measurable spaces, but in a discrete setting. On the other hand, we generalize by considering arbitrary **Set**-endofunctors.

Theorem 5.3.4

Let d_i be the sequence of pseudometrics from Proposition 5.2.26. Then:

1. For every $i \in \mathbb{N}_0$ $d_i^L \leq d_i$.
2. For every φ with $md(\varphi) \leq i$ we have non-expansiveness:
$$[\![\varphi]\!]\colon (X, d_i) \to ([0, \top], d_e)$$
3. $d_\alpha^L \leq d_\alpha$.

Proof:

- We prove (1) and (2) jointly by induction on i.
  - $i = 0$: It is easy to see that for every φ with $md(\varphi) = 0$ the function $[\![\varphi]\!]$ is

constant. Due to this $d_0^L(x,y) = \sup\{d_e([\![\varphi]\!](x), [\![\varphi]\!](y)) \mid md(\varphi) = 0\} = 0 \leq d_0(x,y)$. All constant functions are non-expansive for d_0.

– $i \to i+1$: We first show that for every φ with $md(\varphi) \leq i+1$ that $[\![\varphi]\!] \colon (X, d_{i+1}) \xrightarrow{1} ([0,\top], d_e)$ is non-expansive by structural induction on φ.

  * $\varphi = \top$: This case can not occur for $md(\varphi) > 0$.
  * $\varphi = [\gamma]\psi$: Here $md(\psi) \leq i$ and $[\![[\gamma]\psi]\!] = \gamma \circ F[\![\psi]\!] \circ \alpha$, where we have a composition of non-expansive functions:
    · The evaluation map $\gamma \colon (F[0,\top], d_e^{\uparrow\Gamma}) \xrightarrow{1} ([0,\top], d_e)$ is non-expansive by Lemma 5.2.16.
    · $[\![\psi]\!] \colon (X, d_i) \xrightarrow{1} ([0,1], d_e)$ is non-expansive by the outer induction hypothesis and, since the lifting preserves non-expansive functions, $F[\![\psi]\!] \colon (FX, d_i^{\uparrow\Gamma}) \xrightarrow{1} (F[0,1], d_e^{\uparrow\Gamma})$.
    · By definition $d_{i+1} = d_i^{\uparrow\Gamma} \circ (\alpha \times \alpha)$, hence $\alpha \colon (X, d_{i+1}) \xrightarrow{1} (FX, d_i^{\uparrow\Gamma})$ (it is even an isometry).
  * $\varphi = \min(\psi, \psi')$: We know that $md(\psi), md(\psi') \leq md(\varphi) \leq i+1$. The inner induction hypothesis implies that $[\![\psi]\!]$, $[\![\psi']\!]$ are both non-expansive for d_{i+1}. Using Lemma 5.2.21(3) we know that for all $x, y \in X$

$$\begin{aligned} & d_e([\![\varphi]\!](x), [\![\varphi]\!](y)) \\ = \;& d_e(\min\{[\![\psi]\!](x), [\![\psi']\!](x)\}, \min\{[\![\psi]\!](y), [\![\psi']\!](y)\}) \\ \leq \;& \max\{d_e([\![\psi]\!](x), [\![\psi]\!](y)), d_e([\![\psi']\!](x), [\![\psi']\!](y))\} \leq d_{i+1}(x,y) \end{aligned}$$

  * $\varphi(x) = \neg\psi(x)$: We know that $md(\psi) = md(\varphi) \leq i+1$. The inner induction hypothesis implies that $[\![\psi]\!]$ is non-expansive for d_{i+1}. For $x, y \in X$ we obtain

$$\begin{aligned} & d_e([\![\varphi]\!](x), [\![\varphi]\!](y)) \\ = \;& d_e([\![\top - \psi]\!](x), [\![\top - \psi]\!](y)) = |\top - [\![\psi]\!](x) - (\top - [\![\psi]\!](y))| \\ = \;& |[\![\psi]\!](y) - [\![\psi]\!](x)| = d_e([\![\psi]\!](x), [\![\psi]\!](y)) \leq d_{i+1}(x,y) \end{aligned}$$

  * $\varphi(x) = \psi \ominus q$: We know that $md(\psi) = md(\varphi) \leq i+1$. The inner induction hypothesis implies that $[\![\psi]\!]$ is non-expansive for d_i. For all $x, y \in X$ we know by Lemma 5.2.21(1) that

$$\begin{aligned} d_e([\![\varphi]\!](x), [\![\varphi]\!](y)) &= d_e([\![\psi]\!](x) \ominus q, [\![\psi]\!](y) \ominus q) \\ &\leq d_e([\![\psi]\!](x), [\![\psi]\!](y)) \leq d_{i+1}(x,y) \end{aligned}$$

Thus, we conclude by the non-expansiveness of all formulas φ with $md(\varphi) \leq i$ that $d_i^L(x,y) = \sup\{d_e([\![\varphi]\!](x), [\![\varphi]\!](y) \mid md(\varphi) \leq i\} \leq d_i(x,y)$.

- (3) follows directly from (1) by taking the supremum on both sides and observing that $d_\alpha \geq \sup_{i<\omega} d_i$.

□

Note that from Theorem 5.3.4 it also follows that for every formula φ the function $[\![\varphi]\!]$ is non-expansive. Non-expansiveness is analogous to bisimulation-invariance that holds for formulas in a classical logic. In particular, in the classical case if $x \sim y$, then $[\![\varphi]\!](x) = [\![\varphi]\!](y)$ for every φ, in other words $[\![\varphi]\!]$ is non-expansive for the discrete metric d.

The other inequality ($d_\alpha^L \geq d_\alpha$) is more difficult to prove: we will first show that each d_i is totally bounded and then show that each non-expansive function can be approximated at each pair of points by a modal formula. Since modal formulas are closed under min and max, this enables us to use a variant of a lemma from [Ash72] to prove that the formulas form a dense subset of all non-expansive functions. In order to achieve the approximation, we need all operators of the logic.

We first have to recall some definitions from real-valued analysis. The first definition allows to cover a possibly infinite metric space by a finite number of so-called ε-balls.

Definition 5.3.5: Total Boundedness

A pseudometric space (X,d) is totally bounded iff for every $\varepsilon > 0$ there exist finitely many elements $x_1, \ldots, x_n \in X$ such that $X = \bigcup_{i=1}^n \mathcal{B}_\varepsilon(x_i)$ where $\mathcal{B}_\varepsilon(x_i) = \{z \in X \mid d(z, x_i) \leq \varepsilon\}$ denotes the ε-ball around x_i.

Our first result is to show that the lifting preserves total boundedness, by adapting a proof from [WS+18a] from a specific functor to arbitrary functors.

Proposition 5.3.6

Let (X,d) be a totally bounded pseudometric space, then $(FX, d^{\uparrow\Gamma})$ is totally bounded as well.

Proof: We denote the set of all non-expansive functions $f\colon (X,d) \xrightarrow{1} ([0,\top], d_e)$ with $\mathcal{F}$. We know from [WS+18a, Lemma 5.6] that if (X,d) is a totally bounded space, then $\mathcal{F}$ is also totally bounded with respect to the supremum metric d_e^∞. This is a variation of the Arzelà-Ascoli theorem.[1] Due to this for all $\varepsilon > 0$ there exists

[1] Actually this lemma is stated for $\top = 1$, but the proof can be straightforwardly adapted.

a finite set $\{f_1, \ldots, f_n\}$ with $f_i \in \mathcal{F}$ such that for all $f \in \mathcal{F}$ we have one f_i with $d_e^\infty(f, f_i) \leq \varepsilon$. Given a fixed ε, we denote the function f_i corresponding to f by $\hat{f}$.

Furthermore, since the predicate lifting is non-expansive with respect to the supremum metric (cf. Definition 5.2.18), we know that for all $t \in FX$ $d_e(\tilde{F}_\gamma f(t), \tilde{F}_\gamma \hat{f}(t)) \leq \varepsilon$. And we have, using the triangle inequality,

$$\begin{aligned}
d^{\uparrow\Gamma}(t_1, t_2) &= \sup\{d_e(\tilde{F}_\gamma f(t_1), \tilde{F}_\gamma f(t_2)) \mid f \in \mathcal{F}, \gamma \in \Gamma\} \\
&\leq \sup\{d_e(\tilde{F}_\gamma f(t_1), \tilde{F}_\gamma \hat{f}(t_1)) + d_e(\tilde{F}_\gamma \hat{f}(t_1), \tilde{F}_\gamma \hat{f}(t_2)) \\
&\qquad + d_e(\tilde{F}_\gamma \hat{f}(t_2), \tilde{F}_\gamma f(t_2)) \mid f \in \mathcal{F}, \gamma \in \Gamma\} \\
&\leq \sup\{d_e(\tilde{F}_\gamma \hat{f}(t_1), \tilde{F}_\gamma \hat{f}(t_2)) \mid f \in \mathcal{F}, \gamma \in \Gamma\} + 2 \cdot \varepsilon \\
&= \sup\{d_e(\tilde{F}_\gamma f_i(t_1), \tilde{F}_\gamma f_i(t_2)) \mid i \in \{1, \ldots, n\}\} + 2 \cdot \varepsilon
\end{aligned}$$

By assumption Γ is finite and we assume $\Gamma = \{\gamma_1, \ldots, \gamma_k\}$. Now we define $g \colon FX \to [0, \top]^{k \cdot n}$ with

$$g(t) = (\tilde{F}_{\gamma_1} f_1(t), \ldots, \tilde{F}_{\gamma_1} f_n(t), \ldots, \tilde{F}_{\gamma_k} f_1(t), \ldots, \tilde{F}_{\gamma_k} f_n(t)).$$

Since $[0, \top]^{k \cdot n}$ is compact under the supremum metric there exists finitely many $u_1, \ldots, u_m \in [0, \top]^{k \cdot n}$ such that $\bigcup_{i=1}^m B_\varepsilon(u_i) = [0, \top]^{k \cdot n}$. The preimages of all balls cover FX and each preimage $g^{-1}(B_\varepsilon(u_i))$ has a diameter at most $4 \cdot \varepsilon$: For $s_1, s_2 \in g^{-1}(B_\varepsilon(u_i))$ it holds that:

$$\begin{aligned}
d^{\uparrow\Gamma}(s_1, s_2) &\leq 2 \cdot \varepsilon + \sup_{i \in \{1, \ldots, n\}, \gamma \in \Gamma} \{d_e(\tilde{F}_\gamma f_i(s_1), \tilde{F}_\gamma f_i(s_2))\} \\
&= 2 \cdot \varepsilon + d_e^\infty(g(s_1), g(s_2)) \\
&\leq 2 \cdot \varepsilon + d_e^\infty(g(s_1), u_i) + d_e^\infty(u_i, g(s_2)) \\
&\leq 2 \cdot \varepsilon + 2 \cdot \varepsilon
\end{aligned}$$

Hence, given a $\delta > 0$ we can set $\varepsilon = \frac{\delta}{4}$ and obtain balls $g^{-1}(B_\varepsilon(u_i))$ of diameter at most δ, which cover FX. □

Using this result it can be shown that every pseudometric in the ascending chain from Proposition 5.2.26 (apart from d_ω) is totally bounded.

Proposition 5.3.7

Let d_i be the sequence of pseudometrics from Proposition 5.2.26. Then every (X, d_i) is a totally bounded pseudometric space.

Proof: We proceed by induction on i and start with the trivial (constant) pseudometric d_0, hence(X, d_0) is a totally bounded pseudometric space, since $d_0(x, y) = 0 < \varepsilon$ for all $\varepsilon > 0$.

For (X, d_{i+1}) we know from the induction hypothesis and by Proposition 5.3.6 that $(FX, d_i^{\uparrow\Gamma})$ is totally bounded. So, for every $\varepsilon > 0$, there exist $t_1, \cdots, t_n \in FX$ such that $FX = \bigcup_{i=1}^{n} B_{\frac{\varepsilon}{2}}(t_i)$, which implies $X = \bigcup_{i=1}^{n} \alpha^{-1}(B_{\frac{\varepsilon}{2}}(t_i))$. Let $x, y \in \alpha^{-1}(B_{\frac{\varepsilon}{2}}(t_i))$. Then x, y have at most distance ε: $d_{i+1}(x, y) = d_i^{\uparrow\Gamma}(\alpha(x), \alpha(y)) \leq d_i^{\uparrow\Gamma}(\alpha(x), t_i) + d_i^{\uparrow\Gamma}(t_i, \alpha(y)) \leq \varepsilon$. □

Since total boundedness is not preserved by taking a supremum, d_ω is not necessarily totally bounded and we can not iterate the argument. This is one of the reasons for requiring that the fixpoint is reached in ω steps in Theorem 5.3.9 below.

In the next step we show that the formulas are dense in the non-expansive functions.

Proposition 5.3.8

$\{[\![\varphi]\!] \colon X \to [0,1] \mid md(\varphi) \leq i\}$ is dense (wrt. the supremum metric) in $\{f \colon (X, d_i^L) \xrightarrow{1} ([0, \top], d_e)\}$.

Proof: We have to show that for every $\varepsilon > 0$ and every non-expansive function $f \colon (X, d_i^L) \xrightarrow{1} ([0, \top], d_e)$, there exists a formula φ with $md(\varphi) \leq i$ and $d_e^\infty(f, [\![\varphi]\!]) \leq \varepsilon$.

From Proposition 5.3.7 we know that (X, d_i) is totally bounded and since $d_i^L \leq d_i$ (Theorem 5.3.4) we can infer that (X, d_i^L) is also totally bounded (since $B_\varepsilon^{d_i}(x) = \{z \in X \mid d_i(z, x) \leq \varepsilon\} \subseteq \{z \in X \mid d_i^L(z, x) \leq \varepsilon\} = B_\varepsilon^{d_i^L}(x)$).

We use the following result from [WS+18a, Lemma 5.8], which is a variation of a lemma by Ash [Ash72, Lemma A.7.2], adapted from compact spaces and continuous functions to totally bounded spaces and non-expansive functions. (Actually this lemma is stated for $\top = 1$, but the proof can be straightforwardly adapted.)

> Let (X, d) be a totally bounded pseudometric space and let $\mathcal{G}$ be a subset of $\mathcal{F} = \{f \colon (X, d) \xrightarrow{1} ([0, \top], d_e)$ such that $f_1, f_2 \in \mathcal{G}$ implies $\min\{f_1, f_2\}, \max\{f_1, f_2\} \in \mathcal{G}$. If each $f \in \mathcal{F}$ can be approximated at each pair of points by functions in $\mathcal{G}$, then $\mathcal{G}$ is dense in $\mathcal{F}$ (wrt. d_e^∞).

Here $d = d_i^L$ and $\mathcal{G}$ is the set of functions $[\![\varphi]\!]$, where $md(\varphi) \leq i$. Since min and $\neg$ are operators of the logic and neither increases the modal depth, $\mathcal{G}$ is clearly closed under max and min.

Now let $f \colon (X, d) \xrightarrow{1} ([0, \top], d_e)$, $\delta > 0$ and $x, y \in X$. We have to show that there exists a modal formula φ (with modal depth at most i) that approximates f at these points. That is $d_e(f(x), [\![\varphi]\!](x)) \leq \delta$ and $d_e(f(y), [\![\varphi]\!](y)) \leq \delta$.

We concentrate on the case $f(x) \geq f(y)$, the other case is analogous.

$$\Delta := f(x) - f(y) \leq d_i^L(x, y) = \sup\{d_e([\![\varphi]\!](x), [\![\varphi]\!](y)) \mid md(\varphi) \leq i\}$$

Then there exists a formula ψ with $md(\psi) \leq i$ such that $\Delta - \delta \leq [\![\psi]\!](x) - [\![\psi]\!](y)$ (whenever we find ψ with $\Delta - \delta \leq [\![\psi]\!](y) - [\![\psi]\!](x)$ we use negation). We can assume that $[\![\psi]\!](y) \leq [\![\psi]\!](x)$.

Whenever $[\![\psi]\!](y) \geq f(y)$ we set $\varphi = \min(\psi \ominus r, s)$ with $r, s \in \mathbb{Q}_0 \cap [0, \top]$ chosen as follows:

$$\begin{aligned} r &\in [[\![\psi]\!](y) - f(y) - \delta, [\![\psi]\!](y) - f(y)] \\ s &\in [f(x), f(x) + \delta] \end{aligned}$$

Since the depth of a formula only increases with the modality operator, it holds that $md(\varphi) \leq i$.

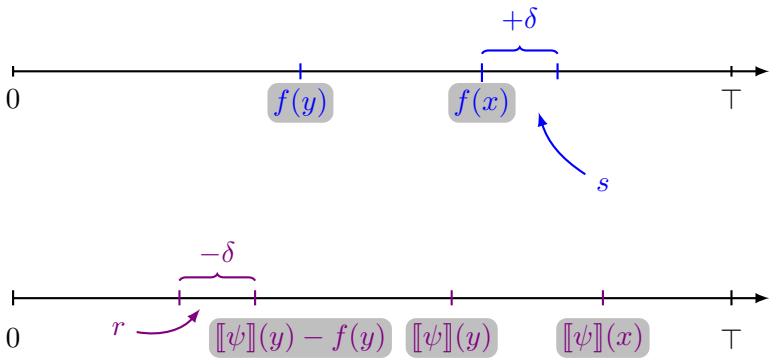


Figure 5.7: The values s and r with their corresponding intervals.

We show that φ is the required formula. First note that $r \leq [\![\psi]\!](y) \leq [\![\psi]\!](x)$ and $f(y) \leq f(x) \leq s$ as illustrated in Figure 5.7.

$$\begin{aligned} [\![\varphi]\!](y) - f(y) &= \min\{[\![\psi]\!](y) - r, s\} - f(y) \\ &= \min\{\underbrace{[\![\psi]\!](y) - f(y) - r}_{\leq \delta}, s - f(y)\} \leq \delta \\ f(y) - [\![\varphi]\!](y) &= f(y) - \min\{[\![\psi]\!](y) - r, s\} \\ &= \max\{\underbrace{f(y) - [\![\psi]\!](y) + r}_{\leq 0}, \underbrace{f(y) - s}_{\leq 0}\} \leq \delta \\ [\![\varphi]\!](x) - f(x) &= \min\{[\![\psi]\!](x) - r, s\} - f(x) \\ &= \min\{[\![\psi]\!](x) - f(x) - r, \underbrace{s - f(x)}_{\leq \delta}\} \leq \delta \\ f(x) - [\![\varphi]\!](x) &= f(x) - \min\{[\![\psi]\!](x) - r, s\} \\ &= \max\{\underbrace{f(x) - [\![\psi]\!](x) + r}_{\leq \delta}, \underbrace{f(x) - s}_{\leq 0}\} \leq \delta \end{aligned}$$

$f(x) - [\![\psi]\!](x) + r \leq \delta$ holds since $[\![\psi]\!](x) - [\![\psi]\!](y) \geq \Delta - \delta = f(x) - f(y) - \delta$. Then $f(x) - [\![\psi]\!](x) + r \leq f(y) - [\![\psi]\!](y) + \delta + [\![\psi]\!](y) - f(y) = \delta$.

Whenever $[\![\psi]\!](y) \leq f(y)$ we define $\varphi = \min(\psi \oplus r, s)$ where $r \in [f(y) - [\![\psi]\!](y) - \delta, f(y) - [\![\psi]\!](y)]$ and s is chosen as above. Note that $\oplus$ with $a \oplus b = \min\{a + b, \top\}$ is not an operator of the logic, but it can be simulated by $\neg(\neg a \ominus b)$. □

Finally we can show under which conditions the inequality $d_\alpha \leq d^L_\alpha$ holds.

Theorem 5.3.9

If the fixpoint d_α is reached in ω steps, it holds that $d_\alpha \leq d^L_\alpha$.

Proof: We will show that d^L is a prefixpoint of the equation $d = d^{\uparrow\Gamma} \circ (\alpha \times \alpha)$. Since the fixpoint d_α is reached in ω steps (i.e., $d_\alpha = \sup_{i<\omega} d_i$) it suffices to show $d_i \leq d^L_i$ by induction on i:

- $i = 0$: The inequality is clearly satisfied since $d_0 = 0$.

- $i \to i+1$: Let $x, y \in X$. We first observe that for every $\varepsilon > 0$ and $f\colon (X, d^L_i) \xrightarrow{1} ([0, \top], de)$ there exists a modal formula ψ with $md(\psi) \leq i$ and $d^\infty_e(f, [\![\psi]\!]) \leq \varepsilon$ (cf. Proposition 5.3.8). With the triangle inequality we have

$$\begin{aligned} & d_e(\tilde{F}_\gamma f(\alpha(x)), \tilde{F}_\gamma f(\alpha(y))) \\ \leq\ & d_e(\tilde{F}_\gamma f(\alpha(x)), \tilde{F}_\gamma [\![\psi]\!](\alpha(x))) + d_e(\tilde{F}_\gamma [\![\psi]\!](\alpha(x)), \tilde{F}_\gamma [\![\psi]\!](\alpha(y))) \\ & \quad + d_e(\tilde{F}_\gamma [\![\psi]\!](\alpha(y)), \tilde{F}_\gamma f(\alpha(y))) \end{aligned}$$

  Non-expansiveness of the predicate lifting with respect to the supremum metric (cf. Definition 5.2.18) implies that $d_e(\tilde{F}_\gamma f(\alpha(x)), \tilde{F}_\gamma [\![\psi]\!](\alpha(x))) \leq \varepsilon$, analogously $d_e(\tilde{F}_\gamma [\![\psi]\!](\alpha(y)), \tilde{F}_\gamma f(\alpha(y))) \leq \varepsilon$. Combined we obtain

$$d_e(\tilde{F}_\gamma f(\alpha(x)), \tilde{F}_\gamma f(\alpha(y))) \leq d_e(\tilde{F}_\gamma [\![\psi]\!](\alpha(x)), \tilde{F}_\gamma [\![\psi]\!](\alpha(y))) + 2 \cdot \varepsilon.$$

  This means that

$$\begin{aligned} & \sup\{d_e(\tilde{F}_\gamma f(\alpha(x)), \tilde{F}_\gamma f(\alpha(y))) \mid\ f\colon (X, d^L_i) \xrightarrow{1} ([0, \top], de), \gamma \in \Gamma\} \\ \leq\ & \sup\{d_e(\tilde{F}_\gamma [\![\psi]\!](\alpha(x)), \tilde{F}_\gamma [\![\psi]\!](\alpha(y))) \mid md(\psi) \leq i, \gamma \in \Gamma\} + 2 \cdot \varepsilon \end{aligned}$$

and since this holds for every $\varepsilon > 0$, the two suprema are equal.

$$
\begin{aligned}
& d_{i+1}(x,y) \\
= \; & \sup\{d_e(\tilde{F}_\gamma f(\alpha(x)), \tilde{F}_\gamma f(\alpha(y))) \mid f\colon (X,d_i) \xrightarrow{1} ([0,\top], de), \gamma \in \Gamma\} \\
& \text{(by the induction hypothesis } d_i \leq d_i^L) \\
\leq \; & \sup\{d_e(\tilde{F}_\gamma f(\alpha(x)), \tilde{F}_\gamma f(\alpha(y))) \mid \; f\colon (X,d_i^L) \xrightarrow{1} ([0,\top], de), \gamma \in \Gamma\} \\
= \; & \sup\{d_e(\tilde{F}_\gamma [\![\psi]\!](\alpha(x)), \tilde{F}_\gamma [\![\psi]\!](\alpha(y))) \mid md(\psi) \leq i, \gamma \in \Gamma\} \\
= \; & \sup\{d_e([\![[\gamma]\psi]\!](x), [\![[\gamma]\psi]\!](y)) \mid md(\psi) \leq i, \gamma \in \Gamma\} \\
\leq \; & \sup\{d_e([\![\varphi]\!](x), [\![\varphi]\!](y))) \mid md(\varphi) \leq i+1\} \\
= \; & d_{i+1}^L(x,y)
\end{aligned}
$$

□

Demonstrating under which conditions both directions $d_\alpha^L \leq d_\alpha$ and $d_\alpha \leq d_\alpha^L$ hold, where the preservation of totally boundedness via lifting plays a central role, we established a quantitative Hennessy-Milner theorem as illustrated in Figure 5.8.

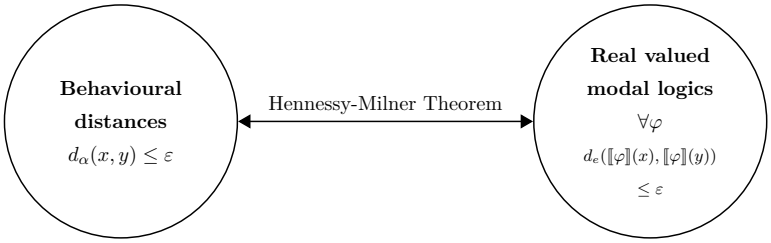


Figure 5.8: Based upon Section 5.2.2 (i.e. the requirements for the modalities from Definition 5.2.18) and due to Propositions 5.3.7 & 5.3.8 inspired by Ash's lemma [Ash72] we have $d_\alpha = d_\alpha^L$.

Finally, in the next section we move to the third point of our agenda: the game theoretical view on behavioural distance.

5.4 Behavioural Distance Games over Set

We will now present the two-player game characterizing the behavioural distance between two states. As described in Section 3.3, in the classical case, the spoiler choses an arbitrary characteristic function over the state set X and the aim of the duplicator is to show that two states are bisimilar (i.e. their behavioural distance is zero). In a quantitative version of the game the moves are given by real-valued functions over X.

5.4.1 Formulation of the Game

The roles of **S** and **D** are similar to those in the classical game (see Section 3.3), where **D** wants to defend the statement that the distance of two states $x, y \in X$ in a coalgebra α is bounded by $\varepsilon \in [0, \top]$, i.e., $d_\alpha(x, y) \leq \varepsilon$ while **S** wants to disprove this claim.

- **Initial situation:** Given a coalgebra $\alpha\colon X \to FX$, we start with (x, y, ε) where $x, y \in X$ and $\varepsilon \in [0, \top]$.
- **Step 1: S** chooses $s \in \{x, y\}$ and a real-valued predicate $p_1\colon X \to [0, \top]$.
- **Step 2: D** takes $t \in \{x, y\}\backslash\{s\}$ if $x \neq y$ and $t = s$ otherwise. In addition, **D** has to answer with a predicate $p_2\colon X \to [0, \top]$, which satisfies

$$d_\ominus(\tilde{F}_\gamma p_1(\alpha(s)), \tilde{F}_\gamma p_2(\alpha(t))) \leq \varepsilon \text{ for all } \gamma \in \Gamma.$$

- **Step 3: S** chooses p_i with $i \in \{1, 2\}$ and some state $x' \in X$.
- **Step 4: D** chooses some state $y' \in X$ with $p_i(x') \leq p_j(y')$ where $j \neq i$
- **Next round:** (x', y', ε') with $\varepsilon' = p_j(y') - p_i(x')$.

After one round the game continues with the initial step, but now **D** tries to show that $d_\alpha(x', y') \leq \varepsilon'$. **D** wins if the game continues forever. In the other case, e.g., **D** has no move at Step 2 or Step 4, **S** wins. Note that the usage of the directed metric allows **D** to *do more*.

The game distance of two states is defined as follows.

Definition 5.4.1: Game distance

Let $\alpha\colon X \to FX$ be a coalgebra and let $x, y \in X$. We define the game distance of x, y as

$$d_\alpha^G(x, y) = \inf\{\varepsilon \mid \mathbf{D} \text{ has a winning strategy for } (x, y, \varepsilon)\}.$$

We now prove that the behavioural distance and the game distance coincide. We first show that d_α^G is indeed a pseudometric.

Proposition 5.4.2

The game distance d_α^G is a pseudometric.

Proof: Here we need to review the three conditions of a pseudometric. Assume that $x, y, z \in X$.

1. *Reflexivity:* we show that there is a strategy for **D** for $(x, x, 0)$ and hence $d_\alpha^G(x, x) = 0$.

   Assume that **S** chooses x and some p_1, then **D** answers with $p_2 = p_1$ and also chooses x. In this case $d_\ominus(\tilde{F}_\gamma p_1(\alpha(x)), \tilde{F}_\gamma p_2(\alpha(x))) = 0$ for every $\gamma \in \Gamma$, due to reflexivity of $d_\ominus$. **S** then chooses some x' and **D** can choose the same x', since $p_2(x') = p_1(x')$. Furthermore $\varepsilon' = p_2(x') - p_1(x') = 0$ and the game continues with $(x', x', 0)$, i.e., in a situation analogous to the previous one.

2. *Symmetry:* we show that $d_\alpha^G(x, y) = d_\alpha^G(y, x)$.

   This is true since **S** can choose either x or y and **D** is then forced to play with the other state. Hence the roles of x and y can be exchanged. Hence

   $$\begin{aligned} d_\alpha^G(x, y) &= \inf\{\varepsilon \mid \mathbf{D} \text{ has a winning strategy for } (x, y, \varepsilon)\} \\ &= \inf\{\varepsilon \mid \mathbf{D} \text{ has a winning strategy for } (y, x, \varepsilon)\} = d_\alpha^G(y, x) \end{aligned}$$

3. *Triangle inequality:* We prove that $d_\alpha^G(x, z) \leq d_\alpha^G(x, y) + d_\alpha^G(y, z)$. We do this by constructing a winning strategy for **D** for $(x, z, \theta + \delta)$ from strategies for $(x, y, \theta), (y, z, \delta)$.

   Now, assume that in the $(x, z, \theta + \delta)$-game **S** chooses p_i, $i \in \{1, 3\}$ and x' in Step 1. We can assume that $i = 1$ (the other case is analogous).

   If **S** chooses p_1, x in the (x, y, θ)-game, **D** can play according to her strategy and choose p_2 and y with $d_\ominus(\tilde{F}_\gamma p_1(\alpha(x)), \tilde{F}_\gamma p_2(\alpha(y))) \leq \theta$ for all γ. Now assume that in the (y, z, δ)-game **S** chooses y and p_2. Then **D** has an answering move p_3 and z with $d_\ominus(\tilde{F}_\gamma p_2(\alpha(y)), \tilde{F}_\gamma p_3(\alpha(z))) \leq \delta$. Combined we have, due to the triangle inequality:

   $$\begin{aligned} & d_\ominus(\tilde{F}_\gamma p_1(\alpha(x)), \tilde{F}_\gamma p_3(\alpha(z))) \\ \leq\ & d_\ominus(\tilde{F}_\gamma p_1(\alpha(x)), \tilde{F}_\gamma p_2(\alpha(y))) + d_\ominus(\tilde{F}_\gamma p_2(\alpha(y)), \tilde{F}_\gamma p_3(\alpha(z))) \leq \theta + \delta \end{aligned}$$

   Hence **D** will answer **S**'s move in the $(x, z, \theta + \delta)$-game by p_3, z in Step 2.

   Now assume that **S** chooses p_i with $i \in \{1, 3\}$ and some state $x' \in X$ (again we assume without loss of generality that $i = 1$) in Step 3. Then, following the strategy for (x, y, θ), **D** chooses y' with $p_1(x') \leq p_2(y')$. Assume that **S** would choose p_2 and y' in the (y, z, δ)-game. Following her strategy **D** would choose z' with $p_2(y') \leq p_3(z')$. Combined, we have $p_1(x') \leq p_3(z')$.

Hence **D** will answer **S**'s move by z' in Step 4.

Furthermore $p_3(z') - p_1(x') = \underbrace{p_3(z') - p_2(y')}_{=\theta'} + \underbrace{p_2(y') - p_1(x')}_{\delta'}$. Hence we are in the situation $(x', z', \theta' + \delta')$ with existing winning strategies for the (x', y', θ')- and (y', z', δ')-games.

Finally, we obtain

$$\begin{aligned} \varepsilon_{x,z} &= \inf\{\varepsilon \mid \mathbf{D} \text{ has a winning strategy for } (x, z, \varepsilon)\} \\ &\leq \inf\{\theta + \delta \mid \mathbf{D} \text{ has a winning strategy for } (x, y, \theta) \text{ and } (y, z, \delta)\} \\ &= d_\alpha^G(x, y) + d_\alpha^G(y, z) \end{aligned}$$

□

Next we show that the game distance is always bounded by the behavioural distance.

The strategy for **D** can be straightforwardly explained whenever X is finite. In particular we want to show that whenever $d_\alpha(x, y) \leq \varepsilon$, then **D** has a winnning strategy for (x, y, ε). Assume that **S** chooses $s \in \{x, y\}$ with $p_1 \colon X \to [0, \top]$. In this case **D** chooses p_2 with $p_2(z) = \sup\{p_1(u) - d_\alpha(u, z) \mid u \in X\}$ in Step 2. From Lemma 5.2.4 we know that $p_1 \leq p_2$ and p_2 is non-expansive. It can be shown that this choice satisfies $d_\ominus(\tilde{F}_\gamma p_1(\alpha(s)), \tilde{F}_\gamma p_2(\alpha(t))) \leq \varepsilon$ for all $\gamma \in \Gamma$. Now S chooses i and $x' \in X$ in Step 3. Then either $i = 1$ and **D** can choose $y' = x'$ in Step 4 and the game continues with x', y' and $\varepsilon' = p_2(y') - p_1(x') \geq 0$. Or $i = 2$ and **D** can choose y' such that $p_1(y') - d_\alpha(y', x') = p_2(x')$ (the supremum is reached since X is finite). This means that $p_1(y') \geq p_2(x')$ and $\varepsilon' = p_1(y') - p_2(x') = d_\alpha(x', y')$. In both cases, the game can continue.

If we compare this to the behavioural equivalence game, in which the duplicator's winning strategy is constructed as the closer p_2 under behavioural equivalence of p_1 (see Theorem 3.3.2), the winning strategy here is based on the behavioural pseudometric d_α. As described above, the spoiler chooses an arbitrary real-valued function $p_1 : X \to [0, \top]$ over X and similar to the construction of p_2 in the classical case, the duplicator wants to construct the least function p_2 over p_1 which does not increase the behavioural distance.

Theorem 5.4.3

It holds that $d_\alpha^G \leq d_\alpha$.

Proof: Let $x, y \in X$. We show for all $\delta > 0$ that whenever $d_\alpha(x, y) \leq \varepsilon - \delta$, then **D** can win the (x, y, ε)-game. This implies in particular that **D** has a winning strategy

for (x, y, ε) with $\varepsilon = d_\alpha(x, y) + \delta$. Since $d^G_\alpha(x, y)$ is the infimum of all such ε we have $d^G_\alpha(x, y) \leq d_\alpha(x, y) + \delta$ and since this holds for all $\delta > 0$ we get the desired inequality.

Assume that **S** chooses $s \in \{x, y\}$ with $p_1 \colon X \to [0, \top]$ in Step 1. In this case **D** chooses in Step 2 $p_2 = \max\{p_1, h - \frac{\delta}{2}\}$ where $h(z) = \sup\{p_1(u) - d_\alpha(u, z) \mid u \in X\}$. Lemma 5.2.4 implies that $p_1 \leq h$ and $h \colon (X, d_\alpha) \xrightarrow{1} ([0, \top], d_e)$ is non-expansive. Clearly $p_1 \leq p_2$.

The inequality $p_1 - p_2 \leq 0$, i.e. $d^\infty_\ominus(p_1, p_2) \leq 0$, and the non-expansiveness of the predicate lifting (cf. Definition 5.2.18) implies $d_\ominus(\tilde{F}_\gamma p_1(\alpha(s)), \tilde{F}_\gamma p_2(\alpha(s))) \leq 0$.

Furthermore we observe that $p_2 \colon (X, d_\alpha + \frac{\delta}{2}) \xrightarrow{1} ([0, \top], d_e)$: Let $u, z \in X$ and we show that $p_2(z) - p_2(u) \leq d_\alpha(u, z) + \frac{\delta}{2}$. Whenever $p_2(u) = p_1(u)$ we have that $p_1(u) \geq h(u) - \frac{\delta}{2}$ and hence:

$$\begin{aligned} p_2(z) - p_2(u) &= \max\left\{p_1(z), h(z) - \frac{\delta}{2}\right\} - p_1(u) \\ &= \max\left\{p_1(z) - p_1(u), h(z) - \frac{\delta}{2} - p_1(u)\right\} \\ &\leq \max\left\{h(z) - h(u) + \frac{\delta}{2}, h(z) - \frac{\delta}{2} - h(u) + \frac{\delta}{2}\right\} \\ &\leq d_\alpha(u, z) + \frac{\delta}{2} \end{aligned}$$

Analogously, whenever $p_2(u) = h(u) - \frac{\delta}{2}$ we have:

$$\begin{aligned} p_2(z) - p_2(u) &= \max\left\{p_1(z), h(z) - \frac{\delta}{2}\right\} - h(u) + \frac{\delta}{2} \\ &= \max\left\{p_1(z) - h(u) + \frac{\delta}{2}, h(z) - h(u)\right\} \\ &\leq \max\left\{h(z) - h(u) + \frac{\delta}{2}, h(z) - h(u)\right\} \\ &\leq d_\alpha(u, z) + \frac{\delta}{2} \end{aligned}$$

Furthermore we know from $d_\ominus \leq d_e$ and the non-expansiveness of γ, p_2 (and hence of $\tilde{F}_\gamma p_2$) that

$$\begin{aligned} d_\ominus(\tilde{F}_\gamma p_2(\alpha(s)), \tilde{F}_\gamma p_2(\alpha(t))) &\leq d_e(\tilde{F}_\gamma p_2(\alpha(s)), \tilde{F}_\gamma p_2(\alpha(t))) \\ &\leq \left(d_\alpha + \frac{\delta}{2}\right)^{\uparrow\Gamma}(\alpha(s), \alpha(t)) \end{aligned}$$

From Proposition 5.2.20 it follows that

$$\left(d_\alpha + \frac{\delta}{2}\right)^{\uparrow\Gamma}(\alpha(s), \alpha(t)) \leq d^{\uparrow\Gamma}_\alpha(\alpha(s), \alpha(t)) + \delta = d_\alpha(s, t) + \delta \leq \ \varepsilon - \delta + \delta = \varepsilon$$

Now the triangle inequality implies

$$\begin{aligned} & d_{\ominus}(\tilde{F}_\gamma p_1(\alpha(s)), \tilde{F}_\gamma p_2(\alpha(t))) \\ \leq \; & \underbrace{d_{\ominus}(\tilde{F}_\gamma p_1(\alpha(s)), \tilde{F}_\gamma p_2(\alpha(s)))}_{=0} + \underbrace{d_{\ominus}(\tilde{F}_\gamma p_2(\alpha(s)), \tilde{F}_\gamma p_2(\alpha(t)))}_{\leq \varepsilon} \leq \varepsilon, \end{aligned}$$

hence p_2 is a valid choice for **D**. **S** now chooses i and $x' \in X$ in Step 3. We distinguish the following cases:

- Case $i = 1$: **D** chooses $y' = x'$ in Step 4. We have $p_1(x') \leq p_2(x') = p_2(y')$ and $\varepsilon' = p_2(x') - p_1(y') \geq 0 = d_\alpha(x', y')$, a situation from which **D** has a winning strategy by copying all moves of **S**.

- Case $i = 2$: If $p_1(x') = p_2(x')$ **D** can again choose $y' = x'$ in Step 4 and we are in a situation analogous to Case $i = 1$ above.

  Otherwise $p_2(x') = h(x') - \frac{\delta}{2}$ and there exists a $y' \in X$ such that $h(x') \leq p_1(y') - d_\alpha(x', y') + \frac{\delta}{4}$, which is chosen by **D** in Step 4. It holds that $p_2(x') = h(x') - \frac{\delta}{2} \leq p_1(y') - d_\alpha(x', y') + \frac{\delta}{4} - \frac{\delta}{2} = p_1(y') - d_\alpha(x', y') - \frac{\delta}{4} \leq p_1(y')$ and $\varepsilon' = p_1(y') - p_2(x') \geq d_\alpha(x', y') + \frac{\delta}{4}$.

  Now the game continues with the next round (x', y', ε'), where $d_\alpha(x', y') \leq \varepsilon' - \frac{\delta}{4}$ and so **D** has a winning strategy.

□

Example 5.4.4

Imagine the initial game situation (x, y, ε) for our example in Figure 5.5 and **S** chooses x with predicate $p_1(4) = 1$ and zero for all remaining states.
Now **D** follows the strategy above and plays a predicate p_2 with $p_2(4) = p_2(5) = p_2(7) = 1$ and zero for all other states. Since $5, 7$ are at distance 0 to 4, they are now mapped to 1 as well. Since in particular 4 and 7 are mapped to 1, we obtain $d_{\ominus}(\tilde{\mathcal{D}}_{\bar{\gamma}_\mathcal{D}} p_1(\alpha(x)), \tilde{\mathcal{D}}_{\bar{\gamma}_\mathcal{D}} p_2(\alpha(y))) = d_{\ominus}(\frac{1}{4}, \frac{1}{2} + \varepsilon) = 0 \leq \varepsilon$ (we obtain the same value for $\gamma_\bullet$). Note again that **D** must be allowed to do *more* than **S**.
Now the winning strategy for **D** is obvious: if **S** picks a terminating state x' and p_i, **D** can also pick a terminating state y' and p_j with $p_j(y') - p_i(x') = 0$ (similarly for non-terminating states). We then end up in $(x', y', 0)$ where x', y' are behaviourally equivalent.
If **S** had instead chosen y a predicate p_1 with $p_1(7) = 1$ and zero for all other states, **D** would choose the same predicate p_2 with $d_{\ominus}(\tilde{\mathcal{D}}_{\bar{\gamma}_\mathcal{D}} p_1(\alpha(y)), \tilde{\mathcal{D}}_{\bar{\gamma}_\mathcal{D}} p_2(\alpha(x))) =$

$d_{\ominus}(\frac{1}{2}+\varepsilon, \frac{1}{2}) = \varepsilon$.

We now demonstrate that in the case of infinite branching, the construction of the winning strategy for the **D** is not as simple as described before.

Example 5.4.5

Consider the coalgebra $\alpha\colon X \to \mathcal{D}X + 1$ in Figure 5.9 on the state space $X = \{y, y_0, x, x_1, x_2, \dots\}$, where the probability of going from x to x_i is $\alpha(x)(x_i) = \frac{1}{2^i}$. For both states x, y the probability to terminate is 1 and hence $x \sim y$. Now imagine that **S** selects x and the real-valued predicate p_1 with $p_1(x_i) = 1 - \frac{1}{2^i}$ and $p_1(x) = 0$. If we would construct the predicate for **D** as above, via $p_2(z) = \sup\{p_1(u) - d_\alpha(u, z) \mid u \in X\}$, this would yield $p_2(y_0) = 1$ since the distance of all terminating states is 0.

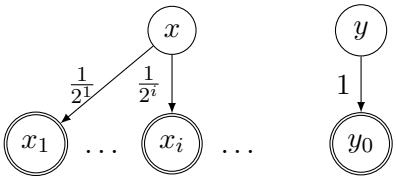


Figure 5.9: Probabilistic transition system for the functor $FX = \mathcal{D}X + 1$, where X is infinite.

Then **S** chooses $x' = y_0$ and p_2 in Step 3 and **D** has no available state y' with which to answer in Step 4. If $y' = x_i$, then $p_1(x_i) = 1 - \frac{1}{2^i} < 1 = p_2(x')$, otherwise $p_1(y') = 0 < 1$.
In fact, **D** has no winning strategy for $\varepsilon = 0$, but we can show that there is a winning strategy for every $\varepsilon > 0$ (since **D** can play a predicate that is below p_2, but at distance ε). Since the game distance is defined as the infinum over all such ε's it still holds that $d^G_\alpha(x, y) = 0$.

Finally, we can show the other inequality.

Theorem 5.4.6

It holds that $d_\alpha \leq d^G_\alpha$.

Proof: Due to Proposition 5.4.2 we know that d^G_α is a pseudometric. We will now show that it is a prefix-point of the defining fixpoint equation, i.e. $d^G_\alpha \geq (d^G_\alpha)^{\uparrow\Gamma} \circ (\alpha \times \alpha)$. Since d_α is the smallest (pre-)fixpoint, this implies that $d_\alpha \leq d^G_\alpha$.

Let $x, y \in X$. It holds that

$$\begin{aligned} & (d_\alpha^G)^{\uparrow\Gamma}(\alpha(x), \alpha(y)) \\ = \; & \sup\{d_e(\tilde{F}_\gamma f(\alpha(x)), \tilde{F}_\gamma f(\alpha(y))) \mid f\colon (X, d_\alpha^G) \xrightarrow{1} ([0, \top], d_e), \gamma \in \Gamma\} \end{aligned}$$

We will show that $d_\ominus(\tilde{F}_\gamma f(\alpha(x)), \tilde{F}_\gamma f(\alpha(y))) \leq d_\alpha^G(x, y)$ and the same holds analogously if the roles of x, y are reversed. This means that every element in the sup is bounded by $d_\alpha^G(x, y)$, from which the inquality follows. It is sufficient to show that $d_\ominus(\tilde{F}_\gamma f(\alpha(x)), \tilde{F}_\gamma f(\alpha(y))) \leq \varepsilon$ whenever $\mathbf{D}$ has a winning strategy for (x, y, ε).

So, assuming that $\mathbf{S}$ chooses x and f at Step 1, we know that $\mathbf{D}$ can at Step 2 choose a real-valued predicate p_2, such that $d_\ominus(\tilde{F}_\gamma f(\alpha(x)), \tilde{F}_\gamma p_2(\alpha(y))) \leq \varepsilon$ for all $\gamma \in \Gamma$.

We infer from the triangle inequality that

$$\begin{aligned} & d_\ominus(\tilde{F}_\gamma f(\alpha(x)), \tilde{F}_\gamma f(\alpha(y))) \\ \leq \; & d_\ominus(\tilde{F}_\gamma f(\alpha(x)), \tilde{F}_\gamma p_2(\alpha(y))) + d_\ominus(\tilde{F}_\gamma p_2(\alpha(y)), \tilde{F}_\gamma f(\alpha(y))). \end{aligned}$$

The first summand is smaller or equal than ε and it remains to show that the second summand equals 0. We show by contradiction that $p_2 \leq f$ and thus we obtain 0 via the non-expansiveness of the predicate lifting with respect to $d_\ominus^\infty$ (cf. Definition 5.2.18). So assume that $p_2 \not\leq f$ and we argue that $\mathbf{D}$ can not win. There is a state x' such that $p_2(x') > f(x')$. $\mathbf{S}$ can then at Step 3 choose p_2 and x'. Now $\mathbf{D}$ chooses at Step 4 y' with $p_2(x') \leq f(y')$ and $\varepsilon' = f(y') - p_2(x')$. In that case $\mathbf{D}$ has no winning strategy: $f(y') - f(x') \leq d_\alpha^G(x', y')$ since f is non-expansive and $f(x') - p_2(x') < 0$, which implies $\varepsilon' = f(y') - f(x') + f(x') - p_2(x') < d_\alpha^G(x', y') + 0 = d_\alpha^G(x', y')$. And this is a contradiction, since $\mathbf{D}$ does not have a winning strategy for (x', y', ε'). □

Analogous as with the classical game in Section 3.3.2 we consider the relation between distinguishing formulas and the winning strategies of the spoiler.

5.4.2 Spoiler Strategy for the Metric Case

The strategy for $\mathbf{S}$ for (x, y, ε) can be derived from a modal formula φ with $d_\ominus([\![\varphi]\!](x), [\![\varphi]\!](y)) > \varepsilon$. If $\varepsilon < d_\alpha(x, y) = \sup\{d_e([\![\varphi]\!](x), [\![\varphi]\!](y)) \mid \varphi\}$, such a formula must exist (since we can use negation to switch x, y if necessary). The spoiler strategy is defined over the structure of φ:

- $\varphi = \top$: this case can not occur.
- $\varphi = [\gamma]\psi$: $\mathbf{S}$ chooses x, $p_1 = [\![\psi]\!]$ at Step 1. After $\mathbf{D}$ has chosen y, p_2 at Step 2, we can observe that $p_2 \not\leq [\![\psi]\!]$ (see proof of Theorem 5.4.7). Now in Step 3

S chooses p_2 and x' such $p_2(x') > [\![\psi]\!](x')$. Now **D** needs to choose y' such that $[\![\psi]\!](y') \geq p_2(x')$ in Step 4 and $\varepsilon' = [\![\psi]\!](y') - p_2(x') < [\![\psi]\!](y') - [\![\psi]\!](x') = d_e([\![\psi]\!](x'), [\![\psi]\!](y'))$ and so the game continues in the situation (x', y', ε') with the formula ψ.

- $\varphi = \min(\psi, \psi')$: In this case either

$$d_e([\![\psi]\!](x), [\![\psi]\!](y)) > \varepsilon \text{ or } d_e([\![\psi']\!](x), [\![\psi']\!](y)) > \varepsilon$$

  (cf. Lemma 5.2.21(3)) and **S** picks ψ or ψ' accordingly.

- $\varphi = \neg\psi$: In this case **S** takes ψ, since

$$d_e([\![\psi]\!](x), [\![\psi]\!](y)) = d_e([\![\varphi]\!](x), [\![\varphi]\!](y)) > \varepsilon.$$

- $\varphi = \psi \ominus q$: In this case

$$d_e([\![\psi]\!](x), [\![\psi]\!](y)) \geq d_e([\![\varphi]\!](x), [\![\varphi]\!](y)) > \varepsilon$$

  (cf. Lemma 5.2.21(1)) and hence **S** takes ψ.

It can be shown that this strategy is indeed correct.

Theorem 5.4.7

Assume that $\alpha\colon X \to FX$ is a coalgebra. Let φ be a formula with $d_e([\![\varphi]\!](x), [\![\varphi]\!](y)) > \varepsilon$. Then the spoiler strategy described above is winning for **S** in the situation (x, y, ε).

Proof: Each step described in the strategy yields a smaller formula by structural induction. Hence the game will eventually terminate.

We only have to consider the case $\varphi = [\gamma]\psi$ in more detail and to show (by contradiction) that **S** can make a valid move in Step 3 by proving that the predicate p_2 chosen by **D** in Step 2 must satisfy $p_2 \not\leq [\![\psi]\!]$.

Hence assume that $p_2 \leq [\![\psi]\!]$. First observe that

$$d_\ominus(\tilde{F}_\gamma p_2(\alpha(y)), \tilde{F}_\gamma [\![\psi]\!](\alpha(y))) = 0$$

using the non-expansiveness of the predicate lifting wrt. $d^\infty_\ominus$ (cf. Definition 5.2.18). Using the triangle inequality, we have

$$\begin{aligned}
& d_\ominus(\tilde{F}_\gamma [\![\psi]\!](\alpha(x)), \tilde{F}_\gamma p_2(\alpha(y))) \\
\geq\ & d_\ominus(\tilde{F}_\gamma [\![\psi]\!](\alpha(x)), \tilde{F}_\gamma [\![\psi]\!](\alpha(y))) - \underbrace{d_\ominus(\tilde{F}_\gamma p_2(\alpha(y)), \tilde{F}_\gamma [\![\psi]\!](\alpha(y)))}_{=0} \\
=\ & d_\ominus([\![\varphi]\!](x), [\![\varphi]\!](y)) \\
>\ & \varepsilon
\end{aligned}$$

which means that **D** can not play p_2 in Step 2 and we have a contradiction. □

Note that Theorem 5.4.6 is not a direct corollary of this theorem, since here we require that a formula φ with $d_e([\![\varphi]\!](x), [\![\varphi]\!](y)) > \varepsilon$ exists, which is not necessarily true in scenarios where the fixpoint iteration does not terminate in ω steps.

Example 5.4.8

We will show how **S** can construct a winning strategy for $(x, y, \frac{\varepsilon}{2})$ based on the formula $\varphi = [\bar{\gamma}_{\mathcal{D}}][\gamma_\bullet]\top$ from Example 5.3.3. The transition system is shown in Figure 5.5.
It holds that $d_\ominus([\![\varphi]\!](y), [\![\varphi]\!](x)) = \varepsilon > \frac{\varepsilon}{2}$. **S** plays y and $p_1 = [\![[\gamma_\bullet]\top]\!]$ which, due to the definition of $\gamma_\bullet$ equals 1 on terminating states and zero on non-terminating states. Now $\bar{\gamma}_{\mathcal{D}}(\mathcal{D}p_1(\alpha(y))) = \frac{1}{2} + \varepsilon$, so **D** must play in such a way that $\bar{\gamma}_{\mathcal{D}}(\mathcal{D}p_2(\alpha(x))) \geq \frac{1}{2} + \frac{\varepsilon}{2}$. This can only be achieved by setting $p_2(3) = \varepsilon$ (or to a larger value). Now **S** chooses p_2, $x' = 3$ and **D** can only take p_1 and either 4, 5 or 7 as y'. In each case we obtain $\varepsilon' = p_1(y') - p_2(x') = 1 - \varepsilon < 1 = d_e(0, 1) = d_e([\![[\gamma_\bullet]\top]\!](x'), [\![[\gamma_\bullet]\top]\!](y'))$.
The spoiler continues to follow his strategy and plays x', $p_1 = [\![\top]\!]$ in the next step, which is successful, since y' is a terminating state and x' is not.

5.5 Conclusion and Discussion

After performing the research introduced in Chapter 3, the first thing we noticed was that a quantitative and general version of the triad had not been presented so far. Besides a quantitative version of a coalgebraic game, there have also been no generical results in the field of logic.

We summarize our contributions where the main parts are illustrated in Figure 5.10:
▷ First of all, inspired by the work in [BB+14] we improved the Kantorovich lifting via a parametrization based on a finite set Λ of evaluation maps. The new version allows to capture additional examples, without going via the somewhat cumbersome multifunctor lifting described in [BB+14].
▷ Secondly, we present a real-valued coalgebraic modal logic and give a Hennessy-Milner theorem for the coalgebraic setting.

After the adaptation of Ash's lemma [Ash72] and proving the preservation of totally boundedness via lifting (Propositions 5.3.7,5.3.6), the quantitative Hennessy-Milner theorem *naturally* follows from the parametrization based on a set Λ of evaluation

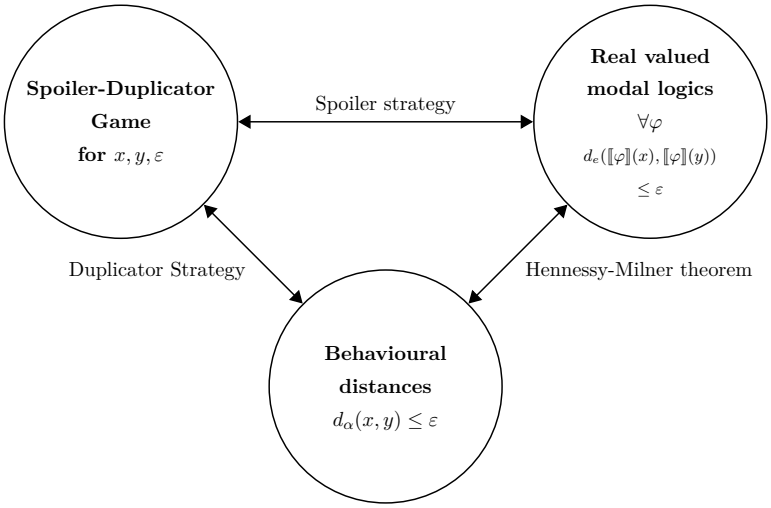


Figure 5.10: Given a coalgebra $\alpha : X \to FX$ we have three ways to characterize quantitative behaviour: behavioural distance d_α, modal logics , and a game-theoretical view.

maps, where the requirements are listed in Definition 5.2.18. The requirement of monotonicity is replaced by local non-expansiveness in the metric case. The fact that monotonicity for partial orders generalizes to non-expansiveness for directed metrics has already been discussed in [TR98].

▷ Thirdly, we worked out a quantitative game version. For this, we follow the outline of Section 3.3, which treats the classical case, with some variations. An important difference is the fact that the metric case is parameterized over a set Γ of evaluation maps. Note that we actually mimic the variant of the game discussed in Paragraph 3.3.1, where we fix evaluation maps, but omit the requirement of weak pullback preservation. Analogously to the results obtained for the real-valued logic, the requirement of monotonicity is replaced by local non-expansiveness.

Comparison to related work can be found in the introduction and throughout the chapter but at this point we summarize everything and close this chapter by discussing some open points and questions.

Our work is inspired by the Hennessy-Milner theorem for probabilistic transition systems [DG+04] and similar results obtained in [WS+18a] for fuzzy logics, on the way to proving a van Benthem theorem. A generalization of our Hennessy-Milner theorem is published in [WS21].

Behavioural distances from a coalgebraic perspective are studied in [BW06] which presents an algorithm to approximate the distance between two states. Another categorical approach is given by the work in [KK+19], which considers games from a

fibrational perspective which provides an elegant way to capture several interesting examples of bisimulation notions, including behavioural metrics. The so-called *codensitiy lifting* of an endofunctor F along such a fibration is based on [SK+18] and has two parameters: a set of modalities (given as F-algebras) and an observation domain. While this approach allows handling examples such as **Meas**, the work in [KK+19] neither considers codensity bisimilarity on Kleisli categories nor includes any results on modal logics.

The relation between the Kantorovich lifting $d^{\uparrow F}$ and the Wasserstein lifting $d^{\downarrow F}$ based on a single evaluation map is studied in [BB+14; Ker16] and under some mild conditions for the evaluation maps $d^{\downarrow F}$ serves as an upper bound for $d^{\uparrow F}$.

An interesting metric game for probabilistic transition systems on the base of the Wasserstein lifting is presented in [WS+18b], where the main difference lies in the fact, that the duplicator does not imitate the moves of the spoiler. Given two probabilistic systems $(X, \alpha_1), (Y, \alpha_2)$, the initialization of the game (x, y, ε) is analogous to ours, but the duplicator makes the first move like in Baltag's game [Bal99]. Via this move the duplicator has to choose a probability distribution μ (which is a coupling of the two probability measures given by $\alpha_1(x)$ and $\alpha_2(y)$) and some distance function $d' : X \times Y \to [0, 1]$ such that the evaluation based on μ and d' is less or equal the threshold ε. Again similar to the rules by Baltag, the duplicator has to define μ, d' in such a way, that she always can answer to any move $\mu(x', y') > 0$ by the spoiler. Similar to our game, the new round is derived from both moves $(x', y', d'(x', y'))$ [WS+18b].

In fact, the variant of the classical game that uses the lifted order $\leq^F$ is more reminiscent of the Wasserstein lifting for metrics. It is future work to define a variant of the metric game via the Wasserstein lifting (or other liftings of metrics).

Another open question is to prove the Hennessy-Milner theorem for the real-valued logic in the case where the fixpoint is not reached in ω steps. The original variant of the Hennessy-Milner-theorem only holds for finitely-branching transition systems, but this result can be generalized if we allow infinite conjunctions (cf. the logic in Section 5.3). A natural question is whether the same solution is applicable to the metric case, by replacing the min- by an inf-operator (of restricted cardinality κ, as in Section 3.2.3). However, for this it seems necessary to generalize the notion of total boundedness to a new variant where we do not require that the set of *anchors* $\{x_1, \ldots, x_n\}$ of Definition 5.3.5 is finite, but bounded by κ.

A related question is the following: does the Kantorovich lifting preserve completeness of metrics? (A metric space (X, d) is complete if every Cauchy sequence

converges in X.) Furthermore we would like to add ∞ as a possible distance value, as in [BB+14]. However, this can not be integrated so easily, for instance it is unclear how to define negation.

Finally, in the quantitative case it could be interesting to know whether we can use existing efficient algorithms (for the probabilistic case), for instance in order to generate the strategy of the spoiler (see, e.g. [CBW12]).

6 Behavioural Equivalence: Coalgebraic Modal Logic and Games Beyond Set

Trace semantics is one possible notion of behavioural equivalence and known as the coarsest one in the linear-time/branching-time spectrum [Gla01]. As discussed in Example 2.2.2 this notion does not capture the non-deterministic branching of LTS, but sometimes such branchings are side-effects and we are rather interested if two states admit the same traces or words. As introduced in Section 2.3.4 so-called *Kleisli extensions* allow us to capture trace semantics via coalgebraic behaviour equivalences for various branching types [JSS15]. Especially, the work for coalgebraic games in the context of behavioural equivalences is restricted to **Set** as introduced in Chapter 3 and via *Kleisli extensions* we switch to *Kleisli categories*. In order to complete the picture of logical and game-theoretical semantics we study the top and the bottom of the linear and branching time spectrum characterized via coinduction [JSS15].

6.1 Introduction

Most of the work in terms of coalgebraic games and modal logics concerns applications where the state based models live in **Set** [Kup07; KM18; Pat03; Sch08]. Inspired by the goal to lift the level of abstraction for coalgebraic logic and games, various high-level frameworks from the categorical literature can be employed.

We start with the fibrational approach (see Definition 2.3.40) pioneered by Hermida and Jacobs in order to reason about predicates or types from a general point of view [HJ98]. Recall, that fibrations are in one-to-one correspondence with indexed categories (see Section 2.3.5) and in terms of games, $CLat_{\wedge}$-fibrations are considered in [KK+19] which are fibrations restricted to fibers which correspond to a complete lattice and reindexing preserves arbitrary meets. Such a framework allows to capture several notions of bisimulation together with their corresponding game characterizations [KK+19]. Moreover, it turns out that the properties of such a fibrational framework are also quite natural in the context of coalgebraic modal logics [KR20; HKC18].

Furthermore, it has already been observed [JS09] that the natural transformation

induced by polyadic predicate liftings [Sch08] are surrounded by a framework known as dual adjunction. To treat coalgebraic modal logics via dual adjunctions is quite common in the literature [Kli07; BK05; KKP04; JS09] since the contravariant part allows to relate coalgebras living in **Set** or **Meas** with their modal logic [JS09]. In order to deal with F-coalgebras the adjunction is extended via an endofunctor F on the left. On the right you see another endofunctor L which captures modal logics. More concretely, L is used for the syntax of the modal operators and additionally it is required that L has an initial algebra to represent other formulas of the logic (see [JS09; KR20]).

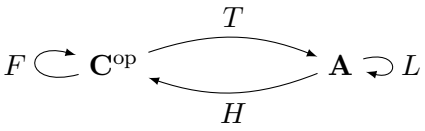


Figure 6.1: A dual adjuntion $H \dashv T$ extended by an F and L-endofunctor [JS09].

Contravariant views are also omnipresent in an approach called indexed category. It is well-known that an indexed category can be transformed into a fibration via the Grothendieck construction and Jacobs observed that predicate liftings are nothing but indexed morphisms. Besides this, Jacobs motivated the indexed category based view to define predicate liftings for monads in the context of Kleisli categories [Jac10].

While the work [KR20] combining fibrations and dual adjunctions introduces a framework which allows to move from binary relations (bisimulations) to quantitative relations (behavioural metrics), we study a way to close the game-theoretical gap with respect to Kleisli categories [JSS15] by answering the following question: *Can we lift the "hidden side effects" via monads known for coalgebraic behavioural equivalence to game-based characterizations?* We are not aware of an approach that generalizes predicate liftings to the setting we study in this chapter, in particular to trace semantics. The papers [KK11; KR16] study generic trace logics and define trace equivalence based on so-called theory maps (or trace maps as in [KK11]) where we define equivalence as the kernel of a coalgebra homomorphism.

Apart from our major objective to set up a game characterization which captures Kleisli categories as well as **Set** we focus on an approach to establish a more abstract definition of logic and games. Our motivation is to provide a game characterization that is *analogous* to the coalgebraic Definition 2.3.17 of behavioural equivalences presented in Chapter 2 and therefore we expand our results from Chapter 3 to trace semantics. Moreover, we extend the ideas based on indexed categories [Jac10] to a *Kleisli extension* of predicate liftings which to our knowledge is new. Summariz-

ing, our contribution is different compared to the games obtained for a fibrational framework [KK+19] which do not address trace semantics or modal logics but brings games for bisimulation and behavioural metrics under one roof.

In the next Section 6.2 we introduce the relevant categorical preliminaries required in this chapter. In Section 6.3 we extend the ideas of Jacobs in the field of Kleisli categories [Jac10; JSS15] and give a recipe to construct predicate liftings for Kleisli categories. Finally, in Section 6.4 we present a framework where we prove general adequacy and expressivity for coalgebraic modal logic and games.

6.2 Foundations

In this chapter we move from logics and games over coalgebras in **Set** to logics and games for coalgebras in the Kleisli category over some **Set**-monad T and we denote such a category with $\mathbf{K}\ell(T)$ as introduced in Definition 2.3.32. The background of this chapter is based on the preliminaries presented in Section 2.3.4 and Section 2.3.5.

In order to talk about elements, we restrict to categories $\mathbf{C}$ such that an *free-forgetful adjunction* exists. This means, that there exists a left adjoint for the forgetful functor $\mathbf{C} \xrightarrow{|\_|} \mathbf{Set}$.

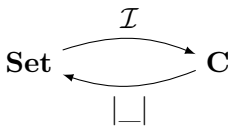


Figure 6.2: For a concrete category $(\mathbf{C}, U)$ over **Set** we assume that there exists an inclusion functor $\mathcal{I}$ from **Set** to our base category $\mathbf{C}$.

Since Kleisli categories are our main motivation, we first present the general construction of an adjunction given by a Kleisli category in Table 6.1. Afterwards, we give an overview in Table 6.2 of the adjunctions used as examples within this chapter.

$\mathcal{I}X = X, \mathcal{I}f = \eta_Y \circ f$	$\lvert X\rvert = TX, \lvert g\rvert = \mu_Y \circ Tg$
for $f : X \to Y, f, X, Y \in \mathbf{Set}$	for $g : X \to Y, g, X, Y \in \mathbf{K}\ell(T)$

Table 6.1: The left $\mathcal{I} : \mathbf{Set} \to \mathbf{K}\ell(T)$ and right $|\_| : \mathbf{K}\ell(T) \to \mathbf{Set}$ adjoint for Kleisli categories induced by a set monad (T, η, μ).

The second monad $\mathcal{M}_{\mathbb{F}}$ in Table 6.2 is derived from the definition presented in Example 2.3.31. Here we restrict to fields instead of semirings to guarantee the decidability of language equivalence (cf. [Sch61]).

$T = \mathcal{P}$	$T = \mathcal{M}_{\mathbb{F}}$
$\mathcal{I}X = X, \mathcal{I}f : X \to \mathcal{P}Y$ $\mathcal{I}f(x) = \{f(x)\}$ for $f : X \to Y \in \mathbf{Set}$ $\lvert X\rvert = \mathcal{P}X, \mathcal{P}X \xrightarrow{\lvert g\rvert} \mathcal{P}Y$ $\lvert g\rvert(X') = \bigcup\{g(x) \mid x \in X'\}, X' \subseteq X$ for $g : X \to Y, g \in \mathbf{K}\ell(\mathcal{P})$	$\mathcal{I}X = X, \mathcal{I}f : X \to \mathcal{M}_{\mathbb{F}}Y$ $\mathcal{I}f(x)(y) = \begin{cases} 1 & f(x) = y \\ 0 & \text{else} \end{cases}$ $\lvert X\rvert = \mathcal{M}_{\mathbb{F}}X, \mathcal{M}_{\mathbb{F}}X \xrightarrow{\lvert g\rvert} \mathcal{M}_{\mathbb{F}}Y$ $\lvert g\rvert(h)(y) = \sum_{h' \in supp_{\Phi}} \Phi(h') \cdot h'(y)$ $h \in \mathcal{M}_{\mathbb{F}}X$ $\mathcal{M}_{\mathbb{F}}g(h) = \Phi \in \mathcal{M}_{\mathbb{F}}(\mathcal{M}_{\mathbb{F}}Y)$ $supp_{\Phi} = \{h' \in \mathcal{M}_{\mathbb{F}}Y \mid \Phi(h') \neq 0\}$ for $g : X \to Y, g \in \mathbf{K}\ell(\mathcal{M}_{\mathbb{F}})$

Table 6.2: Two concrete adjunctions given by the monads: powerset $\mathcal{P}$ and the multiset $\mathcal{M}_{\mathbb{F}}$.

Next we specify for each Kleisli category introduced in Table 6.2 a transition system type and use these applications as running examples.

Example 6.2.1

By following [HJS07; JSS15] we model a non-deterministic automaton (NDA) as a coalgebra living in $\mathbf{C} = \mathbf{Rel} = \mathbf{K}\ell(\mathcal{P})$. Recall the content of Section 2.3.4 that a Kleisli extension $\mathbf{Rel} \xrightarrow{\bar{F}} \mathbf{Rel}$ of $\mathbf{Set} \xrightarrow{F} \mathbf{Set}$ (i.e. $\bar{F} \circ \iota = \iota \circ F$) is in correspondence (see [Mul94, Theorem 2.2] for a general statement) with a distributive law $FT \xrightarrow{\theta} TF$ which is a natural transformation and compatible with the monad structure of T (see Definition 2.3.34 and [Mul94, Theorem 2.2] for a general statement).
Consider $F\_ = A \times \_ + 1$ (where $1 = \{\bullet\}$) with the following distributive law [Jac04, Section 4]:

$$A \times \mathcal{P}X + 1 \xrightarrow{\theta_X} \mathcal{P}(A \times X + 1) \qquad (a, U) \mapsto \{a\} \times U,\ \bullet \mapsto \{\bullet\}.$$

This induces a functor $\mathbf{Rel} \xrightarrow{\bar{F}} \mathbf{Rel}$ which acts on a relation $X \xrightarrow{r} Y$, seen as a Kleisli arrow $X \xrightarrow{r'} \mathcal{P}Y$, as follows: $\bar{F}r = \theta_Y \circ Fr'$. Notice that $\bar{F}$-coalgebras model implicit non-determinism (i.e. this side-effect is hidden to an outside observer as shown in Paragraph 2.3.4) [HJS07], thus behavioural equivalence coincides with *language equivalence* (instead of bisimilarity) in this case.

Example 6.2.2

We consider (linear) weighted automata (LWA) as coalgebras as studied in [JSS15]. LWA are modelled as coalgebras of the endofunctor $\mathcal{M}_{\mathbb{F}}(1 + A \times \_)$, where $\mathbb{F}$ is a field and $\mathcal{M}_{\mathbb{F}}$ is the multiset monad (see Example 2.3.31). We write $x \downarrow_s$ and $x \xrightarrow{a,s'} x'$ whenever $\alpha(x)(\bullet) = s$ and $\alpha(x)(a, x') = s'$ (resp.) for a given LWA $X \xrightarrow{\alpha} \mathcal{M}_{\mathbb{F}}(A \times X + 1)$ in $\mathcal{M}_{\mathbb{F}}$.

Recall that the language of a given LWA α starting from a state $x \in X$ is an inductively defined function $\mathbf{tr}(x)\colon A^\star \to \mathbb{F}$ as follows. Below $a \in A$, $w \in A^\star$, and ε is the empty word.

$$\mathbf{tr}(x)(\varepsilon) = \alpha(x)(\bullet), \qquad \mathbf{tr}(x)(aw) = \sum_{x \xrightarrow{a,s} x'} s \cdot \mathbf{tr}(x')(w).$$

Two states $x, x' \in X$ are *(weighted) language equivalent* iff $\mathbf{tr}(x) = \mathbf{tr}(x')$. This coincides with coalgebraic behavioural equivalence in $\mathbf{K}\ell(\mathcal{M}_{\mathbb{F}})$ (see [JSS15; KK18]). Note that probabilistic automata can be encoded by letting $\mathbb{F} = \mathbb{R}_0$ and restricting the weights to the interval $[0, 1]$.

Analogously to Chapter 3 and Chapter 5 we need to consider predicates and their liftings, since the semantics of modal logic formulas are predicates ($[\![\varphi]\!] : X \to \{0, 1\}$) or correspond to the moves of the players in the games presented within this thesis (see Definition 3.3.1 or Definition 5.4.1).

More precisely, in **Set**, the predicates on a set X are given by characteristic functions over X (i.e. subsets of X). Now given a function $X \xrightarrow{f} Y$ then a predicate p_Y (i.e. a characteristic function over Y) can be transformed into a predicate on X by the pullback operation $f^{-1}(\hat{p_Y}) \subseteq X$ in **Set**. The essential aspect of predicate liftings is the natural way how a subset of X is mapped to a subset of FX (see Definition 3.2.9). In **Set** the naturality follows from the contravariant powerset functor and Jacobs lifts this observation to a high abstract level as introduced in Section 2.3.5.

From now on, we also present our work in the context of indexed categories to get an abstract view on the material: logic and games.

As already mentioned and well-known, indexed morphisms (i.e. predicate liftings) and logic go hand in hand. Regarding the correspondence between games and logic in **Set** (see Figure 3.10) the question arises how indexed morphisms harmonize with games?

Therefore, we consider the following observation on categorical logic where indexed

categories with even more structure are studied [Pit00]. For instance, existential quantification can be interpreted as left adjoint of a functor between fibres [Jac10].

Example 6.2.3

Consider the contravariant powerset functor $\mathbf{Set}^{\mathrm{op}} \xrightarrow{\tilde{\mathcal{Q}}} \mathbf{Cat}$ introduced in Example 2.3.38. One can restrict from **Cat** to **Poset** and obtain for a given function $f : X \to Y \ \in \mathbf{Set}$ the reindexing functor $f^* : (\mathcal{P}Y, \subseteq) \to (\mathcal{P}X, \subseteq)$ that takes subsets $Y' \subseteq Y$ back to subsets of the domain of f (i.e. $f^* = f^{-1}$):

$$f^*(Y') = \{x \mid f(x) \in Y'\}$$

The left adjoint of this functor is the existential quantifier $\exists_f : \mathcal{P}X \to \mathcal{P}Y$. More precisely, $\exists_f$ maps a subset $X' \subseteq X$ to

$$\{y \in Y \mid \exists x \ \in X \text{ s.t. } f(x) = y \text{ and } x \in X'\}$$

Analogously, one can view this from the fibrational perspective based on the Grothendieck construction $\mathbb{E}(\tilde{\mathcal{Q}}) \to \mathbf{Set}$ (see Section 2.3.5) which results in the fibration $\mathbf{Pred} \longrightarrow \mathbf{Set}$ where **Pred** is the category of predicates.
The category **Pred** has as objects pairs (X, p_X) where $p_X : X \to \{0, 1\}$ is a characteristic function over X and the morphisms are given by functions $f : (X, p_X) \to (Y, p_Y)$ such that for all $x \in X$ we have that $p_X(x)$ implies $p_Y(f(x))$.
The elements in $\mathbb{E}(\tilde{\mathcal{Q}})$ are given by tuples (X, X') where $X \in \mathbf{Set}$ and $X' \in \tilde{\mathcal{Q}}X$, and a subset corresponds to a characteristic function $X \to \{0, 1\}$. Thus we get the same elements as in **Pred**. The morphisms in $\mathbb{E}(\tilde{\mathcal{Q}})$ are also given as pairs of functions $(f, p) : (X, X') \to (Y, Y')$ such that $f : X \to Y$ and $p : X' \to f^{-1}[Y']$ with $f^{-1}[Y'] \in \tilde{\mathcal{Q}}X$ [Jac99].

De facto, in Section 6.4.3, this left adjoint is necessary in defining the winning strategy of the duplicator. Based on the fact that an indexed category can be transformed into a fibration by the so called *Grothendieck construction* (see [Jac99, Chapter 9] or Section 2.3.5), we present the following useful result:

Proposition 6.2.4: [Jac99]

An indexed category $\mathbf{C}^{\mathrm{op}} \xrightarrow{\Phi} \mathbf{Cat}$ has the *bifibration* property if, and only if, the reindexing functor f^* (for every $f \in \mathbf{C}$) has a left adjoint $\exists_f$.

Example 6.2.5

The contravariant powerset functor $\mathbf{Set}^{\mathrm{op}} \xrightarrow{\tilde{\mathcal{Q}}} \mathbf{Cat}$ is an example of an indexed category with the bifibration property, i.e. the fibration

$$\mathbf{Pred} \longrightarrow \mathbf{Set}$$

is an opfibration. This is because the reindexing functor f^{-1} (for any function f) has a left adjoint given by the direct image functor (see Example 6.2.3).

From now on, we require that our indexed category has the *bifibration property.*

The next result helps in building new indexed categories having bifibrations from the old ones. We will use this construction later in Section 6.4.3.

Lemma 6.2.6

The bifibration property is preserved by the composition of functors.

Analogously to the foundations of Chapter 3 we need to consider separating sets of predicate liftings (i.e. indexed morphisms) (see Definition 3.2.13 and the third part of the proof of Theorem 3.3.2).

Now that we are also studying a more general view on logic and games we need a more general version of Definition 3.2.13. But therefore, we need to define the evaluation of a predicate by a state since predicates are objects in the categorical sense and states are elements in the set theoretical sence. So we parametrize the evaluation of predicates by a natural transformation $\Phi \xrightarrow{\omega} \tilde{\mathcal{Q}} \circ |\_|$, which transforms an *abstract* predicate into a *concrete* predicate, i.e. as subsets of the underlying state-space. Often ω is the identity natural transformation.

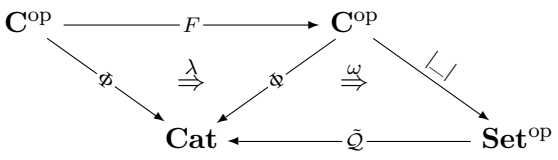


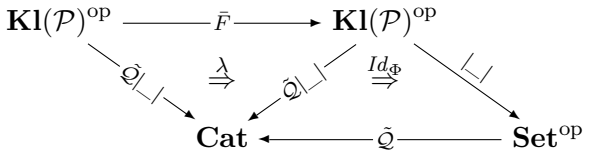
Figure 6.3: Evaluation of predicates in the context of indexed categories.

Definition 6.2.7

A state $c \in |C|$ (for some $C \in \mathbf{C}$) satisfies a predicate $U \in \Phi C$, denoted $c \models_\omega U$, iff $c \in \omega_C U$. We omit the subscript whenever it is clear from the context.

Example 6.2.8

We instantiate our framework by defining an indexed category over our working category $\mathbf{C} = \mathbf{K}\ell(T)$. Recall the standard indexed category $\tilde{\mathcal{Q}}$ on **Set** (Example 6.2.5). Composing with the dual of forgetful functor $\mathbf{K}\ell(T) \xrightarrow{|\_|} \mathbf{Set}$ gives an indexed category on $\mathbf{K}\ell(T)$, where Φ is given by the composition $\mathbf{K}\ell(T)^{\mathrm{op}} \xrightarrow{|\_|} \mathbf{Set}^{\mathrm{op}} \xrightarrow{\tilde{\mathcal{Q}}} \mathbf{Cat}$.



As a result, predicate liftings are of type $\tilde{\mathcal{Q}} \circ |\_| \longrightarrow \tilde{\mathcal{Q}} \circ |\_| \circ F$. Lastly, $\omega = Id_\Phi$ and therefore $\omega_C = id_C$ (for each $C \in \mathbf{C}$) since a predicate on C is a subset of TC in this setting. Our framework is depicted above for the running example on NDAs, where $\bar{F}$ is an extension of $A \times \_ + 1$ (cf. Example 6.2.1) and $\Phi \xrightarrow{\lambda} \Phi \circ F$ is a predicate lifting which is yet to be defined.

Now, we are ready to define a more abstract view on separating sets of predicate liftings (compare with Definition 3.2.13):

Definition 6.2.9

Given a category $\mathbf{C}$. A set Λ of predicate liftings is *separating* with respect to F if, and only if,

$$\forall_{c,c' \in |FC|} \Big(\big(\forall_{U \in \Phi C, \lambda \in \Lambda} \ (c \in \omega_{FC} \lambda_C U \iff c' \in \omega_{FC} \lambda_C U) \big) \implies c = c' \Big).$$

for each $C \in \mathbf{C}$.

6.3 Kleisli Extension of Predicate Liftings

Predicate liftings play a major role for logic and games (see Chapter 3). Moving from a coalgebra $\alpha : X \to TFX$ in **Set** to a coalgebra in $\mathbf{K}\ell(T)$ the question arises how to obtain suitable predicate liftings $\Phi|X| \xrightarrow{\lambda} \Phi|\bar{F}X|$.

Since an indexed category on **Set** induces an indexed category on a Kleisli category (see Example 6.2.8) it seems natural to construct such predicate liftings based on predicate liftings $\Phi \xrightarrow{\sigma^F} \Phi F, \Phi \xrightarrow{\sigma^T} \Phi T$ of F, T which are both endofunctors on

Set. Summarizing the underlying components we have:

1. a functor F where $F : \mathbf{Set} \to \mathbf{Set}$
2. a monad (T, η, μ) where $T : \mathbf{Set} \to \mathbf{Set}$
3. a Kleisli extension $\mathbf{K}\ell(T) \xrightarrow{\bar{F}} \mathbf{K}\ell(T)$ and thus, a
4. distributive law $FT \xrightarrow{\sigma} TF$ such that σ is compatible with the multiplication μ.
5. an indexed category $\Phi \circ |\_|$ on Kleisli

To avoid the effort to create the necessary predicate liftings from scratch it seems natural to derive such predicate liftings (i.e. λ's) from the components listed above as in Equation 6.1.

$$\Phi TC = \Phi|C| \xrightarrow{\sigma^F_{|C|}} \Phi F|C| \xrightarrow{\sigma^T_{F|C|}} \Phi TF|C| \xrightarrow{\exists_{|\theta_C|}} \Phi|\bar{F}C| = \Phi TFC. \tag{6.1}$$

However, the approach presented in Equation 6.1 is not as natural as it seems. More precisely, usually λ fails to be a natural transformation.

Example 6.3.1

In the context of NDAs, i.e. when $\mathbf{C} = \mathbf{Set}$, $T = \mathcal{P}$, $F = A \times \_ + 1$, $\Phi = \tilde{\mathcal{Q}}$, for some alphabet A we will show that λ is not a natural transformation.
Therefore, we give a counterexample i.e. a function $f : X \to Y$ such that the diagram in Figure 6.4 does not commute.
First of all we explain the five different morphisms involved in that computation. Before we proceed, we want to emphasize how our notations are organized where we have (at least) four different functors applied to a non-empty set X:

$\mathbb{U} \in \tilde{\mathcal{Q}}\mathcal{P}X$	$\tilde{U} \in \tilde{\mathcal{Q}}(A \times \mathcal{P}X + 1)$
$\tilde{\mathbb{U}} \in \tilde{\mathcal{Q}}\mathcal{P}(A \times \mathcal{P}X + 1)$	$\bar{\mathbb{U}} \in \tilde{\mathcal{Q}}\mathcal{P}(A \times X + 1)$

Table 6.3: Organization of the notations with respect to the different functors with $U \in \mathcal{P}X$.

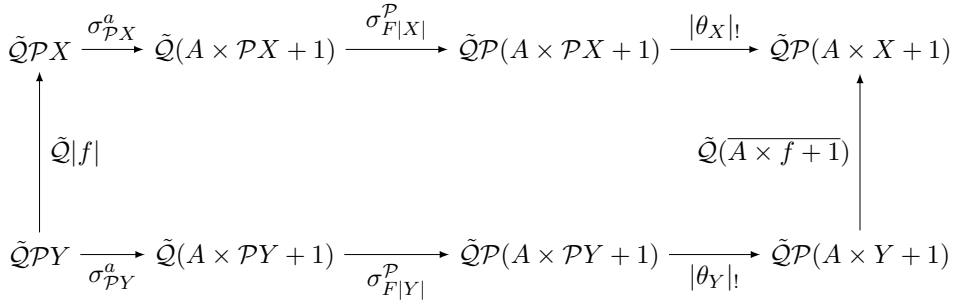


Figure 6.4: Naturality of predicate liftings implies that the given diagram commutes. Unfortunately, this does not hold for the lifting proposed by Equation 6.1.

Note, that in case we apply one of the functors above to a non-empty set Y we use V instead of U as basis for our notations (see Table 6.3).

1. $\tilde{\mathcal{Q}}|f|$ is based on $|f| = \mu_Y \circ \mathcal{P}f$ and defined as follows:

$$|f|(U) = \{y \in Y \mid \exists x \in U \text{ s.t. } y \in f(x)\}$$

   and therefore we conclude:

$$\tilde{\mathcal{Q}}|f|\mathbb{V} = \{X' \in \mathcal{P}X \mid |f|(X') \in \mathbb{V}\}$$

2. Analogously we derive $\tilde{\mathcal{Q}}(\overline{A \times f + 1})$ as follows:

$$\tilde{\mathcal{Q}}(\overline{A \times f + 1})\bar{\mathbb{V}} = \{\bar{U} \in \mathcal{P}(A \times X + 1) \mid \overline{A \times f + 1}(\bar{U}) \in \bar{\mathbb{V}}\}$$

   where $\overline{A \times f + 1}(\bar{U}) =$

$$\{(a', y) \in A \times Y \mid \exists\ x\ (a', x) \in \bar{U} \wedge y \in f(x)\} \cup \{\bullet \mid \bullet \in \bar{U}\}$$

3. $\sigma^a_{\mathcal{P}X}\mathbb{U} = \{(a, \bar{U}) \mid \bar{U} \in \mathbb{U}\}$

4. $\sigma^{\mathcal{P}}_{F|Y|}\tilde{U} = \{U \mid U \subseteq \tilde{U}\}$ is the usual modal $\Box$ operator.

5. $|\theta_X|_!$ is the direct image of $|\theta_X|$ where θ is the distribution law.

Let $X = \{x_1, x_2\}$, $Y = \{y_1, y_2, y_3, y_4\}$, $A = \{a\}$, and a function $f : X \to \mathcal{P}Y$ which maps $x_1 \mapsto \{y_1, y_3\}$ and $x_2 \mapsto \{y_2, y_4\}$.
Now, consider $\mathbb{V} = \{\{y_1, y_2\}, \{y_3, y_4\}\}$ where $\tilde{\mathcal{Q}}|f|\mathbb{V} = \emptyset$ which is lifted via the Construction 6.1 to $\{\emptyset\} \in \tilde{\mathcal{Q}}\mathcal{P}(A \times X + 1)$.

But, $\sigma^a_{\mathcal{P}Y}\mathbb{V} = \{(a,\{y_1,y_2\}),(a,\{y_3,y_4\})\}$ is lifted to

$$\{\emptyset,\{(a,\{y_1,y_2\})\},\{(a,\{y_3,y_4\})\},\{(a,\{y_1,y_2\}),(a,\{y_3,y_4\})\}\}$$

via $\sigma^{\mathcal{P}}_{F|Y|}$. Based on the direct image of the distribution law we get:

$$\{\emptyset,\{(a,y_1),(a,y_2)\},\{(a,y_3),(a,y_4)\},\{(a,y_1),(a,y_2),(a,y_3),(a,y_4)\}\}$$

and we denote this set with $\bar{\mathbb{V}}$. Finally, we see that Construction 6.1 fails to be natural since for $\{(a,x_1),(a,x_2)\}$ we have $\overline{A\times f+1}(\{(a,x_1),(a,x_2)\}) = \{(a,y_1),(a,y_2),(a,y_3),(a,y_4)\} \in \bar{\mathbb{V}}$ and $\{\{(a,x_1),(a,x_2)\}\} \neq \{\emptyset\}$ which is obtained the other way around within the diagram (as explained above).

Since the problem can be traced back to the distributive law $FT \xrightarrow{\sigma} TF$ a solution is given by a *distributive law* between F and a corresponding functor $G : \mathbf{Set} \to \mathbf{Set}$ inspired by abstract determinization techniques for a similar setting [JSS15].

Considering a coalgebra $\alpha : X \to HX$ where H includes a monad T there are two options for T within H:

$$X \to G(TX) \qquad\qquad X \to T(FX)$$

The monad can act inside or influence the branching from the outside. Typically in various applications, G is associated with the branching type of a deterministic version of the corresponding system of interest (like $G = \_^A \times 2$ in the case of NDA). As described in [JSS15] in the case of NDAs both forms are equivalent, but in general this does not hold.

Figure 6.5: The distributive laws of F and G wrt. T.

To circumvent the problem with the distributive law $FT \xrightarrow{\theta} TF$ we consider the functor G equipped with a natural transformation between F and G as illustrated in Figure 6.5 which yields the following idea

$$\Phi|C| \xrightarrow{\sigma^G_{|C|}} \Phi G|C| \xrightarrow{(\gamma_C)^*} \Phi|\bar{F}C| \tag{6.2}$$

Thus, given a monad T, a coalgebra $\alpha : X \to TFX$ and a functor G, the following Lemma 6.3.2 helps to find a distributive law $TF \xrightarrow{\gamma} GT$.

Lemma 6.3.2

Let $\bar{F}$ be a Kleisli extension of F which induces a distributive law $FT \xrightarrow{\theta} TF$. Then every distributive law $TF \xrightarrow{\gamma} GT$ which is compatible with θ and μ in the sense that the following square

$$\begin{array}{ccc} TFTC & \xrightarrow{\gamma_{TC}} & GTTC \\ {\scriptstyle T\theta_C}\downarrow & & \\ TTFC & & \downarrow{\scriptstyle G\mu_C} \\ {\scriptstyle \mu_{FC}}\downarrow & & \\ TFC & \xrightarrow{\gamma_C} & GTC \end{array}$$

commutes, induces a distributive law $|\_| \circ \bar{F} \longrightarrow G \circ |\_|$. Moreover, the converse also holds.

Proof: Let $C \xrightarrow{f} D \in \mathbf{K}\ell(T)$. Then we need to show that the following square on the left commutes. But this follows immediately by the commutative diagram drawn on the right, where the top square commutes due to the naturality of γ.

$$\begin{array}{ccc} |\bar{F}C| & \xrightarrow{\gamma_C} & G|C| \\ {\scriptstyle |\bar{F}f|}\downarrow & & \downarrow{\scriptstyle G|f|} \\ |\bar{F}D| & \xrightarrow{\gamma_D} & G|D| \end{array} \qquad \begin{array}{ccc} TFC & \xrightarrow{\gamma_C} & GTC \\ {\scriptstyle TFf}\downarrow & & \downarrow{\scriptstyle GTf} \\ TFTD & \xrightarrow{\gamma_{TD}} & GTTD \\ {\scriptstyle T\theta_D}\downarrow & & \\ TTFD & & \downarrow{\scriptstyle G\mu_D} \\ {\scriptstyle \mu_{FD}}\downarrow & & \\ TFD & \xrightarrow{\gamma_D} & GTD \end{array}$$

For the converse, take $f = \mathrm{id}_{TC}$ and view it as a Kleisli arrow $TC \longrightarrow C$. □

Now we are ready to define predicate liftings for Kleisli categories. Given an indexed morphism $\Phi \xrightarrow{\sigma^G} \Phi G$ and a distributive law γ as described previously we obtain a predicate lifting λ_C defined in the following way:

$$\Phi|C| \xrightarrow{\sigma^G_{|C|}} \Phi G|C| \xrightarrow{(\gamma_C)^*} \Phi|\bar{F}C| \tag{6.3}$$

Theorem 6.3.3

The above mapping λ is a predicate lifting.

Proof: This result follows from the naturality of σ^G and γ (similar to the one in Definition 3.2.9). In case two diagrams (a left one given by σ^G and one right diagram derived from γ) commute then the composition of these diagrams also commutes.

Given a function $f : X \to Y \in \mathbf{K}\ell(T)$, we know that the digram induced by f and σ^G commutes due to the naturality of σ^G. Therefore, it suffices to show that the naturality of $\gamma_C : TFX \to GTX$ implies the naturality of $\gamma_C^* : GTX \to TFC$ which is just the opposite natural transformation. □

In the context of logic and games we also need to show that we obtain a separating set of predicate liftings. Therefore, we explore our result in more detail on two different transition system types (NDA and LWA). In each application we first apply the construction of Equation 6.3 and then we show that this way we obtain a separating set of predicate liftings.

Non-Deterministic Finite Automata An NDA is modelled via a coalgebra $\alpha : X \to \mathcal{P}(A \times X + 1) \in \mathbf{Set}$. If one wants to work with language equivalence instead of bisimulation it is already described in Example 6.2.1 how the lifting of $F = A \times X + 1$ to $\bar{F}$ (i.e. the Kleisli extension) works. Recall that our Kleisli category is $\mathbf{K}\ell(\mathcal{P})$ which is equivalent to **Rel**.

Now, we want to instantiate the Kleisli extension of a predicate lifting given in Equation 6.3 to NDAs. Our monad is the powerset monad $T = \mathcal{P}$ and the situation is the following:

$G = \_^A \times \{0,1\}$	$F = A \times \_ + 1$
$X \to (\mathcal{P}X)^A \times \{0,1\}$	$X \to \mathcal{P}(A \times X + 1)$

Table 6.4: The **Set** endofunctors G and F in combination with the powerset monad [JSS15].

Recall, that $2 = \{0,1\}$ and we have three different functors applied to a non-empty set X:

$U \in \tilde{\mathcal{Q}}(X)$	$\bar{U} \in \mathcal{P}(A \times X + 1)$	$\mathbb{U} \in \tilde{\mathcal{Q}}\mathcal{P}X$

Next, we define the distributive law $\gamma : TF \to GT$ as follows [JSS15]:

$$\mathcal{P}(A \times X + 1) \xrightarrow{\gamma_X} (\mathcal{P}X)^A \times 2 \qquad \bar{U} \mapsto (\gamma_X^A \bar{U}, \gamma_X^2 \bar{U}),$$
$$\text{where } \gamma_X^A \bar{U}(a) = \{x \mid (a,x) \in \bar{U}\} \qquad \text{and} \qquad \gamma_X^2 \bar{U} = 1 \iff \bullet \in \bar{U}.$$

Moreover, from [JSS15] we know that γ is compatible with θ and the multiplication (i.e. $\bigcup$) in the sense of Lemma 6.3.2. In addition, we also need a predicate lifting

of $G = \_^A \times 2$ with respect to the indexed category $\tilde{\mathcal{Q}}$. So consider the family of liftings $\tilde{\mathcal{Q}}X \xrightarrow{\sigma^a_X} \tilde{\mathcal{Q}}(X^A \times 2)$ (for each $a \in A$) and $\tilde{\mathcal{Q}}X \xrightarrow{\sigma^{\downarrow}_X} \tilde{\mathcal{Q}}(X^A \times 2)$:

$$U \mapsto \{(p, b) \in X^A \times 2 \mid p(a) \in U\} \qquad \text{and} \qquad U \mapsto \{(p, 1) \mid p \in X^A\}.$$

Lemma 6.3.4

The above mappings σ^a_X (for $a \in A$) and $\sigma^{\downarrow}_X$ are predicate liftings.

Proof: Let $V \subseteq Y$ and $X \xrightarrow{f} Y$ be a function and

$$Gf : X^A + 2 \to Y^A + 2 \text{ with } Gf(p, b) = (f \circ p, b)$$

Then,

$$\begin{aligned}
&(Gf)^{-1}\sigma^a_Y(V) = (Gf)^{-1}\{(q, b) \in Y^A \times 2 \mid q(a) \in V\} \\
&= \Big\{(p, b') \in X^A \times 2 \mid (Gf)(p, b') \in \{(q, b) \mid q(a) \in V\}\Big\} \\
&= \Big\{(p, b') \in X^A \times 2 \mid (f \circ p, b') \in \{(q, b) \mid q(a) \in V\}\Big\} \\
&= \{(p, b) \mid f(p(a)) \in V\} \\
&= \{(p, b) \mid p(a) \in f^{-1}(V)\} \\
&= \sigma^a_X(f^{-1}(V)).
\end{aligned}$$

For the mapping associated with termination, we derive

$$\begin{aligned}
&(Gf)^{-1}\sigma^{\downarrow}_Y(V) = (Gf)^{-1}\{(q, 1) \mid q \in Y^A\} \\
&= \Big\{(p, b) \in X^A \times 2 \mid (Gf)(p, b) \in \{(q, 1) \mid q \in Y^A\}\Big\} \\
&= \Big\{(p, 1) \in X^A \times 2 \mid f \circ p \in Y^A\Big\} \\
&= \sigma^{\downarrow}_X(f^{-1}(V)).
\end{aligned}$$

□

Thanks to Theorem 6.3.3, we know that $\gamma^{-1} \circ \sigma^a, \gamma^{-1} \circ \sigma^{\downarrow}$ are valid predicate liftings for the Kleisli extension $\bar{F}$ of an endofunctor $F = A \times \_ + 1$ of **Set** to an endofunctor $\overline{A \times \_ + 1}$ of **Rel**:

$$\mathbf{Rel} \xrightarrow{\overline{A \times \_ + 1}} \mathbf{Rel}$$

Finally, given a set A of actions we obtain for any set X and each $a \in A$ the predicate liftings:

$$\lambda^a_X, \lambda^{\downarrow}_X : \Phi|X| \xrightarrow{\sigma^G_{|X|}} \Phi G|X| \xrightarrow{(\gamma_X)^*} \Phi|\bar{F}X|$$

These liftings map a subset $\mathbb{U} \subseteq \mathcal{P}X$ to a subset of $\mathcal{P}\bar{F}X$ where we have to distinguish between different actions and termination.

For each $a \in A$ we compute

$$\begin{aligned}\lambda^a_X(\mathbb{U}) &= {\gamma_X}^{-1}\sigma^a_{\mathcal{P}X}(\mathbb{U}) \\ &= {\gamma_X}^{-1}\{(p,b) \in (\mathcal{P}X)^A \times 2 \mid p(a) \in \mathbb{U}\} \\ &= \left\{\bar{U} \mid \gamma_X(\bar{U}) \in \{(p,b) \mid p(a) \in \mathbb{U}\}\right\} \\ &= \left\{\bar{U} \mid \gamma^A_X \bar{U}(a) \in \mathbb{U}\right\} \\ &= \left\{\bar{U} \mid \{x \mid (a,x) \in \bar{U}\} \in \mathbb{U}\right\}\end{aligned}$$

and for termination we derive

$$\begin{aligned}\lambda^{\downarrow}_X(\mathbb{U}) &= {\gamma_X}^{-1}\sigma^{\downarrow}_{\mathcal{P}X}(\mathbb{U}) \\ &= {\gamma_X}^{-1}\{(p,1) \mid p \in (\mathcal{P}X)^A\} \\ &= \left\{\bar{U} \mid \gamma_X(\bar{U}) \in \{(p,1) \mid p \in (\mathcal{P}X)^A\}\right\} \\ &= \left\{\bar{U} \mid \gamma^2_X \bar{U} = 1\right\} \\ &= \left\{\bar{U} \mid \bullet \in \bar{U}\right\}.\end{aligned}$$

As a conclusion we relate our predicate liftings to the results of the determinization techniques [JSS15].

Our predicate liftings induce the *action* (λ^a_X) and *termination* ($\lambda^{\downarrow}_X$) modalities after determinization for a given coalgebra $\alpha : X \to \mathcal{P}(A \times X + 1) \in \mathbf{Set}$. Since we apply $\gamma_X \circ \mu \circ |\alpha|$ we determinize α into a system over the state space $\mathcal{P}X$ [JSS15]. The application of $|\alpha|$ results in $\mathcal{P}\mathcal{P}FX$ and therefore the composition $\gamma_X \circ \mu \circ |\alpha|$ with $\mu = \cup$ defines the behavioural dynamics of the automaton (with codomain $G|X|$).

This means, that for a given $\mathbb{U} \in \tilde{\mathcal{Q}}\mathcal{P}X$ a state $U \in \mathcal{P}X$ satisfies a *modality* $[\lambda^a_X]$ if $|\alpha|(U) \in \lambda^a_X(\mathbb{U})$ where as usual in the context of logic $\mathbb{U}$ denotes the semantic of some formula (see Section 3.2.3). For Spoiler-Duplicator games, $\mathbb{U}$ will play a major role in the moves defined for each player.

Therefore, we need to know what does it mean if $|\alpha|(U) \in \lambda^a_X(\mathbb{U})$ holds and the computation ${|\alpha|}^{-1}(\lambda^a_X(\mathbb{U}))$ yields the following:

$$|\alpha|(U) \in \lambda^a_X(\mathbb{U}) \iff \{x' \mid \exists\, x \in U \text{ s.t. } x \xrightarrow{a} x'\} \in \mathbb{U}$$

Analogously, for $\lambda^{\downarrow}_X$ we obtain:

$$|\alpha|(U) \in \lambda^{\downarrow}_X(\mathbb{U}) \iff \exists x \in U \text{ s.t. } \bullet \in \alpha(x)$$

As mentioned at the beginning of this paragraph, we need a separating set Λ of predicate liftings. Such a set Λ is expressive enough to distinguish two different elements $t_1, t_2 \in \mathcal{P}FX$ and the predicate liftings that we derived are indeed separating.

Lemma 6.3.5

The set $\Lambda = \{\lambda^a \mid a \in A\} \cup \{\lambda^1\}$ is separating with respect to $A \times X + 1$.

Proof: Let $t_1, t_2 \in \mathcal{P}(A \times X + 1)$ such that $t_1 \neq t_2$. Hence we have to consider only two cases:

1. For some (a, x) it holds that $(a, x) \in t_1$ but $(a, x) \notin t_2$: Here we consider the predicate $\mathbb{U} = \{\{x\}\}$ together with the lifting λ^a_X. The lifting then yields all such $\bar{U} \subseteq A \times X + 1$ containing at least (a, x), hence $t_2 \notin \lambda^a_X(\mathbb{U})$. Obviously $t_1 \in \lambda^a_X(\mathbb{U})$.

2. For 1 it holds that $\bullet \in t_1$ but $\bullet \notin t_2$: Here it does not matter, which predicate we choose as long as we work with λ^1_X. Based on this lifting, we know that $t_2 \notin \lambda^1_X(\mathbb{U})$ for all $\mathbb{U} \in \tilde{\mathcal{Q}}\mathcal{P}X$, hence all elements in the lifting contain termination.

□

The construction and proofs for our next application work similarly and therefore we just present the results and the proofs can be found in Section A.4 of the appendix.

(Linear) Weighted Automata For LWA we fix $\mathbf{C} = \mathbf{Set}$, $T = \mathcal{M}_{\mathbb{F}}$, $F = A \times \_ + 1$, $G = \_^A \times \mathbb{F}$, and recall from [JSS15, Section 7.3] the distributive laws $A \times \mathcal{M}_{\mathbb{F}}X + 1 \xrightarrow{\theta_X} \mathcal{M}_{\mathbb{F}}(A \times X + 1)$ and $\mathcal{M}_{\mathbb{F}}(A \times X + 1) \xrightarrow{\gamma_X} (\mathcal{M}_{\mathbb{F}}X)^A \times \mathbb{F}$:

$$\theta_X(\bullet)(\heartsuit) = \begin{cases} 1, & \text{if } \heartsuit = \bullet \\ 0, & \text{otherwise.} \end{cases}$$

$$\theta_X(a, \tau)(\heartsuit) = \begin{cases} \tau(x), & \text{if } \heartsuit = (a, x), \text{ for some } x \in X \\ 0, & \text{otherwise.} \end{cases}$$

where

$$\gamma_X(p) = (\gamma^A_X p, p(\bullet)), \text{ where } \gamma^A_X p(a)(x) = p(a, x) \quad (\text{for } a \in A, x \in X)$$

We know (from [JSS15]) that γ is compatible with θ and μ (the multiplication of the monad $\mathcal{M}_{\mathbb{F}}$) in the sense of Lemma 6.3.2. In light of the NDAs, consider the following

predicate liftings to characterize weighted language equivalence $\tilde{\mathcal{Q}}X \xrightarrow{\sigma^a_X, \sigma^s_X} \tilde{\mathcal{Q}}(X^A \times \mathbb{F})$ (for $a \in A$ and $s \in \mathbb{F}$): $U \mapsto \{(p, s) \in X^A \times \mathbb{F} \mid p(a) \in U\}$ and $U \mapsto \{(p, s) \mid p \in X^A\}$, respectively.

Lemma 6.3.6

The above mappings σ^a_X, σ^s_X (for $a \in A, s \in \mathbb{F}$) are indexed morphisms.

The proof of this lemma is similar to the proof of Lemma 6.3.4. And thanks to Theorem 6.3.3, we know that $\gamma^{-1} \circ \sigma^a$ and $\gamma^{-1} \circ \sigma^s$ are valid predicate liftings.

Lemma 6.3.7

For any $\mathbb{U} \subseteq \mathcal{M}_{\mathbb{F}}X$ we find that $\lambda^a_X(\mathbb{U}) = \{\bar{p} \in \mathcal{M}_{\mathbb{F}}(A \times X + 1) \mid \gamma^A_X \bar{p}(a) \in \mathbb{U}\}$ and $\lambda^s_X(\mathbb{U}) = \{\bar{p} \in \mathcal{M}_{\mathbb{F}}(A \times X + 1) \mid \bar{p}(\bullet) = s\}$.

Just like in our running example above, the determinization of an LWA α is the composition:

$$\mathcal{M}_{\mathbb{F}}X \xrightarrow{\mathcal{M}_{\mathbb{F}}\alpha} \mathcal{M}_{\mathbb{F}}\mathcal{M}_{\mathbb{F}}(A \times X + 1) \xrightarrow{\mu_{A \times X + 1}} \mathcal{M}_{\mathbb{F}}(A \times X + 1) \xrightarrow{\gamma_X} (\mathcal{M}_{\mathbb{F}}X)^A \times \mathbb{F}.$$

More concretely, it maps a $p \in \mathcal{M}_{\mathbb{F}}X$ to a pair $(\hat{p}, s)$, where

$$\hat{p}(a)(x') = \sum_{x \in X} p(x) \cdot \alpha(x)(a, x') \text{ and } s = \sum_{x \in X} p(x) \cdot \alpha(x)(\bullet).$$

In terms of SOS rules, determinisation is given as follows:

$$\frac{p \in \mathcal{M}_{\mathbb{F}}X}{p \xrightarrow{a} \hat{p}(a)} \qquad \frac{p \in \mathcal{M}_{\mathbb{F}}X \quad s = \sum_{x \in X} p(x) \cdot \alpha(x)(\bullet)}{p \downarrow_s}$$

where $p \downarrow_s$ denotes the termination weight s of p.

Lemma 6.3.8

For any $\mathbb{U} \subseteq \mathcal{M}_{\mathbb{F}}X$ and $X \xrightarrow{\alpha} \mathcal{M}_{\mathbb{F}}(A \times X + 1)$, we have

$$|\alpha|^{-1}\lambda^a_X(\mathbb{U}) = \{p \in \mathcal{M}_{\mathbb{F}}X \mid p \xrightarrow{a} \hat{p}(a) \implies \hat{p}(a) \in \mathbb{U}\}$$

and $|\alpha|^{-1}\lambda^s_X(\mathbb{U}) = \{p \in \mathcal{M}_{\mathbb{F}}X \mid s = \sum_{x \in X} p(x) \cdot \alpha(x)(\bullet)\}$. Moreover, the set $\Lambda = \{\lambda^a \mid a \in A\} \cup \{\lambda^s \mid s \in \mathbb{F}\}$ is separating with respect to $A \times X + 1$.

We summarize our contribution. In this chapter we have seen how modalities can be constructed in case we consider a coalgebra in Kleisli and are aware of the relevant functor G.

In the next section we define expressive logics and also play games in this setting.

6.4 Coalgebraic Modal Logic and Games Beyond Set

This section is dedicated to our main building blocks within this thesis: *logic* and *games*. Therefore, we present an abstract game characterization based on indexed categories and adapt the well-known coalgebraic view on modal logic.

Since categories such as $\mathbf{K}\ell(\mathcal{P})$ fails to admit all coequalizers, we need to establish a proof-technique for the completeness parts of our proofs. We start with a work-around for the construction of a witnessing coalgebra homomorphism based on a given equivalence relation present in **Set** and lifted to the category **C** (see restrictions of the setting in Section 6.2).

6.4.1 The Witnessing Coalgebra Homomorphisms

Coalgebraically, given a coalgebra $\alpha : X \to FX$ in a concrete category **C** two states $c, c' \in |X|$ are behaviour equivalent if there exists a coalgebra homomorphism $f : X \to Y \in \mathbf{C}$ such that $|f|(x) = |f|(y)$ (compare Definition 2.3.17). In Chapter 3 such a witness with respect to logical/game-theoretical equivalence is constructed based on the fact that **Set** has all coequalizers. Unfortunately this does not hold in general since $\mathbf{K}\ell(\mathcal{P})$ does not admit all coequalizers [Mil00].

Example 6.4.1

We consider $\mathbf{Rel} = \mathbf{K}\ell(\mathcal{P})$ to show that there exist categories which fail to admit all coequalizers (see Definition 2.3.15). For $\mathbf{Rel} = \mathbf{K}\ell(\mathcal{P})$ this observation is mentioned in many publications, but for completeness we consider the partial order $\leq = \{(0,0),(0,1),(1,1)\} \subseteq \{0,1\} \times \{0,1\}$, which is also omnipresent in Chapter 3. Note, that $\leq$ can be seen as a morphism in **Rel**.
The situation is depicted in Figure 6.6, given two parallel relations $\leq, id_{\{0,1\}}$ we search for a coequalizer $q : \{0,1\} \to Q$. Furthermore, for $\leq$ we have that $\leq \circ \leq = \leq \circ id_{\{0,1\}}$ holds and by the properties of a coequalizer q there must exists a unique morphism u with $\leq = u \circ q$.
We need to demonstrate that in general no relation q with the required properties

$$q \circ \leq = \ q \circ \ id_{\{0,1\}} \text{ and } \ \leq = u \circ q \text{ by Definition 2.3.15}$$

exists. Therefore, for any candidate q, Q we notice the following:

1. At least for some $e, e' \in Q$ with $(0,e),(1,e') \in q$ we have that $e \neq e'$ since $(0,0) \in \leq$ implies $(0,e) \in q$ and $(e,0) \in u$ at least for one $e \in Q$. And if $(1,e) \in q$ we have that $(e,0) \notin u$ since $(1,0) \notin \leq$. This implies that we

need at least two different elements $e, e' \in Q$. Therefore $Q = \emptyset$ or $Q = 1$ are excluded.

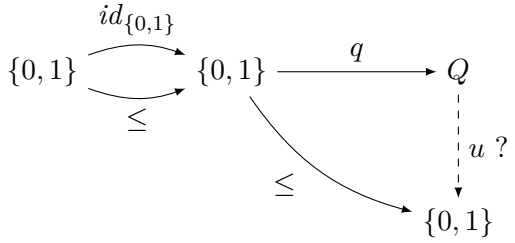


Figure 6.6: $\mathbf{K}\boldsymbol{\ell}(\mathcal{P})$ fails to admit all coequalizers.

2. Next, consider any $Q \in \mathbf{K}\boldsymbol{\ell}(\mathcal{P})$ with $|Q| \geq 2$ and due to (1) we have $e \neq e' \in Q$ such that

$$\begin{array}{lll} (0,e) \in q & (e,0) \in u & \text{since } (0,0) \in \leq \\ (1,e') \in q & (e',1) \in u & \text{since } (1,1) \in \leq \\ & & \text{but } (1,0) \notin \leq = u \circ q \end{array}$$

For q and u given as above we have $\{(0,e),(1,e')\} \neq \{(0,e),(0,e'),(1,e')\}$ and thus $q = q \circ \ id_{\{0,1\}} \neq \ q \circ \leq$. Therefore, for any $(1,e') \in q$ we need to add $(0,e')$ to q to obtain $q \circ \leq = \ q \circ \ id_{\{0,1\}} = q$.

Now we satisfy the first property and $\leq = u \circ q$ holds but we can construct a u' adding $(e,1)$ to u which still satisfies $\leq = u' \circ q$, but this violates the uniqueness of u.

On the otherside, any additional $(0,l) \in q$ with $l \neq e$ again violates the uniqueness of u since we can add $(l,0)$ or $(l,1)$ to u such that $\leq = u \circ q$ holds.

Finally, we conclude that no coequalizer exists (cf. [nca09]).

However, it still could be the case that the specific coequalizers that we are interested in (i.e. based on an equivalence relation $\equiv \in \mathbf{C}$ equipped with the usual projections) exist. But it still would remain unclear if such a coequalizer yield the necessary witnessing coalgebra homomorphism.

Therefore, we consider a special type of subcategories inspired by the work in [AB+12] whenever the category of interest lacks the existence of epi-mono factorizations.

Definition 6.4.2

A category $\mathbf{B}$ is a *reflective* subcategory of $\mathbf{C}$ iff there is an inclusion functor $\mathbf{B} \xrightarrow{\subset \mathcal{I}_r} \mathbf{C}$ and $\mathcal{I}_r$ has a left adjoint $\mathbf{C} \xrightarrow{\mathcal{R}} \mathbf{B}$ (often called *reflector*).

Recall, that summarized we get a composition of two adjunctions where one is derived from the base category (i.e. $\mathbf{K}\ell(T)$) and the forgetful functor which is used for concretization and the second adjunction results from the reflective subcategory which enables the construction of a witnessing coalgebra homomorphism (see Figure 6.7).

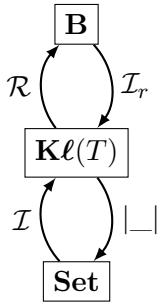


Figure 6.7: A second adjunction results from the reflective subcategory B.

As with the Kleisli extensions where a **Set** functor F has to be lifted to a Kleisli functor $\bar{F}$ (see Figure 2.16) we need to take care under which conditions an F-coalgebra in the base category is reflected to an F-coalgebra in the subcategory.

Theorem 6.4.3: [AB+12; HJ98]

Let $\mathbf{B} \xrightarrow{\subset \mathcal{I}_r} \mathbf{C}$ be a reflective subcategory of $\mathbf{C}$. If $\mathbf{C} \xrightarrow{F} \mathbf{C}$ preserves $\mathbf{B}$, i.e., $\forall_{B,f \in \mathbf{B}}\ (FB \in \mathbf{B} \wedge Ff \in \mathbf{B})$ and $F \circ \mathcal{I}_r = \mathcal{I}_r \circ F$, then we have a diagram $\mathbf{Coalg}_B(F) \underset{-\bar{\mathcal{I}}_r}{\overset{\bar{\mathcal{R}}-}{\leftrightarrows}} \mathbf{Coalg}_C(F)$ with $\bar{\mathcal{R}} \dashv \bar{\mathcal{I}}_r$. Here, $\bar{\mathcal{I}}_r$ is the obvious inclusion.

The reflector $\bar{\mathcal{R}}$ typically results in a form of (on-the-fly) determinization, which will be spelled out in more detail after presenting the main contribution of this subsection. Recall, that the motivation of moving to a reflective subcategory results from the fact that some categories do not have all coequalizers. Therefore, we show how the two adjunctions help to construct a coequalizer in the subcategory.

The proof for the coequalizer construction of $f \in \mathbf{B}$ which serves as a base for the witnessing coalgebra homomorphism $g \in \mathbf{C}$ with respect to logical (game) equivalence can be found in Appendix A.4. Here we give a more intuitive explanation for the construction of $f \in \mathbf{B}$.

Given a logical equivalence $\equiv \subseteq |C| \times |C|$ on the underlying state space of the coalgebra $C \xrightarrow{\alpha} FC$, then the idea is to use the following series of transformations

due to the two adjunctions $\mathcal{R} \dashv \mathcal{I}_r$ and $\mathcal{I} \dashv |\_|$. Below we fix ε, η and ε', η' for the (co)unit of the adjunctions $\mathcal{R} \dashv \mathcal{I}_r$ and $\mathcal{I} \dashv |\_|$, respectively.

$$\frac{\equiv \xrightarrow{\pi_i} |C| \quad \in \mathbf{Set}}{\mathcal{I} \equiv \xrightarrow{\pi_i'} C \xrightarrow{\eta_C} \mathcal{I}_r\mathcal{R}C \quad \in \mathbf{C}}$$
$$\mathcal{R}\mathcal{I} \equiv \xrightarrow{\tilde{\pi}_i} \mathcal{R}C \quad \in \mathbf{B}$$

Note that the counit-unit identities results in $\mathcal{R}\pi_i' = \tilde{\pi}_i$. As $\mathbf{B}$ has all coequalizers, we can construct the coequalizer $\mathcal{R}\mathcal{I} \equiv \xrightarrow[\tilde{\pi}_2]{\tilde{\pi}_1} \mathcal{R}C \xrightarrow{f} B$. Finally, the transpose of f yields the necessary coalgebra homomorphism denoted with g.

In order to show that g is a witnessing coalgebra homomorphism we need to prove that $Ff \circ \bar{\mathcal{R}}\alpha \circ \tilde{\pi}_1 = Ff \circ \bar{\mathcal{R}}\alpha \circ \tilde{\pi}_2$ holds. Therefore, we need the following two properties where g is again the transpose of f, $\bar{\mathcal{R}}$ the lifting of the reflector $\mathcal{R}$ given by Theorem 6.4.3, and $\tilde{\pi}_i$ with $i \in \{1, 2\}$ are the transposes of $\eta_C \circ \pi_i'$. Here π_i' is the transpose of the usual projection π_i of an equivalence relation $\equiv \subseteq |C| \times |C|$ under the adjunction $\mathcal{I} \dashv |\_|$.

1. First we show that for a given equivalence relation $\equiv \subseteq |C| \times |C|$ and all $(c, c') \in \equiv$ we get $|g|c = |g|c'$ where $g \in \mathbf{C}$ is the transpose of the constructed coequalizer $f \in \mathbf{B}$.

2. Second we need to show that $|Fg| \circ |\alpha| \circ \pi_1 = |Fg| \circ |\alpha| \circ \pi_2$ implies
$$Ff \circ \bar{\mathcal{R}}\alpha \circ \tilde{\pi}_1 = Ff \circ \bar{\mathcal{R}}\alpha \circ \tilde{\pi}_2.$$

Therefore, we state the following two results where the proofs can be found in the Appendix A.4:

Lemma 6.4.4

Given an equivalence relation $\equiv \subseteq |C| \times |C|$. If there is a reflective subcategory $\mathbf{B}$ of $\mathbf{C}$ having all the coequalizers, then for the transpose g of f given by $\mathcal{R}\mathcal{I} \equiv \xrightarrow[\tilde{\pi}_2]{\tilde{\pi}_1} \mathcal{R}C \xrightarrow{f} B$ we have that $|g|c = |g|c'$ for each $(c, c') \in \equiv$.

Lemma 6.4.5

Given a coequalizer f under the restrictions described in Lemma 6.4.4 and F preserves $\mathbf{B}$. For its transpose g it holds that $|Fg| \circ |\alpha| \circ \pi_1 = |Fg| \circ |\alpha| \circ \pi_2$ implies $Ff \circ \bar{\mathcal{R}}\alpha \circ \tilde{\pi}_1 = Ff \circ \bar{\mathcal{R}}\alpha \circ \tilde{\pi}_2$.

Lemma 6.4.4 is needed to show, that given the specific properties induced by a logical equivalence relation $\equiv_L$ or by a game equivalence relation $\equiv_G$ in **Set**, the transpose

$g_e \in \mathbf{C}$ of $f_e \in \mathbf{B}$ is indeed a witnessing coalgebra homomorphism for all $(x, y) \in \equiv_e$ with $e \in \{L, G\}$. Directly after this section we will separately define and prove these theorems, since each proof relies on the specific properties of the corresponding equivalence relation.

In addition, we explain in the following paragraph both constructions based on language equivalence for NDA's. Afterwards we proceed with the logical and game-theoretical view.

An example for the construction of a witnessing coalgebra homomorphism Now we want to focus on the coequalizer construction in $\mathbf{K}\ell(\mathcal{P})$ and therefore we consider a simple example. Therefore, we recall the idea described above:

$$\frac{\equiv \xrightarrow{\pi_i} |C| \quad \in \mathbf{Set}}{\mathcal{I} \equiv \xrightarrow{\pi_i'} C \xrightarrow{\eta_C} \mathcal{I}_r\mathcal{R}C \quad \in \mathbf{C}}$$
$$\mathcal{R}\mathcal{I} \equiv \xrightarrow{\tilde{\pi}_i} \mathcal{R}C \quad \in \mathbf{B}$$

Example 6.4.6

In this paragraph we consider an example automaton and introduce the language equivalence relation $\equiv_{La} \subseteq \mathcal{P}X \times \mathcal{P}X$ for the concrete states $\mathcal{P}X$ where X is the state space given by the coalgebra in $\mathbf{K}\ell(\mathcal{P})$.
The equivalence relation $\equiv_{La}$ contains only those pairs of states which accept the same language. This means for all $(V, W) \in \equiv_{La}$ it holds that there is some $x \in V$ that accepts the word $w \in A^*$ iff there is an $y \in W$ that accepts w too.

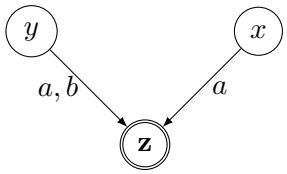


Figure 6.8: A simple NDA.

Let $X = \{x, y, z\} \in \mathbf{K}\ell(\mathcal{P})$ and $A = \{a, b\}$ the set of labels. The automaton in Figure 6.8 is an NDA with one accepting state z. The concrete states are given by $\mathcal{P}X$. Note, that $\{y\}$ and $\{x\}$ are not language equivalent since y accepts the word b but there is no such state in $\{x\}$.
Furthermore, in Figure 6.9 the language equivalent and concrete state pairs are ordered with respect to the words they accept.

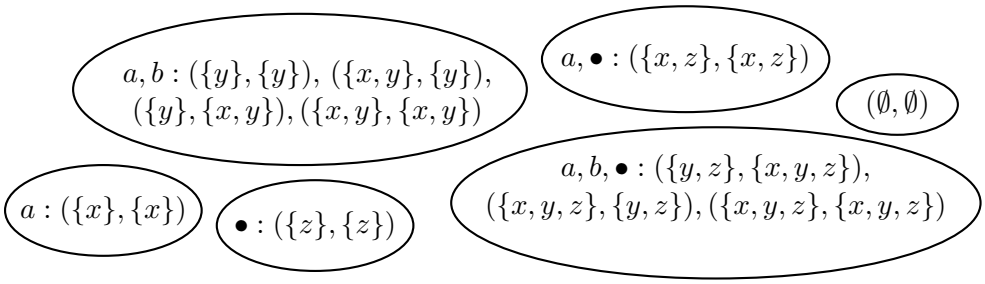


Figure 6.9: In this figure we see the language equivalence relation $\equiv_{La}$ for the NDA in Figure 6.8 with $X = \{x, y, z\}$. The language accepted by this NDA is $L = \{a, b, \bullet\}$ ($\bullet$ corresponds to the empty word). Additionally, all language equivalent state pairs are sorted according to equivalence classes.

Before we construct the coequalizer, we give the concrete definitions of the various arrows involved in it:

1. The projections π_i with $i \in \{1, 2\}$ are the usual projections given by the relation $\equiv_{La}$ and $\equiv_{La} \xrightarrow{\pi_i} |X|$ in **Set**.

2. The counit $\mathcal{P}X \xrightarrow{\varepsilon'_X} X \in \mathbf{K}\boldsymbol{\ell}(\mathcal{P})$ of $\mathcal{I} \dashv |\_|$ is the converse of membership relation, i.e. $(U, x) \in \varepsilon'_X \iff x \in U$.

3. Lifting of π_i to $\pi'_i = \varepsilon'_X \circ \mathcal{I}\pi_i \in \mathbf{K}\boldsymbol{\ell}(\mathcal{P})$ based on the counit ε'_X for $i \in \{1, 2\}$ and $(U, V) \in \equiv_{La} \subseteq \mathcal{P}X \times \mathcal{P}X$:

$$(U, V)\pi'_1 \; x \text{ iff } x \in U$$
$$(U, V)\pi'_2 \; y \text{ iff } y \in V$$

$$\mathcal{I} \equiv_{La} \overset{\mathcal{I}\pi_1}{\underset{\mathcal{I}\pi_2}{\rightrightarrows}} \mathcal{I}\mathcal{P}X \xrightarrow{\varepsilon'_X} \mathcal{I}X$$

4. Lifting of π'_i to $\tilde{\pi}_i = \mathcal{R}(\varepsilon'_X \circ \mathcal{I}\pi_i) = \mathcal{R}(\pi'_i)$ in $\mathbf{Set}^{\mathrm{op}}$ with respect to $\mathcal{R} \dashv \mathcal{I}_r$.

$$\tilde{\pi_1}(W) = \{(U, V) \in \equiv_{La} |\; U \cap W \neq \emptyset\}$$
$$\tilde{\pi_2}(W) = \{(U, V) \in \equiv_{La} |\; V \cap W \neq \emptyset\}$$

$$\tilde{\mathcal{Q}}\mathcal{I} \equiv_{La} \overset{\mathcal{R}\mathcal{I}\pi_1}{\underset{\mathcal{R}\mathcal{I}\pi_2}{\leftleftarrows}} \tilde{\mathcal{Q}}\mathcal{I}\mathcal{P}X \xleftarrow{\mathcal{R}\varepsilon'} \tilde{\mathcal{Q}}\mathcal{I}X$$

Before we proceed, note that the relations π'_i introduced above are quite intuitive. However, we explore two different cases of $\tilde{\pi}_i$ in Example 6.4.7 since the computations given by these arrows are a bit fizzy.

Example 6.4.7

We consider the NDA given in Figure 6.8 and the language equivalence $\equiv_{La}$ depicted in Figure 6.9. For the concrete states $\{x\}$ and $\{y\}$ we obtain two different cases, one where $\tilde{\pi}_1(\_) = \tilde{\pi}_2(\_)$ holds and one for which $\tilde{\pi}_1(\_) = \tilde{\pi}_2(\_)$ fails. First, we consider $\tilde{\pi}_i(\{y\})$ for $i \in \{1, 2\}$:

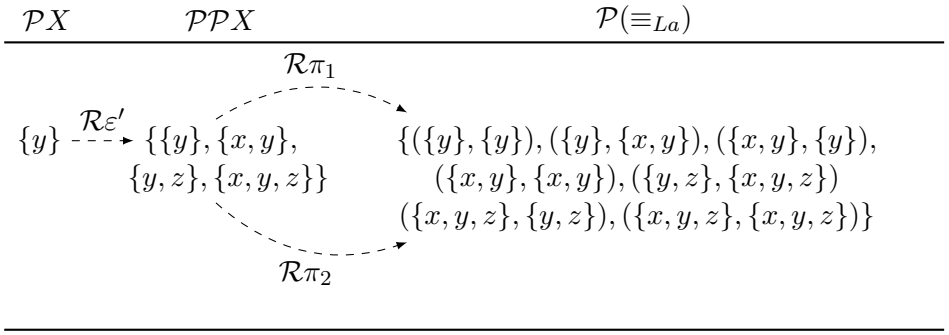


Figure 6.10: $\tilde{\pi}_i(\{y\})$ for $i \in \{1, 2\}$ and $\tilde{\pi}_1(\{y\}) = \tilde{\pi}_2(\{y\})$ holds.

Second, we consider $\tilde{\pi}_i(\{x\})$ for $i \in \{1, 2\}$:

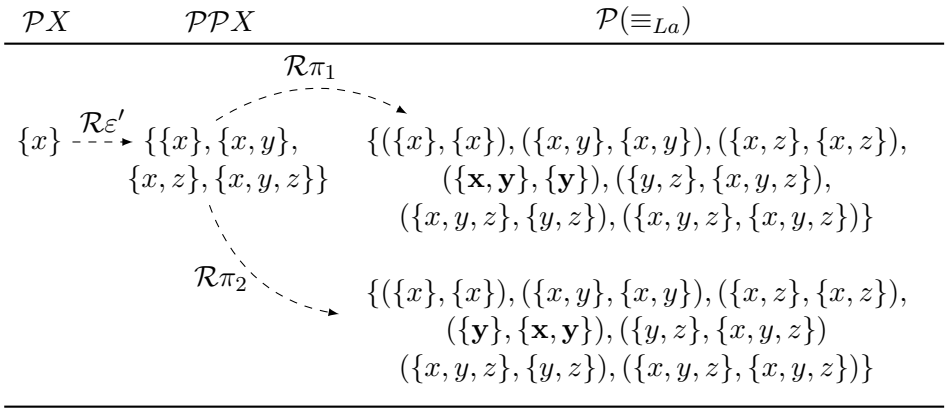


Figure 6.11: $\tilde{\pi}_i(\{x\})$ for $i \in \{1, 2\}$ and $\tilde{\pi}_1(\{x\}) \neq \tilde{\pi}_2(\{x\})$ holds.

Finally, we can move to the coequalizer construction based on the two parallel arrows $\tilde{\pi}_i$ with $i \in \{1, 2\}$ depicted here in Figure 6.12 where

$$B = \{W \in \mathcal{P}X \mid \tilde{\pi}_1(W) = \tilde{\pi}_2(W)\}.$$

Moreover, the coequalizer in $\mathbf{Set}^{\mathrm{op}}$ is nothing but an equalizer in $\mathbf{Set}$; thus, we find that (B, f) is an equalizer of the projections $\tilde{\pi}_i$ (for $i \in \{1, 2\}$).

Now that we know how to construct $f : \mathcal{R}C \to B$ and its transpose $g : C \to \mathcal{I}_r B$

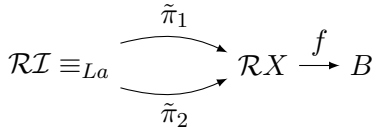


Figure 6.12: Coequalizer based on $\tilde{\pi}_i$.

we want to consider the arrow $B \xrightarrow{\beta} A \times B + 1 \in \mathbf{Set}^{\mathrm{op}}$ which is defined by the following universal property of equalizer in **Set** (see Figure 6.13). Here, $\bar{\mathcal{R}}\alpha$ is the backward determinization of the given coalgebra (as described, e.g. in [BB+12b] as a deterministic automaton accepting the reverse language), i.e. it maps $(a, U) \mapsto \{x \mid \exists_{x' \in U}(a, x') \in \alpha(x)\}$ and $\bullet \mapsto \{x \mid \bullet \in \alpha(x)\}$. Thus, in essence, β acts like $\bar{\mathcal{R}}\alpha$ on the elements of B (see Figure 6.13).

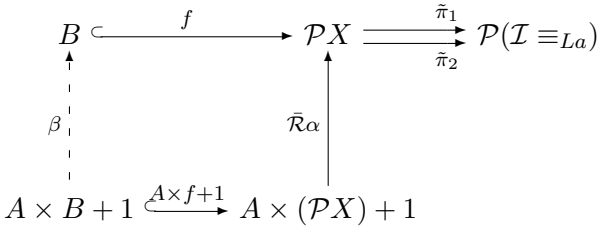


Figure 6.13: The equalizer $f \in \mathbf{Set}$ which is a coequalizer in $\mathbf{Set}^{\mathrm{op}}$.

In this example, we obtain as β the automaton drawn in Figure 6.14 on the right with six states. Note, that $\{x\}, \{x, z\}$ are removed because they do not satisfy $\tilde{\pi_1}(\_) = \tilde{\pi_2}(\_)$ as shown for $\{x\}$ in Example 6.4.7.

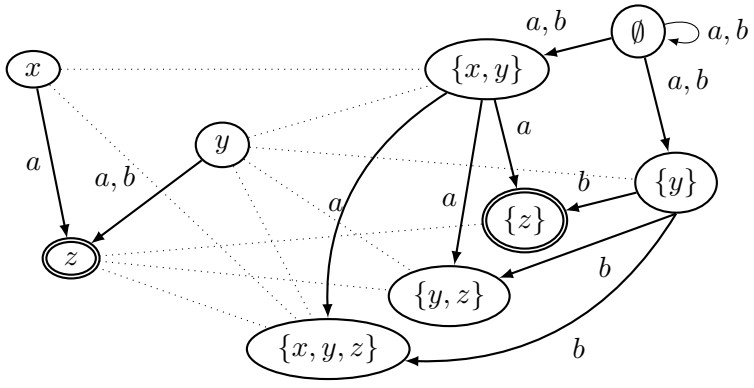


Figure 6.14: The transpose g of the constructed coequalizer f is given by the dashed lines.

The dotted line indicates the relation g obtained as the transpose of f with respect

to $\mathcal{R} \dashv \mathcal{I}_r$; concretely, $(x, U) \in g \iff x \in U$.

More importantly, $g \in \mathbf{K}\ell(\mathcal{P})$ is a witnessing coalgebra homomorphism because $\overline{A \times X + 1}(g) \circ \alpha = \beta \circ g$. This can be validated by inspecting for any $x \in X, W \in B$:

- $\exists_{U \in B}\ (x \in U \wedge (a, W) \in \beta(U)) \iff \exists_{x' \in X}((a, x') \in \alpha(x) \wedge x' \in W)$;
- $\exists_{U \in B}\ (x \in U \wedge \bullet \in \beta(U)) \iff \bullet \in \alpha(x)$.

Note that $|g|$ maps both $\{x, y\}, \{y\} \in |X|$ to $\{\{x, y\}, \{y\}, \{y, z\}, \{x, y, z\}\}$, witnessing the fact that they are language equivalent.

Finally, one can observe that the coequalizer gives us the largest sub-automaton of the backwards determinization that respects $\equiv_{La}$. Removing $\emptyset$, $\{y, z\}$, and $\{x, y, z\}$ will result in smallest such sub-automaton given by the backwards determinization [AB+12].

6.4.2 Logic

Coalgebraic modal logic is induced by a separating set Λ of predicate liftings where each $\lambda \in \Lambda$ is a natural transformation of type $\Phi \xrightarrow{\lambda} \Phi \circ F$. Therefore, we get an abstract modal logic by the following grammar (see [Pat03] or compare with Definition 3.2.18):

$$\varphi \in \mathbb{M}_\Lambda ::= \bigwedge_{i \in I} \varphi_i \mid \neg\varphi \mid [\lambda]\varphi, \qquad \text{for every } \lambda \in \Lambda.$$

To talk about the interpretation of the formulas and especially about the modal operators derived from the predicate liftings (i.e. the indexed morphisms) we need that the fibres of our indexed category are expressive enough. More concretely, the situation is depicted in Figure 6.15 and we require that the fibres provide enough information to interpret the formulas defined on our predicate liftings of type $\Phi \to \Phi$ (i.e indexed morphisms).

For the standard example of an indexed category based on the contravariant powerset functor Jacobs mentioned Boolean algebras (i.e. $\mathbf{Set}^{\mathrm{op}} \xrightarrow{\Phi} \mathbf{BA}$) as one possible option but in the same breath he argues, that **Poset** provides more algebraic structure [Jac10]. One needs to be careful in choosing the fibres induced by Φ (see *Grothendieck C.1*) since there are predicate liftings which do not need to preserve all the structure of a Boolean algebra. Such preservation properties rely on the nature of the concrete application (see [Jac10]).

Regarding our examples introduced in Section 6.3 the corresponding predicate liftings are meet preserving (see Appendix A.4.1). Fibrational frameworks which

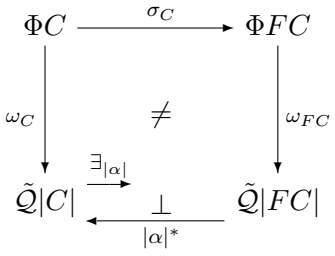


Figure 6.15: The fibers induced by the indexed category Φ should have enough structure due to the semantic of a formula φ.

are quite close to our work require meet preserving functions as used in [KK+19] to define generic games.

Nevertheless, one may also argue that this restriction excludes diamond modality since it does not preserve the meet operation.

Example 6.4.8

In this example we show that the standard $\Diamond$-operator (see Hennessy-Milner logic in Definition 3.2.2) fails to preserve meets. Consider the system in Figure 6.16 with state space $X = \{x, y, z, u\}$ and two formulas $\varphi_1 = \Diamond tt$ and $\varphi_2 = \neg\varphi_1$ where $[\![\varphi_1]\!] = \{x, y\}$ and $[\![\varphi_2]\!] = \{u, z\}$. Therefore we obtain

$$[\![\Diamond\varphi_1 \wedge \Diamond\varphi_2]\!] = [\![\Diamond\varphi_1]\!] \cap [\![\Diamond\varphi_2]\!] = \{x\} \cap \{x, y\} = \{x\} \text{ and}$$

$$[\![\Diamond(\varphi_1 \wedge \varphi_2)]\!] = \{x' \in X \mid \exists z' \in X. x' \to z' \text{ and } z' \in \emptyset\} = \emptyset$$

Clearly, $\emptyset \neq \{x\}$ which implies that $\Diamond$ is not meet preserving.

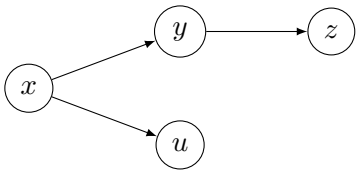


Figure 6.16: A simple unlabelled transition system.

One way to circumvent this situation is to consider the category of complete Boolean algebras with *order preserving* functions as morphisms. This works in our setting (since all our indexed morphisms are meet preserving, thus they are also order preserving) and one could include both box and diamond modalities in the logic [Jac01].

Since all our logics in various concrete cases are Boolean in nature, we therefore impose the following assumption.

Assumption: *Henceforth, indexed categories will be of type* $\mathbf{C}^{op} \longrightarrow \mathbf{cBA}_{\wedge}$*, where* $\mathbf{cBA}_{\wedge}$ *is the category of complete Boolean algebras and meet preserving morphism. Alternatively one can consider* $\mathbf{cBA}$ *the category of complete Boolean algebras and order preserving morphism.*

Another point for clarification at this stage is the purpose of having conjunction and negation in our logic, especially, when one of the aims is to characterize trace equivalence rather than bisimilarity.

Therefore, recall that our main motivation is to work out a logical framework as well as a game-theoretical characterization which behave similar to the notion of coalgebraic behavioural equivalence. Therefore, we need conjunction and negation from a general point of view. This means that on the one side the game-theoretical semantics should coincide with trace equivalence and on the other side with bisimulation.

Just like Hennessy-Milner logical formulae are satisfied by the states of a transition system, here the modal formulae will be satisfied by concrete states $c \in |C|$ of a coalgebra $C \xrightarrow{\alpha} FC \in \mathbf{C}$. However, this will be done in two steps due to the fact that concrete states and coalgebras live in different categories. First, given a formula $\varphi \in \mathbb{M}_{\Lambda}$ and a coalgebra $C \xrightarrow{\alpha} FC$ we construct a predicate $[\![\varphi]\!]_{\alpha} \in \Phi C$, which is obtained by structural induction:

- Let $\varphi = \bigwedge_{i \in I} \varphi_i$. Then $[\![\varphi]\!]_{\alpha} = \bigwedge_{i \in I} [\![\varphi_i]\!]_{\alpha}$.
- Let $\varphi = \neg\varphi'$. Then $[\![\varphi]\!]_{\alpha} = \neg[\![\varphi']\!]_{\alpha}$.
- Let $\varphi = [\lambda]\varphi'$. Then $[\![\varphi]\!]_{\alpha} = \alpha^{*}\lambda_{C}[\![\varphi']\!]_{\alpha}$.

Secondly, a concrete state $1 \xrightarrow{c} |C| \in \mathbf{Set}$ (or, $c \in |C|$) satisfies a formula φ, denoted $c \models_{\alpha} \varphi$, whenever $c^{-1}\omega_{C}[\![\varphi]\!]_{\alpha} = 1$. In other words, $c \models_{\alpha} \varphi \iff c \in \omega_{C}[\![\varphi]\!]_{\alpha}$.

We begin with the soundness part of our logical characterization (see [Pat04]), which is easier to establish and holds without any further restrictions on our framework.

Theorem 6.4.9

Behavioural equivalence implies logical equivalence $\equiv_{L}$.

Proof: Let $c, c' \in |C|$ be two behaviourally equivalent states, i.e., there is a coalgebra homomorphism $(C, \alpha) \xrightarrow{f} (D, \beta)$ such that $|f|c = |f|c'$. First we claim that for any formula $\varphi \in \mathbb{M}_{\Lambda}$ we have $[\![\varphi]\!]_{\alpha} = f^{*}[\![\varphi]\!]_{\beta}$. We prove this by structural induction on φ.

Without loss of generality, let $\varphi = [\lambda]\varphi'$. Then,

$$[\![\varphi]\!]_\alpha = \alpha^* \lambda_C [\![\varphi']\!]_\alpha = \alpha^* \lambda_C f^* [\![\varphi']\!]_\beta = f^* \beta^* \lambda_D [\![\varphi']\!]_\beta = f^* [\![[\lambda]\varphi']\!]_\beta.$$

The second equality from the right in the above equation is due to the following two commutative squares from the left. The left square commutes due to the naturality of λ, the middle square commutes since f is a coalgebra homomorphism, and the right one commutes due to the naturality of ω.

$$\begin{array}{ccccccc}
\Phi C & \xrightarrow{\lambda_C} & \Phi FC & \xrightarrow{\alpha^*} & \Phi C & \xrightarrow{\omega_C} & \tilde{\mathcal{Q}}|C| \\
{\scriptstyle f^*}\uparrow & & {\scriptstyle Ff^*}\uparrow & & {\scriptstyle f^*}\uparrow & & \uparrow{\scriptstyle |f|^{-1}} \\
\Phi D & \xrightarrow[\lambda_D]{} & \Phi FD & \xrightarrow[\beta^*]{} & \Phi D & \xrightarrow[\omega_D]{} & \tilde{\mathcal{Q}}|D|
\end{array}$$

Furthermore, using the naturality of ω, i.e., $\omega_C \circ f^* = |f|^* \circ \omega_D$, we find that

$$\begin{aligned}
c \models_\alpha \varphi &\iff c \in \omega_C f^* [\![\varphi]\!]_\beta \\
&\iff c \in |f|^{-1} \omega_D [\![\varphi]\!]_\beta \\
&\iff c' \in |f|^{-1} \omega_D [\![\varphi]\!]_\beta \\
&\iff c' \models_\alpha \varphi.
\end{aligned}$$

□

Finally, we show that logical equivalence $\equiv_L$ implies behavioural equivalence. We give a proof sketch, where the details of this proof (see Appendix A.4) become similar to the one of the analogue theorem in Chapter 3 (see Appendix A.3) once we realize how to construct the coequalizer from the underlying logical equivalence (see previous Section 6.4.1). Recall, that in **Set** the main point is given by the separating set of predicate liftings property. Here we combine this property with Lemma 6.4.5 and require that ω preserves intersections and unions (i.e. preserves fibred (co)limits).

Theorem 6.4.10

If $\Phi \xrightarrow{\omega} \tilde{\mathcal{Q}} \circ |\_|$ preserves fibred (co)limits and is injective on objects, Λ is separating for F, and there is a reflective subcategory $\mathbf{B}$ of $\mathbf{C}$ having all the coequalizers and F preserves $\mathbf{B}$, then logical equivalence implies behavioural equivalence.

The proof starts with the fact that given for any $U \in \Phi D$ we have a pair $c, c' \in |C|$ such that $c \in \omega_C(g^*U)$ and $c' \notin \omega_C(g^*U)$ then $c \not\equiv_L c'$ holds which can be easily

shown via contradiction. Next, since $c \not\equiv_L c'$ holds we use the distinguishing formula of c vs. c' given by:

$$\varphi_U = \bigwedge_{c' \notin \omega_C(g^*U)} \bigvee_{c \in \omega_C(g^*U)} \varphi_{c,c'}$$

Finally, given the transpose $g : C \to D$ of the constructed coequalizer f the main idea is to show that $|Fg||\alpha|(c)$ and $|Fg||\alpha|(c')$ can not be distinguished by any pair (λ, U) with $\lambda \in \Lambda$, $U \in \Phi D$ in case $(c, c') \in \equiv_L$ (i.e. c, c' satisfy the same formulas).

To obtain this result above we continue our proof in showing the following claim:

$$\hat{c} \models_\alpha \varphi_U \iff \hat{c} \in \omega_C(g^*U), \qquad \text{for any } \hat{c} \in |C|$$

6.4.3 The Game-Theoretical Perspective

This section presents a generalization of the game variant discussed in Section 3.3.1 for coalgebras living in **Set** where the predicate liftings are directly integrated into the game rules. More specifically, here we are (again) dealing with spoiler/duplicator games. Recall, that the spoiler (**S**) is often referred to in the literature as an attacker. This is because **S** denounces the statement that two states have the same behaviour. The opposition, so to speak, is the duplicator or defender (**D**) since **D** must be able to *mimic* every move of **S** (see Section 3.3.1).

The intuition to formulate game-rules using *predicates* arises from the need to specify target states rather than a single state already established in [KM18; DLT08] and explained in more detail via Example 3.3.5. Moreover, adding predicate liftings into the game naturally follows from the fact that predicate liftings are omnipresent in coalgebraic modal logic.

Since predicate liftings are indexed morphisms and our framework allows to take a more abstract view on the material, we first present a game characterization based on indexed categories and later we show how we close the gap between notions of bisimulation and trace semantics with respect to games (which brings us back to Kleisli categories).

Definition 6.4.11

Given a coalgebra $C \xrightarrow{\alpha} FC$, a set of chosen predicate liftings Λ, and two distinct states $c, c' \in |C|$, then the game works as follows from the current game instance (c, c'):

- **Step 1:** Spoiler chooses a state $s \in \{c, c'\}$ and a predicate $U \in \Phi C$.
- **Step 2:** Duplicator picks the remaining state t (i.e., $t = c$ if $s = c'$ or $t = c'$

if $s = c$) and a predicate $U' \in \Phi C$ such that: $|\alpha|(s) \in \omega_{FC}(\lambda_C U) \implies |\alpha|(t) \in \omega_{FC}(\lambda_C U')$ for all $\lambda_C \in \Lambda$.

- **Step 3:** Spoiler chooses a predicate $\bar{U} \in \{U, U'\}$ and a state $d \in |C|$ such that $d \in \omega_C(\bar{U})$.
- **Step 4:** Duplicator chooses a state $d' \in |C|$ with the remaining predicate $\bar{U}'$ such that $d' \in \omega_C(\bar{U}')$.

The game instance changes to (d, d'). **D** wins the game if the game continues forever or if **S** cannot perform Step 3. **S** wins the game whenever **D** has no moves at Step 2 or Step 4.

Analogous to the games presented in Chapter 3 and Chapter 5, we also start here with the simpler part of the proof. However, this time we need an additional requirement to prove soundness. Remember that the coalgebra and both the predicate and the lifted predicate live in ΦC.

Consider the following initial situation, where a pair of behaviourally equivalent states (c, c') is given and **S** has chosen a predicate U in Step 1. Unfortunately, in general we can not simply assume that the construction of the answer-move in Step 2 by **D** works as in the classical case. We recall the situation in Figure 6.17 for a given coalgebra $\alpha : X \to FX \in$ **Set** (see proof of Theorem 3.3.2). Assume, that a coalgebra homomorphism $f : X \to Y$ and the move p_1 of **S** are given. The move p_2 of **D** is constructed via the characteristic function of $f^{-1}(f[\hat{p}_1]) \subseteq X$ (where $\hat{p}$ denotes the subset induced by the characteristic function p).

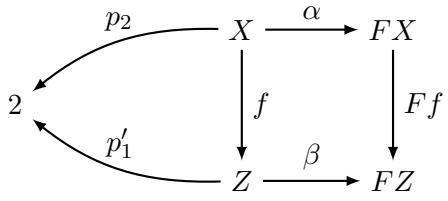


Figure 6.17: Given a coalgebra homomorphism f and the spoiler move p_1, we construct duplicators move $p_2 = p_1' \circ f$ where p_1' is the characteristic function of $f[\hat{p}_1]$ in **Set**.

Since in general $f_!U \in \Phi Y$ is not guaranteed (see Example 2.3.45) , taking the closure of an abstract predicate U choosen by **S** in Step 1 with respect to some coalgebra homomorphism f requires the bifibration property (see proposition 6.2.4). Additionally, we also require that our functor ω_C preserves the bifibration structure

of the indexed categories Φ and $\tilde{\mathcal{Q}}$. This restriction is useful in answering Step 4 for the duplicator.

$$\begin{array}{ccc} \Phi C & \xrightarrow{\omega_C} & \tilde{\mathcal{Q}}|C| \\ {\scriptstyle \exists_f}\downarrow & (6.4) & \downarrow{\scriptstyle |f|_!} \\ \Phi D & \xrightarrow{\omega_D} & \tilde{\mathcal{Q}}|D| \end{array}$$

Theorem 6.4.12

Let $\mathbf{C}^{\mathrm{op}} \xrightarrow{\Phi} \mathbf{cBA}_\wedge$ be an indexed category having the bifibration property such that Square (6.4) commutes for every $f \in \mathbf{C}$. If $c, c' \in |C|$ of a coalgebra $C \xrightarrow{\alpha} FC \in \mathbf{C}$ are behaviourally equivalent then the duplicator wins the game from (c, c').

Proof: Let $c, c' \in |C|$ be behaviourally equivalent states, i.e., there is some coalgebra homomorphism $(C, \alpha) \xrightarrow{f} (D, \beta)$ such that $|f|c = |f|c'$. Suppose the spoiler chooses the state c (the symmetric case when the spoiler chooses c' is similar) and a predicate $U \in \Phi C$. Since $\mathbb{E}(\Phi) \longrightarrow \mathbf{C}$ has the bifibration property, there is an adjunction between the fibre categories:

$$\frac{\exists_f U \longrightarrow V \in \Phi D}{U \longrightarrow f^* V \in \Phi C} \qquad (U \in \Phi C, V \in \Phi D).$$

Thus, from the unit of this adjunction, we take the transpose of the identity map $\exists_f U \longrightarrow \exists_f U$ and get

$$U \longrightarrow f^* \exists_f U \in \Phi C.$$

So, we let the duplicator choose the state c' and fix $U' = f^* \exists_f U$. Next, it suffices to show that the following two claims hold.

$$\omega_{FC}(\lambda_C U) \subseteq \omega_{FC}(\lambda_C U'), \tag{6.5}$$

$$|\alpha|c \in \omega_{FC}(\lambda_C U') \iff |\alpha|c' \in \omega_{FC}(\lambda_C U'). \tag{6.6}$$

For (6.5), let $U \xrightarrow{p} U' \in \Phi C$ be the transpose of the identity map $\exists_f U \longrightarrow \exists_f U$ in the adjunction $\exists_f \dashv f^*$. Then, we get by λ_C, ω_{FC}

$$\omega_{FC}(\lambda_C U) \xrightarrow{\omega_{FC}(\lambda_C p)} \omega_{FC}(\lambda_C U') \in \tilde{\mathcal{Q}}|FC| \implies \omega_{FC}(\lambda_C U) \subseteq \omega_{FC}(\lambda_C U').$$

Therefore, $c \in |\alpha|^{-1} \circ \omega_{FC}(\lambda_C U) \Rightarrow c \in |\alpha|^{-1} \circ \omega_{FC}(\lambda_C U')$. For (6.6), consider the diagram in (6.7) and note that its commutativity follows from the naturality of λ, ω,

and the fact that f is a coalgebra homomorphism.

$$\begin{array}{ccccccc} \Phi C & \xrightarrow{\lambda_C} & \Phi FC & \xrightarrow{\omega_{FC}} & \tilde{\mathcal{Q}}|FC| & \xrightarrow{|\alpha|^{-1}} & \tilde{\mathcal{Q}}|C| \\ \uparrow{\scriptstyle f^*} & & \uparrow{\scriptstyle \Phi Ff} & & \uparrow{\scriptstyle \tilde{\mathcal{Q}}|Ff|} & & \uparrow{\scriptstyle \tilde{\mathcal{Q}}|f|} \\ \Phi D & \xrightarrow[\lambda_D]{} & \Phi FD & \xrightarrow[\omega_{FD}]{} & \tilde{\mathcal{Q}}|FD| & \xrightarrow[|\beta|^{-1}]{} & \tilde{\mathcal{Q}}|D| \end{array} \tag{6.7}$$

Thus, we find that

$$\begin{aligned} |\alpha|c \in \omega_{FC}(\lambda_C f^* \exists_f U) &\iff c \in |\alpha|^{-1}\omega_{FC}\lambda_C f^*(\exists_f U) \\ &\iff c \in \tilde{\mathcal{Q}}|f| \circ \tilde{\mathcal{Q}}|\beta| \circ \omega_{FD} \circ \lambda_D(\exists_f U) \\ &\overset{(6.7)}{\iff} |\beta|(|f|c) \in \omega_{FD} \circ \lambda_D(\exists_f U) \\ &\iff |\beta|(|f|c') \in \omega_{FD} \circ \lambda_D(\exists_f U) \\ &\overset{(|f|c=|f|c')}{\iff} c' \in \tilde{\mathcal{Q}}|f| \circ \tilde{\mathcal{Q}}|\beta| \circ \omega_{FD} \circ \lambda_D(\exists_f U) \\ &\iff c' \in \tilde{\mathcal{Q}}(|\alpha|) \circ \omega_{FC} \circ \lambda_C \circ \Phi(f)(\exists_f U) \\ &\overset{(6.7)}{\iff} |\alpha|c' \in \omega_{FC} \circ \lambda_C \circ \Phi(f)(\exists_f U). \end{aligned}$$

Now, without loss of generality, suppose Spoiler chooses a predicate $\bar{U} \in \{U, U'\}$ and a state $\bar{c} \in |C|$ such that $\bar{c} \in \omega_C \bar{U}$. Then we distinguish the following cases:

1. Let $\bar{U} = U$. Clearly, by construction we have $U \xrightarrow{p} U'$ in ΦC. Moreover, $\omega_C U \xrightarrow{\omega_C(p)} \omega_C U' \in \tilde{\mathcal{Q}}|C|$, i.e., $\omega_C U \subseteq \omega_C U'$. So duplicator can choose $d' = d$ and, clearly, we have $d' \in \omega_C U'$.

2. Let $\bar{U} = U'$. Now, suppose spoiler chooses a state $d' \in |C|$ such that $d' \in \omega_C U'$. Then, $d' \in \omega_C(f^* \exists_f U)$ and we derive

$$\begin{aligned} d' \in \omega_C \Phi f(\exists_f U) &\implies d' \in \tilde{\mathcal{Q}}|f|\omega_D(\exists_f U) \\ &\implies |f|d' \in \omega_D \exists_f U \\ &\implies |f|d' \in |f|_! \omega_C U \end{aligned}$$

   Notice that the leftmost implication is due to the naturality of ω and the rightmost implication is due to (6.4). Thus, we find some $d \in \omega_C U$ such that $|f|d = |f|d'$.

□

For the converse, we work with the relation $\equiv_G \subseteq |C| \times |C|$ defined as follows

$$\equiv_G = \{(c, c') \mid D \text{ has a winning strategy from the instance } (c, c')\}$$

Note that we first have to prove that $\equiv_G$ is an equivalence relation.

Lemma 6.4.13
The above relation $\equiv_G$ is an equivalence relation.

Proof: As is usual, we only need to consider the following three properties:

- $\equiv_G$ is reflexive: (c, c) for every $c \in |C|$.
  Assume **S** chooses c and U then **D** chooses c and U as well, for which we clearly have $c \in |\alpha|^{-1} \circ \omega_{FC}(\lambda_C U) \Rightarrow c \in |\alpha|^{-1} \circ \omega_{FC}(\lambda_C U)$. Then the next game situation is (d, d), since **D** can always answer with the same choice made by **S**.

- $\equiv_G$ is symmetric: $c \equiv_G c'$ implies $c' \equiv_G c$.
  If there is a winning strategy for (c, c') there must always be a winning strategy for (c', c), since **S** can choose either c or c'.

- $\equiv_G$ is transitive: if $c \equiv_G c'$ and $c' \equiv_G b$, then $c \equiv_G b$.

  Assume **S** chooses c and U (the case where **S** chooses c' is analogous, taking into account that $\equiv_G$ is symmetric). We know by $c \equiv_G c'$ that some U' for **D** exists with $c \in |\alpha|^{-1} \circ \omega_{FC}(\lambda_C U) \Rightarrow c' \in |\alpha|^{-1} \circ \omega_{FC}(\lambda_C U')$.

  If **S** were to make the choice of U' and c', we know by $c' \equiv_G b$ that some U'' exists such that $c' \in |\alpha|^{-1} \circ \omega_{FC}(\lambda_C U') \Rightarrow b \in |\alpha|^{-1} \circ \omega_{FC}(\lambda_C U'')$ holds.

  And since implication is transitive we have

  $$c \in \circ|\alpha|^{-1} \circ \omega_{FC}(\lambda_C U) \Rightarrow b \in |\alpha|^{-1} \circ \omega_{FC}(\lambda_C U'')$$

  Now, assume that in Step 3 **S** chooses U, d with $d \in \omega_C(U)$. Again by $(c, c') \in \equiv_G$ we have some answer of **D** who chooses d' with $d' \in \omega_C(U')$ and $d \equiv_G d'$. In addition, if **S** chooses U', d', there is an answer d'' by **D** with $d'' \in \omega_C(U'')$ based on $(c', b) \in \equiv_G$ such that $d' \equiv_G d''$. This state d'' is then finally chosen by **D** in Step 4.

  Now we have the following situation $d \equiv_G d', d' \equiv_G d''$ and we can continue with this strategy for **D**.

□

Theorem 6.4.14

Under the assumptions of Theorem 6.4.10 and assumption (6.4), game equivalence $\equiv_G$ implies behavioural equivalence.

Analogously to the proof in the context of logic the idea is to show that for a state pair $(c, c') \in \equiv_G$ there exits no pair (λ, U) with $U \in \Phi D$ that distinguishes these two states where $D = \mathcal{I}_r B$ is based on the transpose g of the coequalizer $f \in \mathbf{B}$ (see Lemma 6.4.4).

Proof: Following the proof of Theorem 6.4.10 we recall the construction of the witnessing coalgebra homomorphism $C \xrightarrow{g} \mathcal{I}_r B$ with $D = \mathcal{I}_r B$ based on Lemma 6.4.4 and Lemma 6.4.5.

Our aim is to show that $c_1 \equiv_G c_2$ implies that the behaviour of these states coincides i.e $|Fg| \circ |\alpha|(c_1) = |Fg| \circ |\alpha|(c_2)$. Based on Definition 6.2.9 (separation) to prove this, it is sufficient to show that for any $U \in \Phi\mathcal{I}_r B$ we get

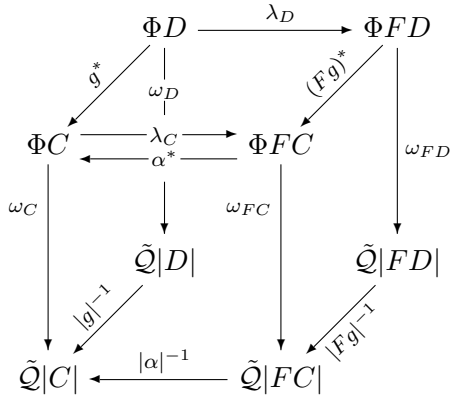
$$|Fg| \circ |\alpha|(c_1) \in \omega_{F\mathcal{I}_r B}(\lambda_{\mathcal{I}_r B}(U)) \Leftrightarrow |Fg| \circ |\alpha|(c_2) \in \omega_{F\mathcal{I}_r B}(\lambda_{\mathcal{I}_r B}(U))$$

with $\lambda_{\mathcal{I}_r B} = \lambda_D$ any predicate lifting living in $\mathbf{C}$. And $\omega_{F\mathcal{I}_r B} = \omega_{FD}$ is the embedding of predicates living in $\Phi F\mathcal{I}_r B = \Phi FD$ into **Set**.

For any $U \in \Phi\mathcal{I}_r B$ with $U_1 = g^*(U)$ and c_1 (the case c_2 is analogous) chosen by the spoiler we get, by the winning strategy of the duplicator some U_2 with

$$|\alpha|(c_1) \in \omega_{FC}(\lambda_C \circ U_1) \Rightarrow |\alpha|(c_2) \in \omega_{FC}(\lambda_C \circ U_2)$$

Next, in Diagram 6.8 we give an overview of several commuting diagrams induced by the naturality of λ, ω combined with the morphism g.



(6.8)

In addition we know that for any $d \in \omega_C(U_i)$ with $i, j \in \{1, 2\}$ duplicator has some $d' \in \omega_C(U_j)$ with $j \neq i$ such that $d \equiv_G d'$ i.e. $|g|d = |g|d'$ holds (see Lemma 6.4.4). Since g is the transpose of the coequalizer $f \in \mathbf{B}$ in the reflective subcategory we get $\omega_C(V) \subseteq \omega_C(g^* U)$ with $U_1 = g^* U$ and $V = U_2$:

Suppose $d \in \omega_C(V)$ then by the winning strategy of the duplicator we know that

$$\begin{aligned}
& d' \in \omega_C(g^*U) & \\
\Leftrightarrow \quad & |g|d' \in \omega_{\mathcal{I}_r B}(U) & \text{(see Diagram 6.8)} \\
\Leftrightarrow \quad & |g|d \in \omega_{\mathcal{I}_r B}(U) & \text{(since } |g|d = |g|d') \\
\Leftrightarrow \quad & d \in \omega_C(g^*U) & \text{(see Diagram 6.8)}
\end{aligned}$$

We now show $V \sqsubseteq g^*U$ with the following two intermediate steps:

1. $|g|_!\ \omega_C(V) = \ |g|_!\ \omega_c(g^*U)$: Based on the winning strategy of the duplicator for $c \equiv_G c'$ we know that for any $d \in \omega_C(V)$ duplicator has some $d' \in \omega_C(g^*U)$ such that $|g|d = |g|d'$ holds (and for any $d' \in \omega_C(g^*U)$ duplicator has some $d \in \omega_C(V)$ such that $|g|d = |g|d'$). Together with the direct image $|g|_!$ we get $|g|_!\omega_C(V) = \ |g|_!\omega_C(g^*U)$.

2. $\exists_g\ V = \exists_g\ g^*U$: Here we mainly work with assumption (6.4), which gives us

$$\omega_{\mathcal{I}_r B}(\exists_g V) \overset{(6.4)}{=} |g|_!\omega_C(V) \quad \text{and} \quad \omega_{\mathcal{I}_r B}(\exists_g g^*U) \overset{(6.4)}{=} |g|_!\omega_C(g^*U)$$

$$\begin{aligned}
\Rightarrow \quad & \omega_{\mathcal{I}_r B}(\exists_g V) = \omega_{\mathcal{I}_r B}(\exists_g g^*(U)) & \text{(see above \& (1))} \\
\Rightarrow \quad & \exists_g V = \exists_g g^*(U) & (\omega_{\mathcal{I}_r B} \text{ injective on objects})
\end{aligned}$$

Now, we can conclude the following again based on assumption (6.4) :

$$\frac{\exists_g V \sqsubseteq \exists_g g^*(U) \in \Phi\mathcal{I}_r B}{V \sqsubseteq g^*\exists_g g^*(U) \in \Phi C} \qquad (V \in \Phi C, U \in \Phi\mathcal{I}_r B)$$

And since we have $g^*\exists_g g^*(U) = g^*(U)$ by adjunction we derive $V \sqsubseteq g^*(U)$.

$$\begin{aligned}
& V \sqsubseteq g^*U \in \Phi C & (\lambda_C \text{ is functor}) \\
\Rightarrow\ & \lambda_C(V) \sqsubseteq \lambda_C(g^*U) \in \Phi C & (\omega_{FC} \text{ is functor}) \\
\Rightarrow\ & \omega_{FC}(\lambda_C(V)) \subseteq \omega_{FC}(\lambda_C(g^*U)) & (6.9)
\end{aligned}$$

Finally, we can derive the assumption mentioned at the beginning, that for any $U \in \Phi\mathcal{I}_r B$ we get

$$|Fg| \circ |\alpha|(c_1) \in \omega_{F\mathcal{I}_r B}(\lambda_{\mathcal{I}_r B}(U)) \Leftrightarrow |Fg| \circ |\alpha|(c_2) \in \omega_{F\mathcal{I}_r B}(\lambda_{\mathcal{I}_r B}(U)).$$

Based on the previous conclusion (6.9) and the fact that this works analogous for the case where spoiler chooses c_2 we get with $U_1 = g^*U$:

$$|\alpha|(c_1) \in \omega_{FC}(\lambda_C \circ U_1) \Rightarrow |\alpha|(c_2) \in \omega_{FC}(\lambda_C \circ U_1)$$
$$|\alpha|(c_2) \in \omega_{FC}(\lambda_C \circ U_1) \Rightarrow |\alpha|(c_1) \in \omega_{FC}(\lambda_C \circ U_1)$$

Since this conclusion holds for all $U \in \Phi\mathcal{I}_r B$ (with $D = \mathcal{I}_r B$) and $U_1 = g^*U$ we finally derive:

$$\begin{aligned}
&(|\alpha|(c_1) \in \omega_{FC}(\lambda_C g^*U)) \Leftrightarrow |\alpha|(c_2) \in \omega_{FC}(\lambda_C g^*U))\\
\Rightarrow\ &(|\alpha|(c_1) \in \omega_{FC}((Fg)^*\lambda_{\mathcal{I}_r B}(U)) \Leftrightarrow |\alpha|(c_2) \in \omega_{FC}((Fg)^*\lambda_{\mathcal{I}_r B}(U)))\\
\Rightarrow\ &(|\alpha|(c_1) \in |Fg|^{-1}\omega_{F\mathcal{I}_r B}(\lambda_{\mathcal{I}_r B}U) \Leftrightarrow |\alpha|(c_2) \in |Fg|^{-1}\omega_{F\mathcal{I}_r B}(\lambda_{\mathcal{I}_r B}U)),
\end{aligned}$$

which is the sufficient property mentioned at the beginning of the proof. □

The following two corollaries follow from the fact, that **Set** and Kleisli categories $\mathbf{K}\ell(T)$ induced by a monad T satisfy (see Lemma 6.2.6) the requirements of both theorems introduced previously.

Corollary 6.4.15

For $\mathbf{C} = \mathbf{B} = \mathbf{Set}$, $\Phi = \tilde{\mathcal{Q}}$, and $\omega = Id_{\tilde{\mathcal{Q}}}$, our game coincides with the game variant based on a separating set of monotone predicate liftings mentioned in Section 3.3.1 [KM18].

Corollary 6.4.16

Given a reflective subcategory $\mathbf{B} \xhookrightarrow{\mathcal{I}_r} \mathbf{K}\ell(T)$ having coequalizers with F preserving this subcategory (i.e, $F \circ \mathcal{I}_r = \mathcal{I}_r \circ F$) and F is separating with respect to Λ, then behaviourally equivalent states are exactly those from which duplicator has a winning strategy.

Finally, we close this section playing a game on one of the two running applications within this chapter.

Example 6.4.17

Here we illustrate how the game works on NDAs. We recall the automaton from Example 6.4.6 drawn in Figure 6.18.

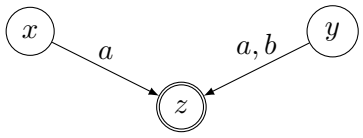


Figure 6.18: An NDA with x, y being not language equivalent.

Consider the initial situation $(\{x\}, \{x, y\})$ (notice the concrete state space is the set $\mathcal{P}\{x, y, z\}$) and **S** chooses $\{x, y\}$ with $U = \{\{z\}\}$, so we have

$$|\alpha|(\{x, y\}) = \{(a, z), (b, z)\} \in \lambda_X^b(\{\{z\}\})$$

To answer this move **D** chooses $\{x\}$ and $U' = \{\emptyset, \{z\}\}$ with

$$|\alpha|(\{x\}) = \{(a, z)\} \in \lambda_X^b(\{\emptyset, \{z\}\})$$

Note here, that the essential component of U' is the $\emptyset$, since no b-transition is observable from $\{x\}$. Next **S** chooses $\emptyset, U'$ and so **D** can only go on with $\{z\}$. Therefore, in Step 1 of the next round **S** selects $\{z\}$ with $|\alpha|(\{z\}) = \{\bullet\}$ and some predicate U and wins in Step 2 due to $\bullet$. **D** is unable to find a valid predicate U' for $|\alpha|(\emptyset) = \emptyset$ satisfying $\emptyset \in \lambda_X^1(U')$.

Note, that since we are interested in the behavioural equivalence of the concrete state pairs we need to play with the predicates obtained by Φ. For the play in Example 6.4.17 this results in a game over the state space $\mathcal{P}X$.

Concerning our second application, since $\mathbf{K}\ell(\mathcal{M}_{\mathbb{F}})$ is the category $\mathbf{Vect}_{\mathbb{F}}$ of vector spaces and linear maps [JSS15] and $\mathbf{Vect}_{\mathbb{F}}$ has all coequalizers (cf. [Bae19]), we may take $\mathbf{K}\ell(\mathcal{M}_{\mathbb{F}})$ as the reflective subcategory of itself.

6.4.4 The Relation between Logic and Games

Winning-strategies can be described intuitively as a scheme of how to play with respect to the move of the opponent in such a way that no matter which move the other player does next, there exists at least one option to win the game.

A formal treatment is studied in Section 3.4 and regarding the proof of Theorem 6.4.12 a winning strategy for the duplicator is (indirectly) described based on the bifibration property and on the witnessing coalgebra homomorphism.

Similar to the classical case in Section 3.3.2 we give a construction to derive the winning strategy for **S** from a modal formula φ which distinguishes two concrete states c_1, c_2 (i.e., $c_1 \models \varphi$ and $c_2 \not\models \varphi$) of a given coalgebra $C \xrightarrow{\alpha} FC$. The spoiler

strategy is defined over the structure of $\varphi \in \mathbb{M}_\Lambda$ (for a fixed set Λ) in the following way:

- If $\varphi = \bigwedge_{i\in I} \varphi_i$ then **S** picks a formula φ_i (for some $i \in I$) such that $c_2 \not\models \varphi_i$.
- If $\varphi = [\lambda]\psi$ (for $\lambda \in \Lambda$), **S** chooses c_1 and $U = [\![\psi]\!]$ in Step 1. After **D** has chosen c_2 and some predicate U' in Step 2, we find that $\omega_C(U') \not\subseteq \omega_C([\![\psi]\!])$. Now in Step 3 **S** chooses U' and a state c_2' with $c_2' \in \omega_C U'$ and $c_2' \not\models \psi$. Then **D** must choose $[\![\psi]\!]$ and a state c_1' with $c_1' \models \psi$ in Step 4 and the game continues with c_1', c_2' and ψ.
- If $\varphi = \neg\psi$, **S** takes ψ and reverses the roles of c_1, c_2.

It can be shown that this strategy is successful for the spoiler to win the game from a pair of behaviourally non-equivalent states.

Theorem 6.4.18
Under all the assumptions of Theorem 6.4.12, the above results in a winning strategy for **S** from a pair of behaviourally different states.

Proof: We prove this by structural induction:

1. Base case $\varphi = [\lambda_C]\psi$: Here we prove that **S** has a winning strategy based on the choice $(c_1, [\![\psi]\!])$. Here we use the fact, that $\omega_C(U_2) \not\subseteq \omega_C([\![\psi]\!]_\alpha)$ with U_2 being a valid answer move by **D** (in the case **D** has no valid move, we are done). We proceed with a proof by contradiction and assume that $\omega_C(U_2) \subseteq \omega_C([\![\psi]\!]_\alpha)$. Similar to the details in the proof of Theorem 6.4.14 we know that

$$\begin{aligned}
& \omega_C(U_2) \subseteq \omega_C([\![\psi]\!]_\alpha) && (U_2 \text{ is a valid move by } D)\\
\Rightarrow\quad & \exists_{|g|}\omega_C(U_2) = \exists_{|g|}\omega_C([\![\psi]\!]_\alpha) && (\text{see } 6.4)\\
\Rightarrow\quad & \omega_{\mathcal{I}_r B}\exists_g(U_2) = \omega_{\mathcal{I}_r B}\exists_g([\![\psi]\!]_\alpha) && (\omega \text{ injective on objects})\\
\Rightarrow\quad & \exists_g(U_2) = \exists_g([\![\psi]\!]_\alpha)\\
\Rightarrow\quad & g^*\exists_g(U_2) = g^*\exists_g(([\![\psi]\!]_\alpha) && (U_2 \sqsubseteq g^*\exists_g(U_2))\\
\Rightarrow\quad & U_2 \sqsubseteq g^*\exists_g(([\![\psi]\!]_\alpha) && (6.10)
\end{aligned}$$

   In addition, since we know U_2 is an valid move by **D**, we have

$$|\alpha|(c_1) \in \omega_{FC}(\lambda_C([\![\psi]\!]_\alpha)) \quad \Rightarrow \quad |\alpha|(c_2) \in \omega_{FC}(\lambda_C(U_2))$$

   and finally we derive based on $[\![\psi]\!]_\alpha = g^*[\![\psi]\!]_\beta$ and the fact that our adjunction $\exists_g \dashv g^*$ is a galois connection including the property $g^*\exists_g g^* = g^*$ [HH90] the

following

$$\begin{aligned} & |\alpha|(c_2) \in \omega_{FC}(\lambda_C(U_2)) && \text{(see 6.10)} \\ \Rightarrow\quad & |\alpha|(c_2) \in \omega_{FC}(\lambda_C(g^* \exists_g (([\![\psi]\!]_\alpha)) \\ \Rightarrow\quad & |\alpha|(c_2) \in \omega_{FC}(\lambda_C(g^* \exists_g g^* [\![\psi]\!]_\beta)) && (g^* \exists_g g^* = g^*) \\ \Rightarrow\quad & |\alpha|(c_2) \in \omega_{FC}(\lambda_C(g^* [\![\psi]\!]_\beta)) \\ \Rightarrow\quad & |\alpha|(c_2) \in \omega_{FC}(\lambda_C([\![\psi]\!]_\alpha)) \end{aligned}$$

But this yields a contradiction $c_2 \models [\lambda_C]\psi$.

Therefore, we go on with $\omega_C(U_2) \not\subseteq \omega_C([\![\psi']\!])$ and this implies we have some state $c_1' \in \omega_C(U_2)$ which does not satisfy ψ'.

2. $\varphi = \bigwedge_{i \in I} \varphi_i$: Then there is some formula φ_i (for some $i \in I$) with $c_2 \not\models \varphi_i$ (i.e. $c_2 \notin [\![\varphi]\!]$), hence **S** chooses c_1 and $[\![\varphi]\!]$.

3. In the case negation exists, we assume it is well behaved $\varphi = \neg\psi$: Then we simply can switch to $c_2 \not\models \neg\psi = \neg(c_2 \models \neg\psi) = \neg(\neg(c_2 \models \psi)) = c_2 \models \psi$ and $c_1 \models \neg\psi$.

□

6.5 Conclusion and Discussion

Coalgebraic behaviour equivalence allows to reason about the states and their behaviour in a high degree of abstraction since one can move to categories different from **Set**. In the sense of system verification the semantics play a major role and one can choose between logical and game-theoretical semantics.

To summarize our contributions:

▷ We close the logical and game-theoretical gap in the scope of the linear time-branching spectrum where in this chapter we focused on language equivalence. In particular, our framework can be applied to derive the characterizations for (weighted) language, trace, failure, and ready semantics [BK+20].

▷ In analogy to the requirements of the categorical work for the linear time-branching spectrum [PT99; JSS15] we give a recipe to derive modalities for Kleisli categories.

However, it still remains unclear how our construction can help to derive the modalities for conditional transition systems (CTS) [BK+17] (e.g. systems present in the Kleisli category induced by the input monad).

Another categorical framework that enables the generalization of classical Hennessy-Milner theorems to coarser notions of behavioural equivalences [DMS19] is graded logics [DMS19]. A comparison of Kleisli (Eilenberg-Moore) and graded logics is given in [DMS19].

We already gave an overview of other related approaches (see [KK+19; KR20]) in the introduction. Regarding our indexed categories restricted to boolean algebras, we want to emphasize that the restriction by $CLat_\wedge$-fibrations to complete lattices where each pullback functor preserves all meets corresponds to an indexed category which maps to the category of complete lattices and meet preserving functions [KK+19]. And a complete boolean algebra can be seen as a complete lattice which shows that the restrictions are more or less the same as they are motivated by the nature of logic.

Furthermore, our setting does not capture **Meas** since the given indexed category fails to admit the bifibration property (see Example 2.3.45). A framework which captures **Meas** is given by dual adjunctions where the idea of this concept is sketched in Figure 6.1.

Nevertheless, the results of logical and game-theoretical nature based on fibration and dual adjunction [KR20; KK+19] do not cover trace equivalence. Concerning dual adjunctions, it is not yet clear how to set up this framework to characterize logic for decorated trace in general. For instance, in the case of $\mathbf{C} = \mathbf{K}\ell(\mathcal{P})$, we need a left adjoint for the composition $\mathbf{K}\ell(\mathcal{P})^{\mathrm{op}} \xrightarrow{|\_|} \mathbf{Set}^{\mathrm{op}} \xrightarrow{\tilde{\mathcal{Q}}} \mathbf{cBA}_\wedge$ before we even start to apply this abstract framework. Unfortunately, a left adjoint for $\tilde{\mathcal{Q}} \circ |\_|$ cannot exists since this composition does not preserves limits. In particular, $\tilde{\mathcal{Q}}|\emptyset| \ncong 1$, where $\emptyset$ and 1 are the terminal objects in $\mathbf{K}\ell(\mathcal{P})$ and $\mathbf{cBA}_\wedge$.

Summarizing, concerning categories beyond **Set** there are still some open questions how coalgebraic behavioural equivalence can be characterized via logical or game-theoretical semantics. But all the frameworks mentioned in the introduction (including indexed categories) come closer step by step in answering this *non-trivial* question [KK+19; KR20].

7

Parity Games over Continuous Lattices

Parity games play a major role in the scope of the modal μ-calculus model checking. Analogously to Hintikka's game, we have two players $\exists$ and $\forall$ where the winning strategies of both players represent an alternative characterization of semantics [Sti95; BW18; Hin68].

Continuous lattices have been introduced by Scott [Sco72] in connection with the semantics of the λ-calculus [Sco72], which is equivalent to Turing machines [Roj15; Tur37]. The work by Scott relates to the semantics of programming languages and is generally known as the origin of *domain theory*. Therefore, a suitable notion of approximation as the one given by continuous lattices is of fundamental interest in the theory of computation [GH+03; AJ94; CPA16]

7.1 Introduction

Model checking is one of the most familiar applications, where system of fixpoint equations are omnipresent, since a property expressed via a logical formula can be transformed into such a system [Sei96]. Another area in the scope of program analysis [NNH99] uses the flow graph of a program to derive a system of fixpoint equations, which describe a set of constraints based on the context of the underlying analysis. The properties of interest can be expressed through greatest fixpoints (i.e. invariant/safety properties) or via least fixpoints (i.e. liveness/reachability properties). One of the most famous examples is bisimilarity that can be characterized as the greatest fixpoint as explained for LTS in Section 2.2.2 or studied from a coalgebraic perspective in Chapter 3.

Least and greatest fixpoint can be profitably mixed, in order to obtain expressive specification logics, among which the μ-calculus [Koz83] is a classical example. The μ-calculus is very expressive, but the nesting of fixpoints increases the complexity of model checking. Common approaches to the model checking problem rely on an encoding in terms of parity games (see, e.g., [BW18; Sti95; EJ91]).

The seminal paper [Jur00] provides an algorithm for the solution of parity games which is polynomial in the number of states and exponential in (half of) the alternation

depth, recently improved to quasi-polynomial in [CJ+17]. A detailed discussion of the complexity of μ-calculus model checking can be found in [BW18].

One of the key ingredients in Jurdzisński's algorithm for solving parity games is the concept of progress measures, which can be understood as *witnesses* for winning strategies. It has been already noticed [HSC16], that this key ingredients can be lifted to a more general setting, i.e. system of fixpoint equations over general lattices. The motivation to move from the standard setting given by powerset lattices to other lattices is implied by the fact, that numerous applications require lattices which capture fuzziness, probabilities or as considered in Chapter 5 quantitative information. Although the work in [HSC16] is extending Jurdzisński's approach, the progress measures play a slightly different role in the form of invariants respectively ranking functions, where Jurdzisński focuses on the computation technique.

Inspired by the mentioned work, in this chapter we present a game-theoretical approach to the solution of systems of fixpoint equations over continuous lattices [Sco72]. They include discrete structures, such as most domains used in program analysis, and continuous structures, such as the real interval $[0, 1]$ or the lattice of open sets of a locally compact Hausdorff space.

Similar to the Hintikka/Henkin game with players $\exists$ and $\forall$, there is a probably folklore game between an existential and an universal player, which has been observed in [Ven08] where the game is referred to as an unfolding game.

As a first result, here we show how the unfolding game can be extended to work for a *system* of fixpoint equations over lattices, resulting in a surprisingly simple game that we refer to as a *fixpoint game*.

As already pointed out, on the one hand, the nesting of least and greatest fixpoint equations has some advantages (e.g. highly expressive logics), but on the other hand, the winning conditions of the parity games extend to non-trivial ones.

We start with the idea of our game, which is given by the following observation: for the simpler case of powerset lattices the interaction between the players in the fixpoint game fundamentally relies on the possibility of testing the presence of elements in the image of a set and on the fact that a subset is completely determined by the elements that belongs to it.

When moving to a more general class of lattices we need to ensure that this kind of interaction can be suitably mimicked.

At this stage continuous lattices enter the game, since they provide exactly the necessary machinery as they come equipped with a notion of *finitary* approximation based on the *way-below* relation and each element arises as the join of the elements

(possibly restricted to a selected basis) which are way-below it, in the same way as a subset is the union of its singletons.

Where our paper [BK+19a] also generalizes the work of Jurdziński for solving parity games [Jur00] to systems of fixpoint equations over continuous lattices, in this thesis we concentrate on the presentation of the general game. Therefore, we obtain the following two central points:

- We propose a game-theoretical characterization of the solution of systems of fixpoint equations over lattices and we identify continuous lattices as a general and appropriate setting for such theory.

- We present a theory of progress measures à la Jurdziński in this general framework based on so-called μ- and ν-approximants. In particular, μ-approximants turn out to be closely related to the progress measures of [HSC16].

The chapter starts with Section 7.2, where we recap the basics of continuous lattices and introduce several notations including also orders of tuples and ordinals. Afterwards in Section 7.3, we present systems of fixpoint equations over a lattice and redefine their solution together with a corresponding notion of approximation. In Section 7.4 we consider two different application scenarios: μ-calculus and program analysis. Section 7.5 includes a game-theoretical approach to the solution of a system of equations over a continuous lattice, where we apply our theory to the applications of Section 7.4. Besides, we also compare bisimulation games and modal μ-calculus model checking games with our game. We briefly discuss the concept of progress measures in the penultimate Section 7.6 before we sum everything up in the final Section 7.7.

7.2 Foundations

The following definitions, examples, and theorems build on the lattice-theory presented in Section 2.1 and are partly taken from [DP02; GH+03].

Recall from Section 2.1, that we denote with L a complete lattice. Since all lattices in this chapter will be complete, we will often omit the qualification *complete*.

Definition 7.2.1: Upward-Closure

Given an element $l \in L$ we define its upward-closure

$$\uparrow l = \{l' \mid l' \in L \ \wedge \ l \sqsubseteq l'\}$$

Definition 7.2.2: Basis

A *basis* for a lattice is a subset $B_L \subseteq L$ such that for each $l \in L$ it holds that $l = \bigsqcup\{b \in B_L \mid b \sqsubseteq l\}$.

Definition 7.2.3: Completely Join-Irreducible

An element $l \in L$ is *completely join-irreducible* if whenever $l = \bigsqcup X$ for some $X \subseteq L$ then $l \in X$.

Since we are only interested in completely join-irreducible elements we will often omit the qualification *completely*. Note that $\bot$ is never an irreducible since $\bot = \bigsqcup \emptyset$ and $\bot \notin \emptyset$.

Example 7.2.4

Three simple examples of lattices, that we will refer to later, are:

- The powerset of any set X, ordered by subset inclusion $(\mathcal{P}X, \sqsubseteq)$. Join is union, meet is intersection, top is X and bottom is $\emptyset$. A basis is the set of singletons $B_{\mathcal{P}X} = \{\{x\} \mid x \in X\}$. These are also the join-irreducible elements. Any set $Y \subseteq X$ with $|Y| > 1$ is not irreducible, since $Y = \bigsqcup_{x \in Y}\{x\}$ but clearly $Y \neq \{x\}$ for any $x \in Y$.
- The real interval $[0, 1]$ with the usual order $\leq$. Join and meet are the sup and inf over real numbers, 0 is bottom and 1 is top. The rationals $\mathbb{Q} \cap (0, 1]$ are a basis. There are no irreducible elements (in fact, for any $x \in [0, 1]$ we have that $x = \bigsqcup\{y \mid y < x\}$ and clearly $x \notin \{y \mid y < x\}$).
- Consider the partial order $W = \mathbb{N}_0 \cup \{\omega, a\}$ depicted in Figure 7.1. It is easy to see that it is a lattice. All elements are irreducible apart from the bottom 0 and the top ω. For the latter notice that, e.g., $\omega = \bigsqcup\{1, a\}$.

We will focus on special lattices where elements are generated by suitably defined approximations. Given a lattice L, a subset $X \subseteq L$ is *directed* if $X \neq \emptyset$ and every pair of elements in X has an upper bound in X.

Definition 7.2.5: Way-Below Relation

Let L be a lattice. Given two elements $l, l' \in L$ we say that l is *way-below* l', written $l \ll l'$ when for every directed set $D \subseteq L$, if $l' \sqsubseteq \bigsqcup D$ then there exists $d \in D$ such that $l \sqsubseteq d$. We denote by $\downarrow l$ the set of elements way-below l, i.e.,

$\downarrow l = \{l' \mid l' \in L \ \wedge \ l' \ll l\}$.

Definition 7.2.6: Continuous Lattices

The lattice L is called *continuous* if $l = \bigsqcup \downarrow l$ for all $l \in L$.

Intuitively, the way-below relation captures a form of finitary approximation: if one imagines that $\sqsubseteq$ is an order on the information content of the elements, then $x \ll y$ means that whenever y can be *covered* by joining (possibly small) pieces of information, then x is covered by one of those pieces. Then a lattice is continuous if any element can be built by joining its approximations.

Example 7.2.7: Non-continuous lattice

The lattice $W = \mathbb{N}_0 \cup \{\omega, a\}$ in Figure 7.1 is not continuous. In fact, $a \not\ll a$ since $a \sqsubseteq \bigsqcup \mathbb{N}$ but $a \not\sqsubseteq i$ for all $i \in \mathbb{N}_0$. Therefore $\downarrow a = \{0\}$ and thus $a \neq \bigsqcup \downarrow a$.

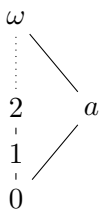


Figure 7.1: A complete lattice W which is not continuous.

Concerning the origin of the name *continuous lattice*, we can quote [Sco72] that says that "One of the justifications (by euphony at least) of the term *continuous lattice* is the fact that such spaces allow for so many continuous functions." For instance, one indication is the fact that meet and join are both continuous in such lattices.

It can be shown that if L is a continuous lattice and B_L is a basis, for all $l \in L$, it holds that $l = \bigsqcup(\downarrow l \cap B_L)$.

Various lattices that are commonly used in semantics enjoy a property stronger than continuity, defined below.

Definition 7.2.8: Compact Element, Algebraic Lattice

Let L be a lattice. An element $l \in L$ is called *compact* whenever $l \ll l$. The lattice L is *algebraic* if the set of compact elements is a basis.

Example 7.2.9

Some examples are as follows:

- All finite lattices are continuous (since every finite directed set has a maximum). More generally, all algebraic lattices (which include all finite lattices) are continuous. The way-below relation is $x \ll y$ if x compact and $x \sqsubseteq y$.
- Given a set X, the powerset lattice $\mathcal{P}(X)$, ordered by inclusion, is an algebraic lattice. The compact elements are the finite subsets. In fact, any set Y is the union of its finite subsets, i.e., $Y = \bigcup\{F \mid F \subseteq Y \ \wedge \ F \text{ finite}\}$. Since $\{F \mid F \subseteq Y \ \wedge \ F \text{ finite}\}$ is a directed set, compactness requires that $Y \subseteq F$ for some finite $F \subseteq Y$, hence $Y = F$. Therefore $Y \ll Z$ holds when Y is finite and $Y \subseteq Z$.
- The interval $[0, 1]$ with the usual order $\leq$ is a continuous lattice. For $x, y \in [0, 1]$, we have $x \ll y$ when $x < y$ or $x = 0$. In fact, each $\emptyset \neq Y \subseteq [0, 1]$ is directed. Imagine that $y \leq \bigsqcup Y$ for such a Y. Then by standard properties of the reals there always exists a $y' \in Y$ such that $x \leq y'$ if and only if $x < y$ or $x = 0$. Note that this lattice is not algebraic since the only compact element is 0.

Tuples and Ordinals We will often consider tuples of elements. Given a set A, an n-tuple in A^n will be denoted by a boldface letter $\boldsymbol{a}$. The components of a tuple $\boldsymbol{a}$ will be denoted by using the same name of the tuple, not in boldface style and with an index, i.e., $\boldsymbol{a} = (a_1, \ldots, a_n)$. For an index $n \in \mathbb{N}$ we use the notation $\underline{n}$ to denote the integer interval $\{1, \ldots, n\}$. Given $\boldsymbol{a} \in A^n$ and $i, j \in \underline{n}$ we write $\boldsymbol{a}_{i,j}$ for the subtuple $(a_i, a_{i+1}, \ldots, a_j)$.

Definition 7.2.10: Pointwise Order

Given a lattice $(L, \sqsubseteq)$ we will denote by $(L^n, \sqsubseteq)$ the set of n-tuples endowed with the *pointwise order* defined, for $\boldsymbol{l}, \boldsymbol{l'} \in L^n$, by $\boldsymbol{l} \sqsubseteq \boldsymbol{l'}$ if $l_i \sqsubseteq l'_i$ for all $i \in \underline{n}$.

The structure $(L^n, \sqsubseteq)$ is a lattice and it is continuous if L is continuous, with the way-below relation given by $\boldsymbol{l} \ll \boldsymbol{l'}$ iff $l_i \ll l'_i$ for all $i \in \underline{n}$ [GH+03, Proposition I-2.1]. More generally, for any set X, the set of functions $L^X = \{f \mid f \colon X \to L\}$, endowed with pointwise order, is a lattice (continuous when L is).

Definition 7.2.11: Lexicographic Order

Given a partial order $(P, \sqsubseteq)$ we will denote by $(P^n, \preceq)$ the set of n-tuples endowed with the *lexicographic order* (where the last component is the most relevant), i.e., inductively, for $\boldsymbol{l}, \boldsymbol{l}' \in P^n$, we let $\boldsymbol{l} \preceq \boldsymbol{l}'$ if either $l_n \sqsubset l'_n$ or $l_n = l'_n$ and $\boldsymbol{l}_{1,n-1} \preceq \boldsymbol{l}'_{1,n-1}$.

When $(L, \sqsubseteq)$ is a lattice also $(L^n, \preceq)$ is a lattice. Given a set $X \subseteq L^n$, the meet of X with respect to $\preceq$ can be obtained by taking the meet of the single components, from the last to the first, i.e., it is defined inductively as $\bigsqcap X = \boldsymbol{l}$ where $l_i = \bigsqcap\{l'_i \mid \boldsymbol{l}' \in X \wedge \boldsymbol{l}'_{i+1,n} = \boldsymbol{l}_{i+1,n}\}$. The join can be defined analogously. Similarly, one can show that $\preceq$ is a well-order whenever $\sqsubseteq$ is.

As in [Jur00; HSC16], we will also need to consider tuples with a preorder arising from the lexicographic order, when some components are considered irrelevant.

Definition 7.2.12: Truncated Lexicographic Order

Let $(P, \sqsubseteq)$ be a partial order and let $n \in \mathbb{N}$. For $i \in \underline{n}$ we define a preorder $\preceq_i$ on P^n by letting, for $\boldsymbol{x}, \boldsymbol{y} \in P^n$, $\boldsymbol{x} \preceq_i \boldsymbol{y}$ if $\boldsymbol{x}_{i,n} \preceq \boldsymbol{y}_{i,n}$. We write $=_i$ for the equivalence induced by $\preceq_i$ and $\boldsymbol{x} \prec_i \boldsymbol{y}$ for $\boldsymbol{x} \preceq_i \boldsymbol{y}$ and $\boldsymbol{x} \neq_i \boldsymbol{y}$. Whenever $\sqsubseteq$ is a well-order, given $X \subseteq P^n$ with $X \neq \emptyset$ and $i \in \underline{n}$, we write $\min_{\preceq_i} X$ for the vector $\boldsymbol{x} = (\bot, \dots, \bot, x_i, \dots, x_n)$ where $\boldsymbol{x}_{i,n} = \min_{\preceq}\{\boldsymbol{l}_{i,n} \mid \boldsymbol{l} \in X\}$.

In words, $\preceq_i$ is the lexicographic order restricted to the components $i, i+1, \dots, n$. For instance, if $P = \mathbb{N}$ with the usual order, then $(6,1,4,7) \prec_2 (5,2,4,7)$, while $(6,1,4,7) =_3 (5,2,4,7)$.

We denote *ordinals* by Greek letters $\alpha, \beta, \gamma, \dots$ and their order by $\leq$. The collection of all ordinals is well-ordered. Given any ordinal α, the collection of ordinals dominated by α is a set $[\alpha] = \{\lambda \mid \lambda \leq \alpha\}$, which, seen as an ordered structure, is a lattice. Meet and join of a set X of ordinals will be denoted by $\inf X$ (which equals $\min X$ if $X \neq \emptyset$) and $\sup X$. The lattice $[\alpha]$ is completely distributive, which follows from classical results. In fact, the complete join-irreducibles are all ordinals which are not limit ordinals. Hence, from [Ran52, Theorems 1 and 2], since every element is the join of completely join-irreducible elements, we can conclude that $[\alpha]$ is completely distributive. A similar argument shows that, for a fixed $n \in \mathbb{N}$ and ordinal α, the lattice of n-tuples of ordinals, referred to as *ordinal vectors*, endowed with the lexicographic order $([\alpha]^n, \preceq)$ is completely distributive. In fact, the only elements that are not complete join-irreducibles are vectors of the kind $(0, \dots, 0, \alpha, \beta_i, \dots, \beta_n)$ where α is a limit ordinal and such vectors can be obtained as

the join of the vectors $(0, \ldots, 0, \beta, \beta_i, \ldots, \beta_n)$, with $\beta < \alpha$ and β a successor ordinal.

Parity Games Since our game is a parity game we introduce a formal definition derived from [Jur00] and proceed with an informal description of the rules.

Definition 7.2.13: Parity Games [Jur00]

A parity graph $G = (V, E, pr)$ is given by a directed graph (V, E) and a priority function $pr : V \to \underline{n}$ where $n \in \mathbb{N}$. A parity game $(V, E, pr, (V_\Diamond, V_\Box))$ consists of a game graph $G = (V, E, pr)$ and of a partition $(V_\Diamond, V_\Box)$ with $V = V_\Diamond \cup V_\Box$.

It is common to assume, that the game graphs are finite and the positions of the existential player Eve ($\exists$) are given by $V_\Diamond$ where $V_\Box$ includes the positions of the universal player Adam ($\forall$). From a position $b \in V_\Diamond \cup V_\Box$ the corresponding player has to choose one of the available successors according to E. In case a player has no option to proceed from her (his) position, the opponent is winning. Therefore, it is simple to determine the winner of such a finite play, where for infinite plays $seq = v_0 v_1 \ldots v_i \ldots$ the *parity* condition determines the winner. Given an infinite sequence seq of moves, we define

$$inf(seq) = \{j \in \underline{n} \mid \text{there are infinitly many i} \in seq \text{ with } j = pr(v_i)\},$$

and $\max(inf(seq))$ determines the winner, i.e. in case the highest priority in (seq) occurring infinitely often is even the existential player $\exists$ is winning. Otherwise, $\forall$ is the winner.

7.3 Fixpoint Equations: Solutions and Approximants

We focus on systems of (fixpoint) equations over some lattice, where, for each equation one can be interested either in the least or in the greatest solution. We define the solution of a system and we devise some results concerning its approximations that will play a major role later.

Almost invariably, in the mentioned settings, the involved functions are monotone and the domains of interest are complete lattices where the key result for deriving the existence of (least or greatest) fixpoints is Knaster-Tarski's fixpoint theorem [Tar55].

Definition 7.3.1: System of Equations

Let L be a lattice. A system of equations E over L is a list of equations of the

following form

$$\begin{aligned} x_1 &=_{\eta_1} f_1(x_1, \ldots, x_m) \\ &\ldots \\ x_m &=_{\eta_m} f_m(x_1, \ldots, x_m) \end{aligned}$$

where $f_i \colon L^m \to L$ are monotone functions and $\eta_i \in \{\mu, \nu\}$. The system will often be denoted as $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$, where $\boldsymbol{x}$, $\boldsymbol{\eta}$ and $\boldsymbol{f}$ are the obvious tuples. We denote by $\emptyset$ the system with no equations.

Systems of equations of this kind have been considered by various authors, e.g., [BC+97; CKS92; Sei96; HSC16]. In particular, [HSC16] works on general lattices.

We next define the pre-solutions of a system as tuples of lattice elements that, replacing the variables, satisfy all the equations of the system. The solution will be a suitably chosen pre-solution, taking into account also the η_i annotations that specify for each equation whether the least or greatest solution is required.

Definition 7.3.2: Pre-Solution

Let L be a lattice and let E be a system of equations over L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$. A *pre-solution* of E is any tuple $\boldsymbol{u} \in L^m$ such that $\boldsymbol{u} = \boldsymbol{f}(\boldsymbol{u})$.

Note that $\boldsymbol{f}$ can be seen as a function $\boldsymbol{f} \colon L^m \to L^m$. In this view, pre-solutions are the fixpoints of $\boldsymbol{f}$. Since all components f_i are monotone, also $\boldsymbol{f}$ is monotone over $(L^m, \sqsubseteq)$. Then, it is well-known that the set of fixpoints of $\boldsymbol{f}$, i.e., the pre-solutions of the system, are a sublattice. In order to define the solution of a system we need some further notation.

Definition 7.3.3: Substitution

Given a system E of m equations over a lattice L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$, an index $i \in \underline{m}$ and $l \in L$ we write $E[x_i := l]$ for the system of $m-1$ equations obtained from E by removing the i-th equation and replacing x_i by l in the other equations, i.e., if $\boldsymbol{x} = \boldsymbol{x}' x_i \boldsymbol{x}''$, $\boldsymbol{\eta} = \boldsymbol{\eta}' \eta_i \boldsymbol{\eta}''$ and $\boldsymbol{f} = \boldsymbol{f}' f_i \boldsymbol{f}''$ then $E[x_i := l]$ is $\boldsymbol{x}'\boldsymbol{x}'' =_{\boldsymbol{\eta}', \boldsymbol{\eta}''} \boldsymbol{f}'\boldsymbol{f}''(\boldsymbol{x}', l, \boldsymbol{x}'')$.
Let $f[x_i := l] \colon L^{m-1} \to L$ be defined by $f[x_i := l](\boldsymbol{x}', \boldsymbol{x}'') = f(\boldsymbol{x}', l, \boldsymbol{x}'')$. Then, explicitly, the system $E[x_i := l]$ has $m-1$ equations,

$$x_j =_{\eta_j} f_j[x_i := l](\boldsymbol{x}', \boldsymbol{x}'') \qquad j \in \{1, \ldots, i-1, i+1, \ldots, n\}$$

We can now recursively define the solution of a system of equations. The notion

is the same as in [HSC16], although we find it convenient to adopt a more succinct formulation

Definition 7.3.4: Solution

Let L be a lattice and let E be a system of m equations on L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$. The *solution* of E, denoted $sol(E) \in L^m$, is defined inductively as follows:

$$\begin{aligned} sol(\emptyset) &= () \\ sol(E) &= (sol(E[x_m := u_m]),\, u_m) \text{ where} \\ &\quad\; u_m = \eta_m(\lambda x.\, f_m(sol(E[x_m := x]), x)) \end{aligned}$$

The i-th component of the solution will be denoted $sol_i(E)$.

Here we show that the definition of the solution of an equational system in [HSC16] is equivalent to our Definition 7.3.4. In both definitions the solution $\boldsymbol{u} = (u_1, \ldots, u_m)$ is solved recursively based on interim solutions by calculating fixpoints.

Definition 7.3.5: Solution of an Equational System [HSC16]

Let L be a lattice and let E be a system of $m \geq 1$ equations on L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$. For each $i \in \underline{m}$ and $j \in \underline{i}$ we define monotone functions $f^{\ddagger}\colon L^{m-i+1} \to L$ and $l_j^{(i)}\colon L^{m-i} \to L$ as follows, inductively on i:

1. $i = 1$:

$$\begin{aligned} f_1^{\ddagger}(l_1, \ldots, l_m) &:= f_1(l_1, \ldots, l_m) \\ l_1^{(1)}(l_2, \ldots, l_m) &:= \eta_1[f_1^{\ddagger}(\_, l_2, \ldots, l_m)\colon L \to L] \end{aligned}$$

   with $\eta_1 \in \{\mu, \nu\}$.

2. $i = i + 1$:

$$\begin{aligned} f_{i+1}^{\ddagger}(l_{i+1}, \ldots, l_m) &:= f_{i+1}(l_1^{(i)}(l_{i+1}, \ldots, l_m), \ldots, l_i^{(i)}(l_{i+1}, \ldots, l_m), l_{i+1}, \ldots, l_m) \\ l_{i+1}^{(i+1)}(l_{i+2}, \ldots, l_m) &:= \eta_{i+1}[f_{i+1}^{\ddagger}(\_, l_{i+2}, \ldots, l_m)\colon L \to L] \end{aligned}$$

   with $\eta_{i+1} \in \{\mu, \nu\}$. The $l_{i+1}^{(i+1)}$ solution is then used to obtain the $(i+1)$-th interim solutions for each $j \in \underline{i}$:

$$l_j^{(i+1)}(l_{i+2}, \ldots, l_m) := l_j^{(i)}(l_{i+1}^{(i+1)}(l_{i+2}, \ldots, l_m), l_{i+2}, \ldots, l_m)$$

Proposition 7.3.6

Let L be a lattice and let E be a system of $m \geq 1$ equations on L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$. Then the solution from Definition 7.3.5 coincides with the solution from Definition 7.3.4.

Proof: Let $l_{i+1}, \ldots, l_m \in L$ be given. We show

$$l_j^{(i)}(l_{i+1}, \ldots, l_m) = sol_j(E[\boldsymbol{x}_{i+1,m} := \boldsymbol{l}_{i+1,m}])$$

for $j \in \underline{i}$ by induction on i:

1. $i = 1$: We define $E' = E[\boldsymbol{x}_{2,m} := \boldsymbol{l}_{2,m}]$ and according to Definition 7.3.4 we have

   $$sol(E[\boldsymbol{x}_{2,m} := \boldsymbol{l}_{2,m}]) = sol(E') = sol(E'[x_1 := u_1]), u_1) = (u_1)$$

   where $u_1 = \eta_1(\lambda x.\, f_1(x))$. In Definition 7.3.5 for $i = 1$ we only have to consider $l_1^{(1)}(l_2, \ldots, l_m) = \eta_1[f_1^{\ddagger}(\_, l_2, \ldots, l_m)] = \eta_1[f_1(\_, l_2, \ldots, l_m)]$ which corresponds to $sol_1(E[\boldsymbol{x}_{2,m} := \boldsymbol{l}_{2,m}]) = u_1 = \eta_1(\lambda x.\, f_1(x))$.

2. $i \to i + 1$: We define $E' = E[\boldsymbol{x}_{i+2,m} := \boldsymbol{l}_{i+2,m}]$. Here we need to distinguish two cases to prove that $l_j^{(i+1)}(l_{i+2}, \ldots, l_m) = sol_j(E')$ for all $j \in \underline{i}$.

   (a) $j = i + 1$: From Definition 7.3.4 we have

   $$sol(E[\boldsymbol{x}_{i+2,m} := \boldsymbol{l}_{i+2,m}]) = sol(E') = sol(E'[x_{i+1} := u_{i+1}]), u_{i+1})$$

   where $u_{i+1} = \eta_{i+1}(\lambda x.\, f_{i+1}(sol(E'[x_{i+1} := x]), x, l_{i+2}, \ldots, l_m))$.
   Hence $sol_{i+1}(E[\boldsymbol{x}_{i+2,m} := \boldsymbol{l}_{i+2,m}]) = u_{i+1}$ and from Definition 7.3.5 we obtain

   $$l_{i+1}^{(i+1)}(l_{i+2}, \ldots, l_m) = \eta_{i+1}[f_{i+1}^{\ddagger}(x, l_{i+2}, \ldots, l_m)]$$

   where $f_{i+1}^{\ddagger}(x, l_{i+2}, \ldots, l_m)$ is defined by

   $$f_{i+1}(l_1^{(i)}(x, l_{i+2}, \ldots, l_m), \ldots, l_i^{(i)}(x, l_{i+2}, \cdots, l_m), x, l_{i+2} \ldots, l_m).$$

   From the induction hypothesis it follows that

   $$\begin{aligned} l_j^{(i)}(x, l_{i+2}, \ldots, l_m) &= sol_j(E[\boldsymbol{x}_{i+1,m} := x, \boldsymbol{l}_{i+2,m}]) \\ &= sol_j(E'[x_{i+1} := x]) \qquad\qquad = l_j \end{aligned}$$

   for $j \in \underline{i}$. We define $(l_1, \ldots, l_i) = sol(E'[x_{i+1} := x])$ and observe that $l_{i+1}^{(i+1)}(l_{i+2}, \ldots, l_m)$ is the η_i-fixpoint of $\lambda x. f_{i+1}(l_1, \ldots l_i, x, l_{i+2}, \ldots, l_m)$. The same is true for u_{i+1} and hence we conclude

   $$l_{i+1}^{(i+1)}(l_{i+2}, \ldots, l_m) = u_{i+1} = sol_{i+1}(E').$$

(b) $j \leq i$: First, from Definition 7.3.4 we obtain

$$sol_j(E') = sol_j(E[\boldsymbol{x}_{i+2,m} := \boldsymbol{l}_{i+2,m}]) = sol_j(E[\boldsymbol{x}_{i+1,m} := \boldsymbol{l}_{i+1,m}])$$

where $l_{i+1} = sol_{i+1}(E')$. From the induction hypothesis we know that $sol_j(E[\boldsymbol{x}_{i+1,m} := \boldsymbol{l}_{i+1,m}]) = l_j^{(i)}(l_{i+1}, l_{i+2}, \ldots, l_m)$. On the other hand we have from Definition 7.3.5 that

$$l_j^{(i+1)}(l_{i+2} \ldots l_m) = l_j^{(i)}(l_{i+1}^{(i+1)}(l_{i+2}, \ldots l_m), l_{i+2}, \ldots, l_m)$$

and from (2a) we finally get $l_{i+1} = l_{i+1}^{(i+1)}(l_{i+2}, \ldots l_m)$.

□

In words, for solving a system of m equations, the last variable is considered as a fixed parameter x and the system of $m-1$ equations that arises from dropping the last equation is recursively solved. This produces an $(m-1)$-tuple parametric on x, i.e., we get $\boldsymbol{u}_{1,m-1}(x) = sol(E[x_m := x])$. Inserting this parametric solution into the last equation, we get an equation in a single variable

$$x =_{\eta_m} f_m(\boldsymbol{u}_{1,m-1}(x), x)$$

that can be solved by taking for the function $\lambda x.\, f_m(\boldsymbol{u}_{1,m-1}(x), x)$, the least or greatest fixpoint, depending on whether the last equation is a μ- or ν-equation. This provides the m-th component of the solution $u_m = \eta_m(\lambda x.\, f_m(\boldsymbol{u}_{1,m-1}(x), x))$. The remaining components of the solution are obtained inserting u_m in the parametric solution $\boldsymbol{u}_{1,m-1}(x)$ previously computed, i.e., $\boldsymbol{u}_{1,m-1} = \boldsymbol{u}_{1,m-1}(u_m)$.

The next lemma will be helpful in several places. In particular, it shows that the definition above is well-given, since we are taking (least or greatest) fixpoints of monotone functions.

Lemma 7.3.7: Solution is Monotone

Let E be a system of $m > 0$ equations of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$ over a lattice L. For $i \in \underline{m}$ the function $g\colon L \to L^{m-1}$ defined by $g(x) = sol(E[x_i := x])$ is monotone.

Proof: The proof proceeds by induction on m. The base case $m = 1$ holds trivially since necessarily $i = 1$ and for any $x \in L$, the system $E[x_i := x]$ is empty, with empty solution.

Let us assume $m > 1$. We distinguish two subcases according to whether $i = m$ or $i < m$. If $i = m$ then by definition of solution

$$g(x) = sol(E[x_m := x]) = (sol(E[x_m := x][x_{m-1} := u_{m-1}(x)]), u_{m-1}(x)) \quad (7.1)$$

where $u_{m-1}(x) = \eta_{m-1}(\lambda y.\, f_{m-1}(sol(E[x_m := x][x_{m-1} := y]), y, x)$.

Next observe that the function $h\colon L^2 \to L^{m-2}$ defined by

$$h(x, y) = sol(E[x_m := x][x_{m-1} := y])$$

is monotone. In fact, it is monotone in y by inductive hypothesis, and also in x, again by inductive hypothesis, since $E[x_m := x][x_{m-1} := y] = E[x_{m-1} := y][x_m := x]$. Observe that u_{m-1} can be written as

$$u_{m-1}(x) = \eta_{m-1}(\lambda y.\, f_{m-1}(h(x, y), y, x))$$

Recalling that also f_{m-1} is monotone, we deduce that u_{m-1} is monotone.

Finally, using the definition of g and u_{m-1}, from (7.1) we can derive

$$g(x) = (h(x, u_{m-1}(x)), u_{m-1}(x))$$

which allows us to conclude that g is monotone.

If instead, $i < m$, just note that

$$g(x) = sol(E[x_i := x]) = (sol(E[x_i := x][x_m := u_m(x)]), u_m(x)) \quad (7.2)$$

where $u_m(x) = \eta_m(\lambda y. f_m\,(sol(E[x_i := x][x_m := y]), y, x)$. Then the proof proceeds as in the previous case. □

It can be easily proved that the solution of a system is, as anticipated, a special pre-solution.

Lemma 7.3.8: Solution is Pre-Solution

Let E be a system of m equations over a lattice L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$ and let $\boldsymbol{u}$ be its solution. Then $\boldsymbol{u}$ is a pre-solution, i.e., $\boldsymbol{u} = \boldsymbol{f}(\boldsymbol{u})$.

Proof: The proof proceeds by induction on m. The base case $m = 0$ trivially holds. For any $m > 0$, let $\boldsymbol{u} = \boldsymbol{u'}u_m$, $\boldsymbol{f} = \boldsymbol{f'}f_m$ and $\boldsymbol{x} = \boldsymbol{x'}x_m$. Since $\boldsymbol{u'} = sol(E[x_m := u_m])$, by inductive hypothesis, we have that

$$\boldsymbol{u'} = \boldsymbol{f'}[x_m := u_m](\boldsymbol{u'}) = \boldsymbol{f'}(\boldsymbol{u}). \quad (7.3)$$

Moreover, again by definition of solution, we have that

$$u_m = \eta_m(\lambda x.\, f_m(sol(E[x_m := x]), x))$$

. Hence $u_m = f_m(sol(E[x_m := u_m]), u_m))$. Recalling that $sol(E[x_m := u_m]) = \boldsymbol{u'}$ we deduce $u_m = f_m(\boldsymbol{u'}, u_m) = f_m(\boldsymbol{u})$, that together with (7.3) gives $\boldsymbol{u} = \boldsymbol{f}(\boldsymbol{u})$ as desired. □

The game theoretical characterisation of the solution of a system of fixpoint equations discussed later will rely on a notion of approximation of the solution that is reminiscent of the lattice progress measure in [HSC16].

Definition 7.3.9: Approximants

Let E be a system of m equations over the lattice L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$. Given any tuple $\boldsymbol{l} \in L^m$, let $f_{i,\boldsymbol{l}} \colon L \to L$ be the function defined by

$$f_{i,\boldsymbol{l}}(x) = f_i(sol(E[\boldsymbol{x}_{i+1,m} := \boldsymbol{l}_{i+1,m}][x_i := x]), x, \boldsymbol{l}_{i+1,m}).$$

We say that a tuple $\boldsymbol{l} \in L^m$ is a *μ-approximant* when for all $i \in \underline{m}$, if $\eta_i = \nu$ then $l_i = \nu(f_{i,\boldsymbol{l}})$, else if $\eta_i = \mu$ then $l_i = f^{\alpha}_{i,\boldsymbol{l}}(\bot)$ for some ordinal α. Dually, $\boldsymbol{l} \in L^m$ is a *ν-approximant* when for all $i \in \underline{m}$, if $\eta_i = \nu$ then $l_i = f^{\alpha}_{i,\boldsymbol{l}}(\top)$ for some ordinal α, else if $\eta_i = \mu$ then $l_i = \mu(f_{i,\boldsymbol{l}})$.
Whenever $\boldsymbol{l}$ is a μ-approximant we write $ord(\boldsymbol{l})$ to denote the least m-tuple of ordinals $\boldsymbol{\alpha}$ such that for any $i \in \underline{m}$, if $\eta_i = \mu$ then $l_i = f^{\alpha_i}_{i,\boldsymbol{l}}(\bot)$ else, if $\eta_i = \nu$, $l_i = f^{\alpha_i}_{i,\boldsymbol{l}}(\top) = \nu(f_{i,\boldsymbol{l}})$.

Observe that, spelling out the definition of the solution of a system of equations, it can be easily seen that $sol_i(E[\boldsymbol{x}_{i+1,m} := \boldsymbol{l}_{i+1,m}]) = \eta_i(f_{i,\boldsymbol{l}})$. Then a μ-approximant is obtained by taking under-approximations for the least fixpoints and the exact value for greatest fixpoints. In fact, in the case of μ-approximants, for each $i \in \underline{m}$, if $\eta_i = \nu$, the i-th component is set to $\nu(f_{i,\boldsymbol{l}})$ which is i-th component $sol_i(E[\boldsymbol{x}_{i+1,m} := \boldsymbol{l}_{i+1,m}])$ of the solution. Instead, if $\eta_i = \mu$ the component l_i is set to $f^{\alpha}_{i,\boldsymbol{l}}(\bot)$ for some ordinal α, which is an underapproximation of $\mu(f_{i,\boldsymbol{l}}) = sol_i(E[\boldsymbol{x}_{i+1,m} := \boldsymbol{l}_{i+1,m}])$, obtained by iterating $f_{i,\boldsymbol{l}}$ over $\bot$ up to ordinal α. For ν-approximants the situation is dual.

We remark that the function $f_{i,\boldsymbol{l}}$ depends only on the subvector $\boldsymbol{l}_{i+1,m}$. In particular $f_{m,\boldsymbol{l}}$ does not depend on $\boldsymbol{l}$. In fact, $f_{m,\boldsymbol{l}} = \lambda x.\, f_m(sol(E[x_m := x]), x)$. Using $\boldsymbol{l}$ as subscript instead of the subvector is a slight abuse of notation that makes the notation lighter.

Approximants can be given an inductive characterisation. Besides shedding some light on the notion of approximant, the following easy result will be useful at a technical level.

Lemma 7.3.10: Inductive characterisation of Approximants

Let E be a system of $m > 0$ equations over the lattice L, of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$ and let $g_m \colon L \to L$ be the function $g_m(x) = f_m(sol(E[x_m := x]), x)$. A tuple $\boldsymbol{l} \in L^m$ is a μ-approximant iff the following conditions hold

1. either $\eta_m = \mu$ and $l_m = g_m^\alpha(\bot)$ for some ordinal α, or $\eta_m = \nu$ and $l_m = \nu g_m$
2. $\boldsymbol{l}_{1,m-1}$ is a μ-approximant of $E[x_m := l_m]$.

Proof: Immediate. □

As mentioned above, μ-approximants are closely related to lattice progress measures in the sense of [HSC16]. In fact, from Lemma 7.3.10 we can infer that, given a vector $\boldsymbol{\alpha}$ of ordinals, the μ- or ν-approximant $\boldsymbol{l} \in L^m$ with $ord(\boldsymbol{l}) = \boldsymbol{\alpha}$ is uniquely determined. More precisely, a μ-approximant $\boldsymbol{l}$ is determined by the subvector of $ord(\boldsymbol{l})$ consisting only of the m-tuple of components of $ord(\boldsymbol{l})$ corresponding to μ-indices. Hence we can define a function that maps each such m-tuple of ordinals to the corresponding μ-approximant and this turns out to be a lattice progress measures in the sense of [HSC16]. Actually, as proved in [BK+18, Appendix B], it is the greatest one. It can be seen to coincide with the measure used in [HSC16, Theorem 2.13] (completeness part).

We next observe that the name approximant is appropriate, i.e., μ-approximants provide an approximation of the solution from below, while ν-approximants from above. The solution is thus the only pre-solution which is both a μ- and a ν-approximant.

Lemma 7.3.11: Solution and Approximants

Let E be a system of m equations over the lattice L, of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$. The solution of E is the greatest μ-approximant and the least ν-approximant.

Proof: The solution $\boldsymbol{u}$ is clearly a μ-approximant. We prove that it is the greatest μ-approximant by induction on m. If $m = 0$ the thesis is vacuously true. If $m > 0$, consider another μ-approximant $\boldsymbol{l}$. We distinguish two subcases according to whether $\eta_m = \mu$ or $\eta_m = \nu$. If $\eta_m = \mu$, we know that $l_m = f_{m,\boldsymbol{l}}^\alpha(\bot)$ for some ordinal α. Observe that $f_{m,\boldsymbol{l}} = \lambda x.\, f_m(sol(E[x_m := x]), x))$ is the function for which u_m is the least fixpoint, hence

$$l_m \sqsubseteq u_m. \tag{7.4}$$

Moreover, by Lemma 7.3.10, $\boldsymbol{l}_{1,m-1}$ is a μ-approximant for the system $E[x_m := l_m]$.

Hence, by inductive hypothesis

$$\boldsymbol{l}_{1,m-1} \sqsubseteq sol(E[x_m := l_m]) \tag{7.5}$$

Moreover, by monotonicity of the solution (Lemma 7.3.7), since $l_m \sqsubseteq u_m$, we get $sol(E[x_m := l_m]) \sqsubseteq sol(E[x_m := u_m]) = \boldsymbol{u}_{1,m-1}$. Therefore, combined with (7.4) and (7.5), we conclude $\boldsymbol{l} \sqsubseteq \boldsymbol{u}$.

The proof for ν-approximants is dual. □

We conclude with a technical lemma that will be used to locally modify approximations in the game.

Lemma 7.3.12: Updating Approximants

Let E be a system of m equations over the lattice L, of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$ and let $\boldsymbol{l}$ be a μ-approximant with $ord(\boldsymbol{l}) = \boldsymbol{\alpha}$. For any $i \in \underline{m}$ and ordinal $\alpha \leq \alpha_i$

1. if $\eta_i = \mu$, then $\boldsymbol{l'} = (sol(E[\boldsymbol{x}_{i+1,m} := \boldsymbol{l}_{i+1,m}][x_i := l'_i]), l'_i, \boldsymbol{l}_{i+1,m})$, with $l'_i = f^{\alpha}_{i,\boldsymbol{l}}(\bot)$ for some ordinal α, is a μ-approximant
2. if $\eta_i = \nu$, then $\boldsymbol{l'} = (sol(E[\boldsymbol{x}_{i+1,m} := \boldsymbol{l}_{i+1,m}]), \boldsymbol{l}_{i+1,m})$ is a μ-approximant

and in both cases $ord(\boldsymbol{l'}) \preceq_i ord(\boldsymbol{l})$. A dual result holds for ν-approximants.

Proof: Let us focus on (1). In order to show that

$$\boldsymbol{l'} = (sol(E[\boldsymbol{x}_{i+1,m} := \boldsymbol{l}_{i+1,m}][x_i := l'_i]), l'_i, \boldsymbol{l}_{i+1,m})$$

is a μ-approximant, first observe that the components $l_{i+1}, \ldots, l_m$ do not change. Component l'_i is of the desired shape by definition. Finally, for $j < i$ the component l'_j is defined as $sol_j(E[\boldsymbol{x}_{i+1,m} := \boldsymbol{l}_{i+1,m}][x_i := l'_i])$ and thus, by definition of solution of a system, if $\eta_j = \nu$ then $l'_j = \nu(f_{j,\boldsymbol{l'}})$ and if $\eta_j = \mu$ then $l'_j = \mu(f_{j,\boldsymbol{l'}}) = f^{\beta}_{j,\boldsymbol{l'}}(\bot)$ for some ordinal β, as desired. Finally observe that since $\boldsymbol{l}$ and $\boldsymbol{l'}$ coincide on components $i+1, \ldots, m$, and $l_i = f^{\alpha_i}_{i,\boldsymbol{l}}(\bot)$, while $l'_i = f^{\alpha}_{i,\boldsymbol{l}}(\bot)$, with $\alpha \leq \alpha_i$, clearly $ord(\boldsymbol{l'}) \preceq_i ord(\boldsymbol{l})$.

The proof for (2) is analogous. In fact, also in this case the components $i+1 \ldots, m$ are unchanged and finally, for $j \leq i$ the component l'_j is defined as

$$sol_j(E[\boldsymbol{x}_{i+1,m} := \boldsymbol{l}_{i+1,m}][x_i := l'_i]),$$

thus the same reasoning as above applies.

Both can be easily dualized for ν-approximants. □

7.4 Application Scenarios

We will consider two different verification questions and demonstrate how to reformulate both analysis frameworks via a system of fixpoint equations as described in Definition 7.3.1.

The content of the following two sections is taken from [NNH99; BW18; BK08] where the beginnings of model checking go back further (cf. [QS82; CE81; GV08]).

7.4.1 Modal μ-Calculus

The system behaviour should match the specification and bisimulation (see Section 2.2) is one way to verify if an implementation or design satisfies the underlying requirements. In case the specification is quite complex, *model checking* provides a suitable alternative technique where several *properties* are considered instead of one large specification model [FV99]. This concept can be described as follows (see [BK08; FV99]):

1. Provide a *model* of the system design or implementation which is usually given by a state-based model (see Section 2.2).

2. Formalize the required properties in terms of the underlying *specification language*.

3. Verify if the model satisfies each of the properties.

Since state-based models have already been discussed in Section 2.2 we move to the second point and focus on an expressive logic. Therefore, we consider the standard μ-calculus which allows to encode also formulas from *linear temporal logic (LTL)* and *computational tree logic (CTL)*. LTL and CTL are basic languages for verifying *linear temporal* and *branching* properties (for the interested reader we refer to [BK08]).

The *syntax* of the standard μ-calculus extends the set of logical operators given by Hennessy-Milner introduced in Section 3.2.1. For fixed disjoint sets $PVar$ of propositional variables, ranged over by $x, y, z, \ldots$ and $Prop$ of propositional symbols, ranged over by $p, q, r, \ldots$, formulas are defined by

$$\varphi \ ::= \ \mathbf{t} \ \mid \ \mathbf{f} \ \mid \ p \mid \ x \ \mid \ \varphi_1 \wedge \varphi_2 \ \mid \ \varphi_1 \vee \varphi_2 \ \mid \ \Box\varphi \ \mid \ \Diamond\varphi \ \mid \ \eta x.\, \varphi$$

where $p \in Prop$, $x \in PVar$ and $\eta \in \{\mu, \nu\}$. Formulas of the kind $\eta x.\, \varphi$ are called fixpoint formulas.

We explain the *semantics* of such a formula with respect to an unlabelled transition system (or Kripke structure) $(S, \to)$ where S is the set of states and $\to \subseteq S \times S$ is the transition relation. For the sake of completeness we recall the semantics of the standard and modal operators by Hennessy-Milner presented in Section 3.2.1. Given a formula φ and an environment $\rho\colon \mathit{Prop} \cup \mathit{PVar} \to \mathcal{P}S$ mapping each proposition or propositional variable to the set of states where it holds, we denote by $[\![\varphi]\!]_\rho$ the semantics of φ which are defined as follows (see, e.g., [BW18]):

$$[\![\mathbf{t}]\!]_\rho = S \qquad [\![\Box\varphi]\!]_\rho = \{s \in S \mid \forall_{t\in S}\ s \to t \Rightarrow t \in [\![\varphi]\!]_\rho\}$$
$$[\![\mathbf{f}]\!]_\rho = \emptyset \qquad [\![\Diamond\varphi]\!]_\rho = \{s \in S \mid \exists_{t\in S}\ s \to t \text{ and } t \in [\![\varphi]\!]_\rho\}$$
$$[\![x]\!]_\rho = \rho(x) \qquad [\![\neg\varphi]\!]_\rho = S \setminus [\![\varphi]\!]_\rho$$

We consider the fixpoint operators μ and ν separately and by the monotonicity of all logical operators above we derive the following semantics [BW18]:

$$[\![\mu x.\varphi]\!]_\rho = \bigcap\{S' \subseteq S \mid [\![\varphi]\!]_{\rho[x:=S']} \subseteq S'\}$$

$$[\![\nu x.\varphi]\!]_\rho = \bigcup\{S' \subseteq S \mid S' \subseteq [\![\varphi]\!]_{\rho[x:=S']}\}$$

The definition above is based on Knaster-Tarski (see Section 2.1) and since the lattice $(\mathcal{P}S, \subseteq)$ satisfies the requirements of Kleene's Theorem (see Theorem 2.1.3) we can compute $\mu x.\varphi$ and $\nu x.\varphi$ respectively based on $\bot$ and $\top$.

We will explain the semantics using a small Example 7.4.1. A detailed introduction to modal μ-calculus can be found in [BW18].

Example 7.4.1

In Figure 7.2 a simple Kripke structure is given where only state b satisfies the proposition p. Consider the formula $\varphi = \nu x_2.((\mu x_1.(p \vee \Diamond x_1)) \wedge \Box x_2)$ requiring that from all reachable states there exists a path that eventually reaches a state where p holds.

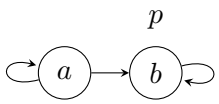


Figure 7.2: A simple Kripke structure with one proposition p.

We apply the fixpoint iteration technique and therefore we interpret x_2 via $\top = \{a, b\}$.

Next, we need to compute the least fixpoint with respect to the inner formula

$(\mu x_1.(p \vee \Diamond x_1)) \wedge \Box x_2$. Note, that x_2 does not occur within $\varphi' = \mu x_1.(p \vee \Diamond x_1)$ and we start the fixpoint iteration with $x_1 = \bot = \emptyset$:

$$\begin{array}{lllll} x_1 = \emptyset: & \{b\} \cup \Diamond\emptyset & = \{b\} & \Longrightarrow [\![\varphi']\!]_{\rho[x_1:=\emptyset]} & \neq \emptyset \\ x_1 = \{b\}: & \{b\} \cup \Diamond\{b\} & = \{a,b\} & \Longrightarrow [\![\varphi']\!]_{\rho[x_1:=\{b\}]} & \neq \{b\} \\ x_1 = \{a,b\}: & \{b\} \cup \Diamond\{a,b\} & = \{a,b\} & \Longrightarrow [\![\varphi']\!]_{\rho[x_1:=\{a,b\}]} & = \{a,b\} \end{array}$$

We proceed with the outer fixpoint operator ν and as computed above $x_1 = \{a,b\}$ and we start with $x_2 = \top = \{a,b\}$:

$$x_2 = \{a,b\}: \quad \{a,b\} \cup \Box\{a,b\} \quad = \{a,b\} \quad \Longrightarrow [\![\varphi]\!]_{\rho[x_2:=\{a,b\}]} \quad = \{a,b\}$$

Therefore we obtain $x_1 = x_2 = \{a,b\}$ and $[\![\varphi]\!] = \{a,b\}$. This solution corresponds to the fact that the formula φ holds in every state due to the system in Figure 7.2.

Next, note that any μ-calculus formula can be expressed in equational form, by inserting an equation for each propositional variable (see also [CKS92; Sei96]). The reverse translation is also possible, hence these specification languages are equally expressive [HSC16]. Here, we will only depict the relation via an example, the formal treatment is given in [BK+18; CKS92].

Example 7.4.2

We consider the system in Figure 7.2 and translate the formula $\varphi = \nu x_2.((\mu x_1.(p \vee \Diamond x_1)) \wedge \Box x_2)$ into an equational system.
The fixpoint operator of the inner formula $(\mu x_1.(p \vee \Diamond x_1)) \wedge \Box x_2$ binding the variable x_1 yields the first equation where the outer fixpoint operator binding the variable x_2 results in a second equation:

$$\begin{array}{llll} x_1 & =_\mu & p \vee \Diamond x_1 & (x_1 =_\mu \{b\} \cup \Diamond x_1) \\ x_2 & =_\nu & x_1 \wedge \Box x_2 & (x_2 =_\nu x_1 \cap \Box x_2) \end{array}$$

Figure 7.3: Equational form of $\varphi = \nu x_2.((\mu x_1.(p \vee \Diamond x_1)) \wedge \Box x_2)$.

The solution $x_1 = x_2 = \{a,b\}$ is computed inductively via fixpoints and therefore we obtain the same computations as in Example 7.4.1.

Example 7.4.3

Consider the formula $\varphi' = \nu x_2.(\Box x_2 \wedge \mu x_1.((p \wedge \Diamond x_2) \vee \Diamond x_1))$ requiring that from all reachable states there is a path along which p holds infinitely often. The equational form of φ' is:

$$\begin{aligned} x_1 &=_\mu (p \wedge \Diamond x_2) \vee \Diamond x_1 \\ x_2 &=_\nu \Box x_2 \wedge x_1 \end{aligned}$$

On the same transition system of the previous example (see Figure 7.2), the solution of the corresponding system is $x_1 = x_2 = S$. Notice that this time the order of the equations is relevant, while in the previous example it was not. Indeed, if we swap the two equations in the system, the solution becomes $x_1 = x_2 = \emptyset$. In general, the order of the equations is important whenever there is alternation of fixpoints (mutual dependencies between least and greatest fixpoint equations).

7.4.2 Data Flow Analysis

Data-flow analysis of programs is another area where fixpoints play a major role. One can easily state a program analysis question in this setting as a system of fixpoint equations, based on the flow graph of the program under consideration.

There are several data-flow analyses as *liveness* or *available expressions*, but we focus on the well-known *constant propagation analysis* (see, e.g., [NNH99]). Its aim is to show that the value of a variable is always constant at a certain program point, allowing us to optimize the program by replacing the variable by the constant. Consider for instance the while program in Figure 7.4, where variables contain integer values and blocks are numbered in order to easily reference them.

```
[y:=6]^1;
[x:=y+1]^2;
while [*]^3 do
      [y:=x+y]^4
od
```

Figure 7.4: A simple while program.

The condition for the while loop (block 3) is irrelevant and is hence replaced by $*$. Note that variable `x` always has value 7 in block 4 and hence the assignment in this

block could be replaced by `y:=7+y`.

Following [NNH99] we analyze such programs by setting up an instance of a monotone framework. In particular we will use the following lattice to record the results of the analysis:

$$L = (\mathbb{Z} \cup \{\bot\})^{Var} \cup \{\top\}$$

where Var is the set of variables. That is, a lattice element is either $\top$ or a function $\rho\colon Var \to \mathbb{Z} \cup \{\bot\}$ that assigns a variable $\mathtt{x}$ to a value in $\mathbb{Z}$ (if $\mathtt{x}$ is known to have constant value $\rho(\mathtt{x})$ at this program point) or to $\bot$ (to indicate that $\mathtt{x}$ is possibly non-constant). As usual, we are allowed to over-approximate and $\bot$ might be assigned although the value of the variable is actually constant.

The lattice order is defined as follows: two assignments $\rho_1, \rho_2\colon Var \to \mathbb{Z} \cup \{\bot\}$ are ordered, i.e. $\rho_1 \sqsubseteq \rho_2$, if for each $\mathtt{x} \in Var$ either $\rho_1(\mathtt{x}) = \rho_2(\mathtt{x})$ or $\rho_1(\mathtt{x}) = \bot$. That is, we consider a flat order where $\bot$ is smaller than the integers and the integers themselves are incomparable, and extend it pointwise to functions. Clearly, $\top$ is the largest lattice element and we use some overloading and denote by $\bot$ the function that maps every variable to $\bot$. Note that this order deviates from the usual convention in program analysis which states that smaller values should be more precise than larger values. We do this since our game characterizes whether a lattice element is *below* the solution. Since we want to check that the solution is more precise than a given threshold, we have to reverse the order.

Let us write $\rho' = \rho[\mathtt{x} \mapsto z]$ for $z \in \mathbb{Z}$ to denote function update, that is $\rho'(\mathtt{x}) = z$ and $\rho'(\mathtt{y}) = \rho(\mathtt{y})$ for $\mathtt{y} \neq \mathtt{x}$. When $\rho = \top$ we define $\rho[\mathtt{x} \mapsto z] = \top$.

Observe that L is algebraic (and hence continuous). The compact elements are $\top$ and those functions which have finite support, i.e., functions of the kind $\bot[\mathtt{x}_1 \mapsto z_1, \ldots, \mathtt{x}_n \mapsto z_n]$ where only finitely many variables are not mapped to $\top$. In particular we can use as a basis the functions $\bot[\mathtt{x} \mapsto z]$ for some $\mathtt{x} \in Var$ and $z \in \mathbb{Z}$. Note also that L is not distributive. For instance if $\rho_i = \bot[\mathtt{x} \mapsto i]$ for $i \in \{1, 2, 3\}$ then $(\rho_1 \sqcap \rho_2) \sqcup \rho_3 = \bot \sqcup \rho_3 = \rho_3$ while $(\rho_1 \sqcup \rho_3) \sqcap (\rho_2 \sqcup \rho_3) = \top \sqcap \top = \top$.

Example 7.4.4

From the program in Figure 7.4 we can easily derive the system of fixpoint equations in Figure 7.5, where we use ρ_i to record the lattice value for the entry of block i.

$$\begin{aligned}
\rho_1 &=_\nu \bot \\
\rho_2 &=_\nu \rho_1[\mathtt{y} \mapsto 6] \\
\rho_3 &=_\nu \rho_2[\mathtt{x} \mapsto \rho_2(\mathtt{y}) + 1] \sqcap \rho_4[\mathtt{y} \mapsto \rho_4(\mathtt{x}) + \rho_4(\mathtt{y})] \\
\rho_4 &=_\nu \rho_3
\end{aligned}$$

Figure 7.5: The equation system for the corresponding constant propagation analysis with respect to the program in Figure 7.4.

At the beginning, no variable is constant. Then the equation system mimics the control flow of the program. In block 3 we have to take the meet to obtain an analysis result that is less precise than the results coming from block 2 respectively block 4. Furthermore, since precision increases with the order, we are interested in the greatest solution, which means that we have only ν-equations.
The expected solution is $\rho_1 = \bot$, $\rho_2 = \bot[\mathtt{y} \mapsto 6]$, $\rho_3 = \rho_4 = \bot[\mathtt{x} \mapsto 7]$ witnessing that at block 2 variable `y` has constant value 6 and at blocks 3 and 4 variable `x` has constant value 7.

7.5 Fixpoint Games over Continuous Lattices

In this section we present a game-theoretical approach to the solution of a system of fixpoint equations over a continuous lattice. More precisely, given a lattice with a fixed basis, the game allows us to check whether an element of the basis is smaller (with respect to $\sqsubseteq$) than the solution of a selected equation. This corresponds to solving the associated verification problem. For instance, when model checking the μ-calculus, one is interested in establishing whether a system satisfies a formula φ, which amounts to check whether $\{s_0\} \subseteq u_\varphi$ where s_0 is the initial state and u_φ is the last component of the solution of the system of equations associated with φ.

7.5.1 Definition of the Game

The fixpoint game that we introduce has been inspired by the unfolding game described in [Ven08], that works for a single fixpoint equation over the powerset lattice. We adopted the name *fixpoint game*, analogously to [HK+17].

Definition 7.5.1: Fixpoint Game

Let L be a continuous lattice and let B_L be a basis of L such that $\bot \notin B_L$. Given a system E of m equations over L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$, the corresponding fixpoint game is a parity game, with an existential player $\exists$ and a universal player $\forall$, defined as follows:

- The positions of $\exists$ are pairs (b, i) where $b \in B_L$ and $i \in \underline{m}$ and those of $\forall$ are tuples $\boldsymbol{l} \in L^m$.
- From (b, i) the possible moves of $\exists$ are $\mathbf{E}(b, i) = \{\boldsymbol{l} \mid \boldsymbol{l} \in L^m \ \wedge \ b \sqsubseteq f_i(\boldsymbol{l})\}$.
- From $\boldsymbol{l} \in L^m$ the possible moves of $\forall$ are $\mathbf{A}(\boldsymbol{l}) = \{(b, i) \mid i \in \underline{m} \wedge b \in B_L \wedge b \ll l_i\}$.

The game is schematized in Table 7.1. For a finite play, the winner is the player whose opponent is unable to move. For an infinite play, let h be the highest index that occurs infinitely often in a pair (b, i):

$$\text{If } \eta_h = \nu \text{ then } \exists \text{ wins, else } \forall \text{ wins.}$$

Observe that the fixpoint game is a parity game [EJ91; Zie98] (on an infinite graph) and the winning condition is the natural formulation of the standard winning condition in this setting.

Position	Player	Moves
(b, i)	$\exists$	$(l_1, \ldots, l_m)$ such that $b \sqsubseteq f_i(l_1, \ldots, l_m)$
$(l_1, \ldots, l_m)$	$\forall$	(b', j) such that $b' \ll l_j$

Table 7.1: The fixpoint game

Hereafter, whenever we consider a continuous lattice L, we assume that a basis B_L is fixed such that $\bot \notin B_L$. Elements of the basis will be denoted by letters b with super or subscripts.

We will prove correctness and completeness of the game, i.e., we will show that if $\boldsymbol{u}$ is the solution of the system, given a basis element $b \in B_L$ and $i \in \underline{m}$, if $b \sqsubseteq u_i$ then starting from (b, i) the existential player has a winning strategy, otherwise the universal player has a winning strategy.

Example 7.5.2

As an example, consider the μ-calculus formula

$$\varphi = \nu x_2.((\mu x_1.(p \vee \Diamond x_1)) \wedge \Box x_2)$$

and the corresponding equation system of Example 7.4.2:

$$\begin{aligned} x_1 &=_\mu \{b\} \cup \Diamond x_1 \\ x_2 &=_\nu x_1 \cap \Box x_2 \end{aligned}$$

Recall that the lattice is $(\mathcal{P}S, \subseteq)$ and let us fix as a basis the set of singletons $B_{\mathcal{P}S} = \{\{a\}, \{b\}\}$.

A portion of the fixpoint game is graphically represented as a parity game in Figure 7.6. Diamond nodes correspond to positions of player $\exists$ and the box nodes to positions of player $\forall$. Only a subset of the possible positions for $\forall$ are represented. The positions which are missing, such as $(\{a, b\}, \{a, b\})$, can be shown to be redundant, in a sense formalized in [BK+18]), so that the subgame is equivalent to the full game. Numbers in the diamond nodes correspond to priorities. Box nodes do not have priorities (or we can assume priority 0). Since index 1 and 2 corresponds to a μ and a ν equation, respectively, in this specific case the winning condition for player $\exists$ is exactly the same as for parity games: either the play is finite and $\exists$ plays last or the play is infinite and the highest priority that occurs infinitely often is even (in this case 2).

Let (u_1, u_2) be the solution of the system. We can check if $a \in u_2$, i.e., if a satisfies φ, by playing the game from the position $(\{a\}, 2)$. In fact, $\{a\} \sqsubseteq u_2$ amounts to $a \in u_2$.

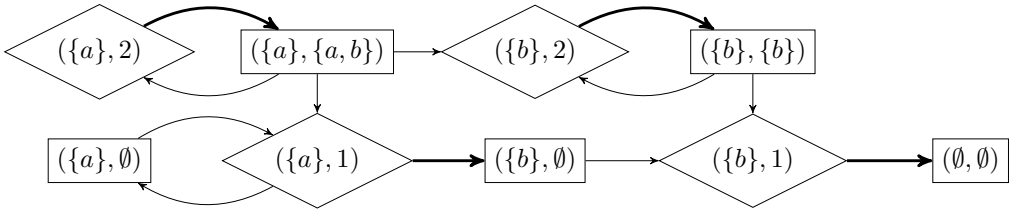


Figure 7.6: Graphical representation of a fixpoint game

Indeed player $\exists$ has a winning strategy that we can represent as a function ς from the positions of the game (for any play) to the corresponding moves of player $\exists$, i.e., $\varsigma\colon B_{\mathcal{P}S} \times \underline{2} \to \mathcal{P}S \times \mathcal{P}S$. A winning strategy for $\exists$ is given by $\varsigma(\{a\}, 1) = (\{b\}, \emptyset)$, $\varsigma(\{a\}, 2) = (\{a\}, \{a, b\})$, $\varsigma(\{b\}, 1) = (\emptyset, \emptyset)$ and $\varsigma(\{b\}, 2) = (\{b\}, \{b\})$. In Fig. 7.6 we depict by bold arrows the choices prescribed by the strategy.

A possible play of the game could be the following, where $\overset{x}{\rightsquigarrow}$ denotes a move of $x \in \{\exists, \forall\}$:

$$(\{a\}, 2) \overset{\exists}{\rightsquigarrow} (\{a\}, \{a, b\}) \overset{\forall}{\rightsquigarrow} (\{a\}, 1) \overset{\exists}{\rightsquigarrow} (\{b\}, \emptyset) \overset{\forall}{\rightsquigarrow} (\{b\}, 1) \overset{\exists}{\rightsquigarrow} (\emptyset, \emptyset) \overset{\forall}{\not\rightsquigarrow},$$

hence $\exists$ wins. Another (infinite) play is the following. It is also won by $\exists$ since the highest index that occurs infinitely often is 2, which is a ν-index:

$$(\{a\}, 2) \overset{\exists}{\rightsquigarrow} (\{a\}, \{a, b\}) \overset{\forall}{\rightsquigarrow} (\{a\}, 2) \overset{\exists}{\rightsquigarrow} (\{a\}, \{a, b\}) \overset{\forall}{\rightsquigarrow} \dots$$

Note that if $\exists$ always plays as specified by ς, she will always win.

7.5.2 Correctness and Completeness

Before proving correctness and completeness of the game in the general case, as a warm up, we give some intuition and outline the proof for the case of a single equation. Let $f \colon L \to L$ be a monotone function on a continuous lattice L and consider the equation $x =_\eta f(x)$, where $\eta \in \{\nu, \mu\}$, with solution $u = \eta f$. In this case the positions for $\exists$ are simply basis elements $b \in B_L$ and $\exists$ must choose $l \in L$ such that $b \sqsubseteq f(l)$. Positions of $\forall$ are lattice elements $l \in L$ and moves are elements of the basis $b \in B_L$, with $b \ll l$. In the case of $\eta = \mu$, player $\forall$ wins infinite plays and in the case of $\eta = \nu$, player $\exists$ wins infinite plays.

When $\eta = \mu$, if $b \sqsubseteq u$, then $b \sqsubseteq f^\alpha(\bot)$ for some ordinal α. The idea is that $\exists$ can win by descending the chain $f^\beta(\bot)$. E.g., if $\beta = \gamma + 1$ is a successor ordinal, then $\exists$ can play $f^\gamma(\bot)$. If instead, $\eta = \nu$, then the existential player can win just by identifying some post-fixpoint l such that $b \sqsubseteq l$. In fact, if l is a post-fixpoint, i.e., $l \sqsubseteq f(l)$ we know that $l \sqsubseteq u$. Moreover, if $b \sqsubseteq l$ then $b \sqsubseteq f(l)$ and thus $\exists$ can cycle on l and win. More formally:

(Case $\eta = \mu$)

In this case $u = f^\alpha(\bot)$ for some ordinal α.

- *Completeness:* We show that whenever $b \sqsubseteq f^\beta(\bot)$, for some ordinal β (i.e., b is below some μ-approximant), then $\exists$ has a winning strategy, by transfinite induction on β. First observe that $\beta > 0$. In fact, otherwise $b \sqsubseteq f^0(\bot) = \bot$, hence $b = \bot$, while $\bot \notin B_L$ by hypothesis. Hence we have two possibilities:
  - If β is a limit ordinal, player $\exists$ plays $l = f^\beta(\bot)$, which is a post-fixpoint and hence $b \sqsubseteq f^\beta(\bot) \sqsubseteq f(f^\beta(\bot))$. Then $\forall$ chooses $b' \ll f^\beta(\bot) = \bigsqcup_{\gamma<\beta} f^\gamma(\bot)$.

Since this is a directed join, by definition of the way-below relation there exists $\gamma < \beta$ with $b' \sqsubseteq f^\gamma(\bot)$.

– If $\beta = \gamma + 1$, $\exists$ plays $l = f^\gamma(\bot)$ and $\forall$ chooses $b' \ll f^\gamma(\bot)$, hence $b' \sqsubseteq f^\gamma(\bot)$.

Note that $\exists$ always has a move and the answer of $\forall$ is some $b' \sqsubseteq f^\gamma(\bot)$, with $\gamma < \beta$, from which there exists a winning strategy for $\exists$ by the inductive hypothesis.

- *Correctness:* We show that whenever $b \not\sqsubseteq u$, player $\forall$ has a winning strategy.

  Observe that a move of $\exists$ will be some l such that $b \sqsubseteq f(l)$. Note that there must be a $b' \ll l$ with $b' \not\sqsubseteq u$. In fact, otherwise, if for all $b' \ll l$ it holds that $b' \sqsubseteq u$, since L is a continuous lattice, we would have $l = \bigsqcup\{b' \mid b' \ll l\} \sqsubseteq u$ and furthermore $b \sqsubseteq f(l) \sqsubseteq f(u) = u$, which is a contradiction.

  Hence $\forall$ can choose such a $b' \ll l$ with $b' \not\sqsubseteq u$ and the game can continue. Then either $\exists$ runs out of moves at some point or we end up in an infinite play. In both cases $\forall$ wins.

(Case $\eta = \nu$)

In this case $u = f^\alpha(\top)$ for some ordinal α.

- *Completeness:* We show that when $b \sqsubseteq u$, then $\exists$ has a winning strategy. In fact, in this case $\exists$ simply plays $l = u$, which satisfies $b \sqsubseteq u = f(u)$ and $\forall$ answers with some $b \ll u$, hence $b \sqsubseteq u$. The game can thus continue forever, leading to an infinite play which is won by $\exists$.

- *Correctness:* We show that whenever $b \not\sqsubseteq f^\beta(\top)$, for some ordinal β (i.e., b is not below some ν-approximant), then $\forall$ has a winning strategy, by transfinite induction on β. First observe that $\beta > 0$. In fact, otherwise $b \not\sqsubseteq f^0(\top) = \top$ would be a contradiction. Hence we distinguish two cases:

  – If β is a limit ordinal $b \not\sqsubseteq f^\beta(\top) = \bigsqcap_{\gamma<\beta} f^\gamma(\top)$, which means that there exists $\gamma < \beta$ such that $b \not\sqsubseteq f^\gamma(\top)$.

  Now any move of $\exists$ is some l with $b \sqsubseteq f(l)$. Therefore $l \not\sqsubseteq f^\gamma(\top)$, since otherwise $b \sqsubseteq f(l) \sqsubseteq f(f^\gamma(\top)) = f^{\gamma+1}(\top) \sqsubseteq f^\beta(\top)$ (since $\gamma + 1 < \beta$). Hence there must be $b' \ll l$ with $b' \not\sqsubseteq f^\gamma(\top)$. Otherwise, as above, if for all $b' \ll l$ we had $b' \sqsubseteq f^\gamma(\top)$, then by continuity of the lattice, we would conclude $l = \bigsqcup\{b' \mid b' \ll l\} \sqsubseteq f^\gamma(\top)$. Such a b' can be chosen by $\forall$, and the game continues.

– If $\beta = \gamma + 1$ we know that $b \not\sqsubseteq f^\beta(\top) = f(f^\gamma(\top))$.

Any move of $\exists$ is l with $b \sqsubseteq f(l)$, which as above implies that $l \not\sqsubseteq f^\gamma(\top)$ and thus the existence of $b' \ll l$ with $b' \not\sqsubseteq f^\gamma(\top)$. The basis element b' is chosen by $\forall$ and the game continues.

Hence $\forall$ always has a move, ending up in $b' \not\sqsubseteq f^\gamma(\top)$, from which there exists a winning strategy for $\forall$ by the induction hypothesis.

Observe that cases of a μ- and a ν-equation are not completely symmetric. In the completeness part, for showing that $l \sqsubseteq \nu f$ we use the fact that νf is the greatest post-fixpoint. Instead, for showing that $l \sqsubseteq \mu f$ we use the fact that $l \sqsubseteq f^\alpha(\bot)$ for some α and provide a proof that we can descend to $\bot$, similarly to what happens for ranking functions in termination analysis. Note that in order to guarantee that we truly descend, also below limit ordinals, we require that $\forall$ plays b with $b \ll l$. Then we can use the fact that whenever b is way-below a directed join, then it is smaller than one of the elements over which the join is taken. We remark that choosing b with $b \sqsubseteq l$ instead would not be sufficient (see Proposition 7.5.11). In the correctness part, despite the asymmetry, both proofs use the fact that each element is the join of all elements way-below it, for which it is essential to be in a continuous lattice (see Proposition 7.5.9). Instead, for completeness, the continuity hypothesis does not play a role.

For the general case, correctness and completeness of the game are proved by relying on the notions of μ- and ν-approximant. Additionally, a general version of the completeness proof fails in case one replaces $\ll$ by $\sqsubseteq$.

We prove the two properties separately where an overview of the proof structure is given in Table 7.2. Completeness exploits a result that shows how $\exists$ can play descending along a chain of μ-approximants and, as in the case of a single equation, it can be proved for general lattices, without assuming the continuity hypothesis.

Lemma 7.5.3: Descending on μ-approximants

Let E be a system of m equations over a lattice L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$. For each μ-approximant $\boldsymbol{l} \in L^m$ and $(b, i) \in \mathbf{A}(\boldsymbol{l})$ there exists a μ-approximant $\boldsymbol{l}' \in \mathbf{E}(b, i)$ such that $ord(\boldsymbol{l}) \succeq_i ord(\boldsymbol{l}')$. Moreover, if $\eta_i = \mu$, the i-th component strictly decreases and thus the inequality is strict.

Proof: Let $\boldsymbol{l} \in L^m$ be a μ-approximant and let (b, i) in $\mathbf{A}(\boldsymbol{l})$, i.e., $b \in B_L$ and $i \in \underline{m}$ with $b \ll l_i$. We distinguish various cases:

Table 7.2: Overview of the building blocks for the completeness and correctness proofs of the fixpoint game. Both proofs rely on μ- and ν-approximates where the former is used within the completeness part and the second in the correctness proof. Continuity is needed to prove correctness where in general completeness fails if one only works with $\sqsubseteq$.

<table>
<tr><th colspan="3">Fixpoint Games over Continuous Lattices</th></tr>
<tr><th colspan="2">Completeness</th><th>Correctness</th></tr>
<tr><td colspan="2">Descending on μ-approximants
(Lemma 7.5.3)</td><td rowspan="3">Ascending on ν-approximants
(Lemma 7.5.6)
Continuity</td></tr>
<tr><td>Non-Algebraic</td><td>Algebraic</td></tr>
<tr><td>$\ll$</td><td>$\sqsubseteq$ or $\ll$</td></tr>
</table>

1. ($\eta_i = \mu$) This means that $l_i = f^{\alpha}_{i,\boldsymbol{l}}(\bot)$ for some ordinal α. Since $f^{0}_{i,\boldsymbol{l}}(\bot) = \bot$ and $b \ll \bot$ would imply $b = \bot$, while $\bot \notin B_L$, necessarily $\alpha \neq 0$. We distinguish two subcases:

   (a) $\alpha = \beta + 1$ *is a successor ordinal*
   Let $l'_i = f^{\beta}_{i,\boldsymbol{l}}(\bot)$ and $(l'_1, \ldots, l'_{i-1}) = sol(E[\boldsymbol{x}_{i+1,m} := \boldsymbol{l}_{i+1,m}][x_i := l'_i])$. Then define

   $$\boldsymbol{l'} = (l'_1, \ldots, l'_{i-1}, l'_i, \boldsymbol{l}_{i+1,m})$$

   Observe that $\boldsymbol{l'}$ is a μ-approximant by Lemma 7.3.12. Moreover $\boldsymbol{l'} \in \mathbf{E}(b, i)$. In fact

   $$\begin{aligned} b \sqsubseteq l_i &= f^{\beta+1}_{i,\boldsymbol{l}}(\bot) \\ &= f_{i,\boldsymbol{l}}(f^{\beta}_{i,\boldsymbol{l}}(\bot)) \\ &= f_{i,\boldsymbol{l}}(l'_i) \\ &= f_i(sol(E[\boldsymbol{x}_{i+1,m} := \boldsymbol{l}_{i+1,m}][x_i := l'_i]), l'_i, \boldsymbol{l}_{i+1,m}) \\ &= f_i(l'_1, \ldots, l'_{i-1}, l'_i, \boldsymbol{l}_{i+1,m}) \\ &= f_i(\boldsymbol{l'}) \end{aligned}$$

   Finally, note that $ord(\boldsymbol{l'}) \prec_i ord(\boldsymbol{l})$ since vectors $\boldsymbol{l}$ and $\boldsymbol{l'}$ coincide on the components $i+1, \ldots, m$, and $l_i = f^{\beta+1}_{i,\boldsymbol{l}}(\bot)$ while $l'_i = f^{\beta}_{i,\boldsymbol{l}}(\bot)$.

   (b) α *is a limit ordinal*
   Since $b \ll l_i = f^{\alpha}_{h,\boldsymbol{l}}(\bot) = \bigsqcup_{\beta<\alpha} f^{\beta}_{i,\boldsymbol{l}}(\bot)$, which is a directed join, by

definition of the way-below relation, there is $\beta < \alpha$ such that $b \sqsubseteq f^{\beta}_{i,\boldsymbol{l}}(\bot)$. We set $l'_i = f^{\beta}_{i,\boldsymbol{l}}(\bot)$ and $(l'_1, \ldots, l'_{i-1}) = sol(E[\boldsymbol{x}_{i+1,m} := \boldsymbol{l}_{i+1,m}][x_i := l'_i])$. Then we define

$$\boldsymbol{l'} = (l'_1, \ldots, l'_{i-1}, l'_i, \boldsymbol{l}_{i+1,m})$$

The vector $\boldsymbol{l'}$ is a μ-approximant by Lemma 7.3.12. Moreover $\boldsymbol{l'} \in \mathbf{E}(b, i)$ since

$$\begin{aligned}
& b \sqsubseteq l'_i \\
& \text{[since } l'_i = f^{\beta}_{i,\boldsymbol{l}}(\bot) \text{ is a post-fixpoint]} \\
& \sqsubseteq f_{i,\boldsymbol{l}}(l'_i) \\
& = f_i(sol(E[\boldsymbol{x}_{i+1,m} := \boldsymbol{l}_{i+1,m}][x_i := l'_i]), l'_i, \boldsymbol{l}_{i+1,m}) \\
& = f_i(l'_1, \ldots, l'_{i-1}, l'_i, \boldsymbol{l}_{i+1,m}) \\
& = f_i(\boldsymbol{l'})
\end{aligned}$$

Finally, note that $ord(\boldsymbol{l'}) \prec_i ord(\boldsymbol{l})$ since vectors $\boldsymbol{l}$ and $\boldsymbol{l'}$ coincide on the components $i+1, \ldots, m$, and $l_i = f^{\alpha}_{i,\boldsymbol{l}}(\bot)$ while $l'_i = f^{\beta}_{i,\boldsymbol{l}}(\bot)$, with $\beta < \alpha$.

2. $(\eta_i = \nu)$
In this case $l_i = \nu(f_{i,\boldsymbol{l}})$. Let $(l'_1, \ldots, l'_{i-1}) = sol(E[\boldsymbol{x}_{i,m} := \boldsymbol{l}_{i,m}])$. Then define

$$\boldsymbol{l'} = (l'_1, \ldots, l'_{i-1}, \boldsymbol{l}_{i,m})$$

The vector $\boldsymbol{l'}$ is a μ-approximant by Lemma 7.3.12. Moreover, observe that $\boldsymbol{l'} \in \mathbf{E}(b, i)$, since

$$\begin{aligned}
& b \sqsubseteq l_i \\
& \text{[since } l_i \text{ is a fixpoint]} \\
& = f_{i,\boldsymbol{l}}(l_i) \\
& = f_i(sol(E[\boldsymbol{x}_{i,m} := \boldsymbol{l}_{i+1,m}]), \boldsymbol{l}_{i,m}) \\
& = f_i(l'_1, \ldots, l'_{i-1}, \boldsymbol{l}_{i,m}) \\
& = f_i(\boldsymbol{l'})
\end{aligned}$$

Finally, note that $ord(\boldsymbol{l'}) \preceq_i ord(\boldsymbol{l})$ since vectors $\boldsymbol{l}$ and $\boldsymbol{l'}$ coincide on the components $i, \ldots, m$.

□

The previous result allows us to prove that player $\exists$ can always win starting from a μ-approximant. Roughly, relying on Lemma 7.5.3, we can prove that player $\exists$ can play on μ-approximants in a way that each time the i-th equation is chosen, the ordinal vector associated to the approximant decreases with respect to $\preceq_i$, and it strictly decreases when the i-th equation is a μ-equation. This, together with the fact that the order on ordinals is well-founded, allows one to conclude that either the play is finite and $\exists$ plays last or the highest index on which one can cycle is necessarily the index of a ν-equation. In both cases player $\exists$ wins.

Lemma 7.5.4: $\exists$ wins on μ-approximants

Let E be a system of m equations over a lattice L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$ and let $\boldsymbol{l} \in L^m$ be a μ-approximant. Then in a game starting from $\boldsymbol{l}$ (which is a position of $\forall$) player $\exists$ has a winning strategy.

Proof: We first describe the strategy for player $\exists$ and then prove that it is a winning strategy.

The key observation is that $\exists$ can always play a μ-approximant, where she plays the solution in the first step. In fact, let $\boldsymbol{l'} \in L^m$ be the current μ-approximant. For any possible move $(b', i') \in \mathbf{A}(\boldsymbol{l'})$ of $\forall$, by Lemma 7.5.3 there always exists a move $\boldsymbol{l''} \in \mathbf{E}(b', i')$ of $\exists$ which is a μ-approximant such that $ord(\boldsymbol{l'}) \succeq_i ord(\boldsymbol{l''})$. Additionally, if $\eta_i = \mu$ the inequality is strict.

Since $\exists$ player has always a move, either the play finishes because $\forall$ has no moves, hence $\exists$ wins or the play continues forever.

In this last case, note that, if h is the largest index occurring infinitely often, then necessarily $\eta_h = \nu$, hence $\exists$ wins. In fact, assume by contradiction that $\eta_h = \mu$. Consider the sequence of turns of the play starting from the point where all indexes repeat infinitely often.

Let $\boldsymbol{l'}$, (b', j), $\boldsymbol{l''}$ be consecutive turns. By the choice of h, necessarily $j \leq h$. Moreover, by construction, if

$$ord(\boldsymbol{l'}) \succeq_j ord(\boldsymbol{l''})$$

Observing that for $j \leq j'$ it holds $\boldsymbol{\alpha} \succeq_j \boldsymbol{\alpha'}$ implies $\boldsymbol{\alpha} \succeq_{j'} \boldsymbol{\alpha'}$, we deduce that

$$ord(\boldsymbol{l'}) \succeq_h ord(\boldsymbol{l''})$$

i.e., the sequence is decreasing. Moreover, since $\eta_h = \mu$, whenever $j = h$, $ord(\boldsymbol{l'}) \succ_h ord(\boldsymbol{l''})$, i.e., the sequence strictly decreases. This contradicts the well-foundedness of $\succ_h$. □

Since the solution of a system of equation is a μ-approximant (the greatest one), completeness is an easy corollary of Lemma 7.5.4.

Corollary 7.5.5: Completeness

Let E be a system of m equations over a lattice L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$. Given any μ-approximant $\boldsymbol{l} \in L^m$, $b \in B_L$ and $i \in \underline{m}$, if $b \sqsubseteq l_i$ then $\exists$ has a winning strategy from position (b, i).

Proof: Just observe that at the first turn $\exists$ can play the μ-approximant $\boldsymbol{l}$ that is in $\mathbf{E}(b, i)$ by hypotheses. Then using Lemma 7.5.4 we conclude that $\exists$ wins. □

For correctness we rely on a result, dual to Lemma 7.5.3, that allows to ascend along ν-approximants. However, in this case, the fact of working in a continuous lattice is crucial (see Proposition 7.5.9).

Lemma 7.5.6: Ascending on ν-approximants

Let E be a system of m equations over a continuous lattice L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$. Given a ν-approximant $\boldsymbol{l} \in L^m$, an element $b \in B_L$ and an index $i \in \underline{m}$ with $b \not\sqsubseteq l_i$, for all tuples $\boldsymbol{l'} \in \mathbf{E}(b, i)$ there are a ν-approximant $\boldsymbol{l''}$ and $(b'', j) \in \mathbf{A}(\boldsymbol{l'})$ such that (1) $b'' \not\sqsubseteq l''_j$ and (2) $ord(\boldsymbol{l}) \succeq_i ord(\boldsymbol{l''})$. Moreover, if $\eta_i = \nu$, the i-th component strictly decreases and thus the inequality in item 2 above is strict.

Proof: Let $\boldsymbol{l} \in L^m$ be a ν-approximant, let $b \in B_L$ and let $i \in \underline{m}$ with $b \not\sqsubseteq l_i$. Take $\boldsymbol{l'} \in \mathbf{E}(b, i)$, i.e., such that $b \sqsubseteq f_i(\boldsymbol{l'})$. We prove that there are a ν-approximant $\boldsymbol{l''}$ and $(b'', j) \in \mathbf{A}(\boldsymbol{l'})$ satisfying (1) and (2) above, by distinguishing various cases:

(i) ($\eta_i = \mu$) Define $\boldsymbol{l''} = (sol(E[\boldsymbol{x}_{i,m} := \boldsymbol{l}_{i,m}]), l_i, \boldsymbol{l}_{i+1,m})$, which is a ν-approximant by Lemma 7.3.12. Note that, since $l_i = \mu(f_{i,\boldsymbol{l}})$,

$$l_i = f_{i,\boldsymbol{l}}(l_i) = f_i(sol(E[\boldsymbol{x}_{i,m} := \boldsymbol{l}_{i,m}]), l_i, \boldsymbol{l}_{i+1,m}) = f_i(\boldsymbol{l''})$$

We first prove prove (1), i.e., that there exists $(b'', j) \in \mathbf{A}(\boldsymbol{l'})$, i.e., $j \in \underline{m}$ and $b'' \in B_L$, $b'' \ll l'_j$ with $b'' \not\sqsubseteq l''_j$. In fact, otherwise, if for any j and $b'' \ll l'_j$ we had $b'' \sqsubseteq l''_j$, then for any j, since B_L is a basis and L a continuous lattice:

$$l'_j = \bigsqcup\{b'' \mid b'' \in B_L \wedge\ b'' \ll l'_j\} \sqsubseteq l''_j.$$

However, by monotonicity of f_i, this would imply $f_i(\boldsymbol{l'}) \sqsubseteq f_i(\boldsymbol{l''}) = l_i$, that together with the hypothesis $b \sqsubseteq f_i(\boldsymbol{l'})$, would contradict $b \not\sqsubseteq l_i$.

For point (2), note that $ord(\boldsymbol{l}') \preceq_i ord(\boldsymbol{l})$ since vectors $\boldsymbol{l}$ and $\boldsymbol{l}'$ coincide on all components $i, \ldots, m$, and $l_i = f^{\alpha}_{i,\boldsymbol{l}}(\bot)$.

(ii) ($\eta_i = \nu$) This means that $l_i = f^{\alpha}_{i,\boldsymbol{l}}(\top)$ for some ordinal α, necessarily $\alpha \neq 0$ (since otherwise $l_i = \top$ and $b \not\sqsubseteq l_i$ could not hold). We distinguish two subcases

(a) $\alpha = \beta + 1$ *is a successor ordinal*
Define $\boldsymbol{l}'' = (sol(E[\boldsymbol{x}_{i+1,m} := \boldsymbol{l}_{i+1,m}][x_i := f^{\beta}_{i,\boldsymbol{l}}(\top)]), f^{\beta}_{i,\boldsymbol{l}}(\top), \boldsymbol{l}_{i+1,m})$. Then we have

$$\begin{aligned} & b \not\sqsubseteq l_i \\ & = f^{\beta+1}_{i,\boldsymbol{l}}(\top) \\ & = f_{i,\boldsymbol{l}}(f^{\beta}_{i,\boldsymbol{l}}(\top)) \\ & = f_i(sol(E[\boldsymbol{x}_{i+1,m} := \boldsymbol{l}_{i+1,m}][x_i := f^{\beta}_{i,\boldsymbol{l}}(\top)]), f^{\beta}_{i,\boldsymbol{l}}(\top), \boldsymbol{l}_{i+1,m}) \\ & = f_i(\boldsymbol{l}'') \end{aligned}$$

Recalling that $b \sqsubseteq f_i(\boldsymbol{l}')$, as in case (i) we deduce point (1), i.e., that there exists $(b'', j) \in \mathbf{A}(\boldsymbol{l}')$ such that $b'' \not\sqsubseteq l''_j$.

Concerning point (2), note that $ord(\boldsymbol{l}') \prec_i ord(\boldsymbol{l})$ since vectors $\boldsymbol{l}$ and $\boldsymbol{l}'$ coincide on the components $i+1, \ldots, m$, and $l_i = f^{\beta+1}_{i,\boldsymbol{l}}(\bot)$ while $l'_i = f^{\beta}_{i,\boldsymbol{l}}(\bot)$.

(b) α *is a limit ordinal*
In this case

$$b \not\sqsubseteq l_i = f^{\alpha}_{i,\boldsymbol{l}}(\top) = \textstyle\bigsqcap_{\beta<\alpha} f^{\beta}_{i,\boldsymbol{l}}(\top) = \bigsqcap_{\beta<\alpha} f^{\beta+1}_{i,\boldsymbol{l}}(\top)$$

Therefore there exists $\beta < \alpha$ such that $b \not\sqsubseteq f^{\beta+1}_{i,\boldsymbol{l}}(\top)$. Hence, we can define $l''_i = f^{\beta}_{i,\boldsymbol{l}}(\top)$ and take the ν-approximant

$$\boldsymbol{l}'' = (sol(E[\boldsymbol{x}_{i+1,m} := \boldsymbol{l}_{i+1,m}][x_i := l''_i]), l''_i, \boldsymbol{l}_{i+1,m})$$

Then we have

$$\begin{aligned} & b \not\sqsubseteq f^{\beta+1}_{i,\boldsymbol{l}}(\top) \\ & = f_{i,\boldsymbol{l}}(f^{\beta}_{i,\boldsymbol{l}}(\top)) \\ & = f_i(sol(E[\boldsymbol{x}_{i+1,m} := \boldsymbol{l}_{i+1,m}][x_i := f^{\beta}_{i,\boldsymbol{l}}(\top)]), f^{\beta}_{i,\boldsymbol{l}}(\top), \boldsymbol{l}_{i+1,m}) \\ & = f_i(\boldsymbol{l}'') \end{aligned}$$

and thus, again, recalling that $b \sqsubseteq f_i(\boldsymbol{l}')$, as in case (i) we deduce point (1), i.e., that there exists $(b'', j) \in \mathbf{A}(\boldsymbol{l}')$ such that $b'' \not\sqsubseteq l''_j$.

Concerning point (2), note that $ord(\boldsymbol{l}'') \prec_i ord(\boldsymbol{l})$ since vectors $\boldsymbol{l}$ and $\boldsymbol{l}''$ coincide on the components $i{+}1, \ldots, m$, and $l_i = f^{\alpha}_{i,\boldsymbol{l}}(\bot)$ while $l''_i = f^{\beta}_{i,\boldsymbol{l}}(\bot)$, with $\beta < \alpha$.

□

As in the dual case, correctness is an easy corollary of the above lemma, recalling that the solution is the least ν-approximant.

Lemma 7.5.7: Correctness

Let E be a system of m equations over a continuous lattice L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$. For a ν-approximant $\boldsymbol{l} \in L^m$, $b \in B_L$ and $i \in \underline{m}$, if $b \not\sqsubseteq l_i$ then $\forall$ has a winning strategy from position (b, i).

Proof: We first describe the strategy for the universal player and then prove that it is a winning strategy.

Let $\boldsymbol{l} \in L^m$ be a ν-approximant, $b \in L$ and $i \in \underline{m}$ such that $b \not\sqsubseteq l_i$. Starting from (b, i), for any possible move $\boldsymbol{l}' \in \mathbf{E}(b, i)$ of $\exists$. Then $\forall$ can play a pair $(b', j) \in \mathbf{A}(\boldsymbol{l}')$, whose existence is ensured by Lemma 7.5.6, such that there is a ν-approximant $\boldsymbol{l}''$ satisfying $b'' \not\sqsubseteq l''_j$ and $ord(\boldsymbol{l}'') \preceq_i ord(\boldsymbol{l})$. Additionally, if $\eta_i = \nu$ the inequality is strict.

According to the strategy defined above $\forall$ player has always a move. Thus either the play finishes because $\exists$ has no moves, hence $\forall$ wins or the play continues forever.

In this last case, with an argument dual with respect to that in Lemma 7.5.4, we can show that if h is the largest index occurring infinitely often, then necessarily $\eta_h = \mu$, hence $\forall$ win. □

Combining Corollary 7.5.5 and Lemma 7.5.7 we reach the desired result.

Theorem 7.5.8: Correctness and Completeness

Given a system of m equations E over a continuous lattice L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$ with solution $\boldsymbol{u}$, then for all $b \in B_L$ and $i \in \underline{m}$,

$$b \sqsubseteq u_i \quad \text{iff} \quad \exists \text{ has a winning strategy from position } (b, i).$$

Proof: Immediate corollary of Lemma 7.3.11, Corollary 7.5.5 and Lemma 7.5.7. □

Note that even when the fixpoint is reached in more than ω steps, thanks to the fact that the order on the ordinals is well-founded and players *descend* over the order, ordinals do not play an explicit role in the game. In particular plays are not

transfinite and whenever $\forall$ or $\exists$ win due to the fact that the other player cannot make a move, this happens after a *finite* number of steps. This can be a bit surprising at first since the game works for general continuous lattices, including, for instance, intervals over the reals.

We close this subsection by proving two results that, in a sense, show that the choice of continuous lattices and the design of the game based on the way-below relation are *the right ones*.

We first observe that the restriction to continuous lattices is not only sufficient but also necessary for the correctness of the game. The proofs of the following two propositions could be found in [BK+18].

Proposition 7.5.9: Correctness – exactly in Continuous Lattices

Let L be a lattice and let B_L be a fixed basis with $\bot \notin B_L$. The game is correct for every system of equations over L if and only if L is continuous.

Example 7.5.10

As a counterexample, consider the lattice W in Figure 7.1, which is not continuous and let B_W be any basis such that $0 \notin B_W$. First note that necessarily $a \in B_W$, otherwise $a \neq \bigsqcup\{x \in B_W \mid x \sqsubseteq a\} = \bigsqcup \emptyset = 0$. Secondly, $\downarrow a = \{0\}$ since $a \not\ll a$. Then, consider the equation $x =_\mu f(x)$, where the function $f\colon W \to W$ is defined by $f(0) = 0$, and $f(x) = \omega$ for $x \neq 0$. Clearly f is monotone and its least fixpoint is $\mu f = 0$. However, the player $\exists$ can win any play of the game from position a, despite the fact that $a \not\sqsubseteq \mu f = 0$. In fact, the first move of $\exists$ can be a, since $a \sqsubseteq f(a) = \omega$. But then player $\forall$ has no moves since $\downarrow a \cap B_W = \emptyset$. And so player $\exists$ always wins while she should not.

The second observation is that using the lattice order instead of the way-below relation may break completeness. More precisely, consider the natural variant of the game where the way-below relation is replaced by the lattice order. Let us call it *weak game*. Since the set of possible moves of player $\forall$ is enlarged, correctness clearly continues to hold. Instead, as we hinted before, completeness could fail. We show that it is exactly on algebraic lattices that completeness still holds for the weak game.

Proposition 7.5.11: Way-below is needed in Non-Algebraic Lattices

Let L be a lattice. The weak game is complete on every system of equations over L if and only if B_L consists of compact elements (which in turn means that L is algebraic).

Note that when the elements of the basis are compact, the way-below relation with respect to elements of the basis is the lattice order. Hence the result above essentially states that the weak game is complete exactly when it coincides with the original game, thus further supporting the appropriateness of our formulation of the game.

Example 7.5.12

As a counterexample, consider the continuous lattice $[0,1]$ with the usual order and basis $B_{[0,1]} = \mathbb{Q} \cap (0,1]$. Recall that $[0,1]$ is not algebraic (the only compact element is 0) and way-below relation is the strict order $<$. Let $g\colon [0,1] \to [0,1]$ be the function defined by $g(x) = \frac{x+1}{2}$. The fixpoint equation $x =_\mu g(x)$ has solution $\mu g = 1$.

In the weak game, from position $l \in [0,1]$, player $\forall$ can play any $b \leq l$ (instead, of $b < l$). Then player $\exists$ loses any play starting from position 1, despite the fact that $1 \leq \mu g = 1$. In fact, the only possible move of player $\exists$ is 1, and $\forall$ can play any $x \leq 1$. In particular, playing 1 the game will continue forever and will thus be won by $\forall$.

Notice that, instead, in the original game, from position 1, player $\forall$ has to play an element $1 - \varepsilon$ for some $\varepsilon > 0$. Then, it is easy to see that at each step i player $\exists$ will be able to play some $z_i \leq 1 - 2^i\varepsilon$. This means that after finitely many steps $\exists$ will be allowed to play 0, thus leaving no possible answer to $\forall$ and winning the game.

We close this section getting back to constant propagation analysis introduced in Section 7.4.2.

Example 7.5.13

Recall that the system of fixpoint equations expressing the analysis in Figure 7.5 had solution $\rho_1 = \bot$, $\rho_2 = \bot[\mathtt{y} \mapsto 6]$, $\rho_3 = \rho_4 = \bot[\mathtt{x} \mapsto 7]$. We next describe a game that shows that indeed $\bot[\mathtt{x} \mapsto 7] \sqsubseteq \rho_4$ and hence $\mathtt{x}$ has constant value 7 at block 4. Induced by the order, every directed set of size greater than two includes

$\top$.

The game $(\bot[\mathtt{x} \mapsto 7], 4)$ proceeds as follows:

$$\begin{aligned}
&\overset{\exists}{\rightsquigarrow} (\bot, \bot, \bot[\mathtt{x} \mapsto 7], \bot) && (\bot[\mathtt{x} \mapsto 7] \sqsubseteq f_4(\bot, \bot, \bot[\mathtt{x} \mapsto 7], \bot)) \\
&\overset{\forall}{\rightsquigarrow} (\bot[\mathtt{x} \mapsto 7], 3) && (\bot[\mathtt{x} \mapsto 7] \ll p_3 = \bot[\mathtt{x} \mapsto 7]) \\
&\overset{\exists}{\rightsquigarrow} (\bot, \bot[\mathtt{y} \mapsto 6], \bot, \bot[\mathtt{x} \mapsto 7]) && (\bot[\mathtt{x} \mapsto 7] \sqsubseteq f_3(\bot, \bot[\mathtt{y} \mapsto 6], \bot, \bot[\mathtt{x} \mapsto 7]) \\
&\overset{\forall}{\rightsquigarrow}
\end{aligned}$$

Now the universal player has two options: either choose $(\bot[\mathtt{x} \mapsto 7], 4)$, which brings him back to an earlier game situation and might potentially lead to an infinite game. Since we are considering greatest fixpoints, this means that $\exists$ wins. If he chooses the other option (i.e. $(\bot[\mathtt{y} \mapsto 6], 2)$), the game continues as follows, where eventually $\forall$ has no move left and $\exists$ wins as well:

$$(\bot[\mathtt{y} \mapsto 6], 2) \overset{\exists}{\rightsquigarrow} (\bot, \bot, \bot, \bot) \overset{\forall}{\not\rightsquigarrow}$$

7.5.3 Comparison with Other Games

We discuss how our parity game over systems of fixpoint equations relates to other game characterizations. Therefore, we consider bisimulation for LTS and model checking via the μ-calculus.

Bisimulation Games: The bisimulation games presented in Chapter 3 yield a game-theoretical interpretation of behavioral equivalence. Analogous to the fixpoint game, we have two players, spoiler (**S**) and duplicator (**D**), where **S** wants to show that two states are non-bisimilar and **D** wants to prove the opposite.

Given a LTS $(X, \Sigma, \to)$ we recall the function $\mathcal{F}(\mathcal{R}) : \mathcal{P}(X \times X) \to \mathcal{P}(X \times X)$ introduced in Section 2.2.2 defined as follows:

$$\begin{aligned}
\mathcal{F}(\mathcal{R}) = \quad & \{(x, y) \in X \times X \mid \forall\, a \in \Sigma : \\
& \quad \forall\, x' \text{ with } x \xrightarrow{a} x', \exists\, y' \text{ s.t. } y \xrightarrow{a} y' \text{ and } (x', y') \in \mathcal{R}; \qquad \text{and} \\
& \quad \forall\, y' \text{ with } y \xrightarrow{a} y', \exists\, x' \text{ s.t. } x \xrightarrow{a} x' \text{ and } (x', y') \in \mathcal{R} \\
& \}
\end{aligned} \tag{7.6}$$

Therefore, we obtain the following single equation system

$$\mathcal{R} =_\nu \mathcal{F}(\mathcal{R})$$

where the solution characterizes bisimilarity. Regarding our framework the lattice is given by $L = \mathcal{P}(X \times X)$ and the basis B_L consists of all singleton relations.

In the fixpoint game $\forall$ has the same role as **S**, since $\{(x, y)\} \not\subseteq \nu\mathcal{F}$ is equivalent to $x \nsim y$. Therefore, **D** and $\exists$ pursue the same target, to prove $x \sim y$. In case $\exists$ has a winning-strategy for the initial situation $(\{(x, y)\}, 1)$ one can conclude that $x \sim y$ holds or equivalently $\{(x, y)\} \subseteq \nu\mathcal{F}$.

Example 7.5.14

As an example, consider the LTS given in Figure 7.7. A bisimulation is given by $\{(x_0, y_0), (x_1, y_1), (x_1, y_2), (x_2, y_3), (x_3, y_3), (x_4, y_4)\}$.

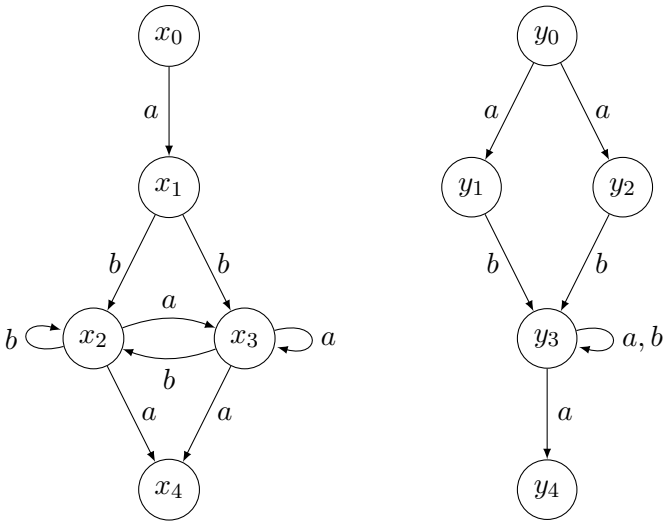


Figure 7.7: A LTS where $x_0 \sim y_0$ holds and the existential player $\exists$ has a winning-strategy for (x_0, y_0) in the fixpoint game.

Example 7.5.15

We apply the fixpoint game to the LTS given in Figure 7.7. Due to the fact that the system of interest includes a single equation we write b instead of (b, i) for the $\exists$-positions.

The initial situation is given by $\{(x_0, y_0)\}$ and possible game sequences are described in Figure 7.8.

The positions of the existential player $\exists$ are given by diamonds and some options for valid moves by $\exists$ are represented by rectangles, which in turn yield the positions for the universal player $\forall$. Note, that $\forall$ has to work with singletons due to the definition of B_L.

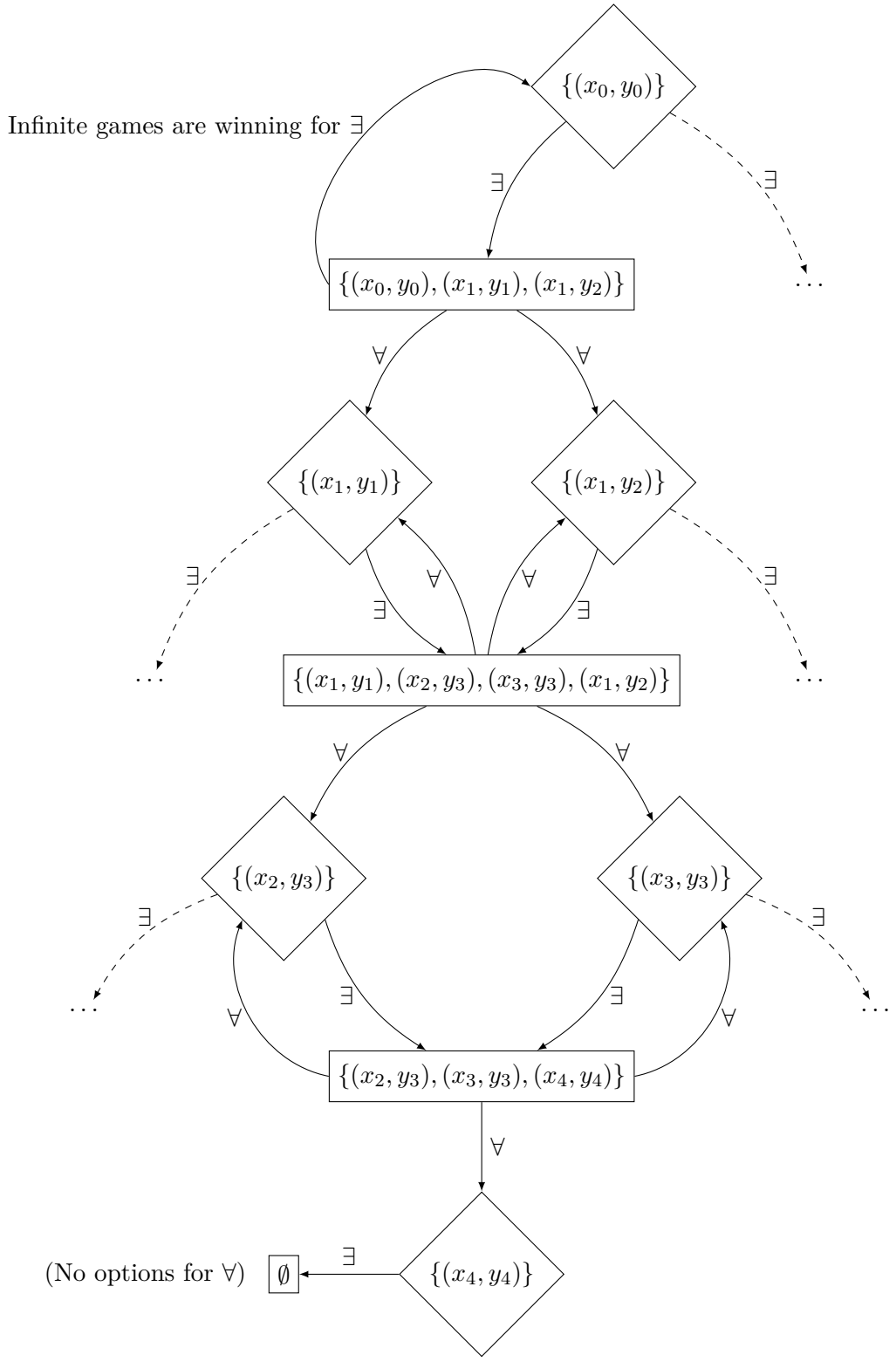


Figure 7.8: Partial game board for the initial situation $\{(x_0, y_0)\}$. Other options for $\exists$ are denoted with dashed lines and all the positions $(b, 1)$ with $b \in L_B$ (the diamonds) have a ν-index.

As depicted in the partial game board in Figure 7.8 the ∃-player either wins because the ν-index is visited infinitely often or ∀ has no move due to $\emptyset$, which is a valid move for ∃ at position $\{(x_4, y_4)\}$.
For the sake of completeness, consider the non-bisimilar state pair (x_4, y_3). Note, that ∃ has no way to win by the rules of the game since no matter which R ∃ plays the pair (x_4, y_3) will not be included in $\mathcal{F}(R)$.

This game is more reminiscent of Baltag's game [Bal99] than the two game versions given by Definition 3.2.1 and Definition 3.3.1. All games have in common that the moves of the existential player (or duplicator) consider the direct successors of the state pair of the current game round.

The difference lies in the fact that in Baltag's game as well as in the fixpoint game, the first move is given by the existential player via a relation R satisfying $\{(x, y)\} \subseteq \mathcal{F}(R)$ or equivalently $(\alpha(x), \alpha(y)) \in \Gamma R$ where α is the given LTS and Γ lifts R to a relation $\Gamma R \subseteq FX \times FX$ with $F = \mathcal{P}(A \times \_)$ in correspondence to the generalization of bisimulation by Aczel and Mendler [Bal99]. Thus, all the direct successors of x, y are taken into account via such a relational move. In case $\{(x, y)\} \subseteq \nu\mathcal{F}$ the existential player works with (partial) bisimulations which in turn yields positions satisfying $\subseteq \nu\mathcal{F}$. Otherwise (i.e. $x \nsim y$), either at least one non-bisimilar state pair exists by the move of ∃ such that ∀ will eventually win or no relation R with $\{(x, y)\} \subseteq \nu\mathcal{F}(R)$ exists.

As already mentioned above, in the game versions presented in Chapter 3 the duplicator has the same role as ∃, but in those games she only has to take care of the states included by the predicate given by the move of the spoiler, i.e. **D** has to mimic the moves of **S**.

The Modal μ-Calculus Parity Games We already introduced how μ-calculus formulas are used in the process of system verification (see Section 7.4.1). It is generally considered difficult to infer the interpretation of a μ-calculus formula using the semantics defined in Section 7.4.1. Thus, the motivation to combine *parity games* with the modal μ-calculus lies in the simplification of the interpretation process [BW18].

Therefore, we will consider parity games over μ-calculus formulas, where our game also relates to other classical techniques for model checking the μ-calculus, as tableau systems or automata (see, e.g., [Eme85; SW91]). A detailed comparison with the tableaux technique can be found in [BK+18].

The following definitions and examples are derived form [BW18; Ven08; Jur00] where [Sti95] presents a *property checking game* over modal μ-calculus formulas with parity-conditions for infinite sequences originating from [EJ91].

We refer to the formal definition of parity games [Jur00] (see Definition 7.2.13) and present an informal description of how a formula can be encoded into a game [BW18].

To compare the winning conditions of our fixpoint game over equational systems with the game version played on a μ-calculus formula φ, we need to encode φ into a parity game. The reduction of the model checking problem to a parity game relies on two aspects: the inductive syntax given by φ and the alternation depth. These transformations are described in more detail in [BW18] where positions are tuples (s, φ') with s being a state of the underlying state-based model and φ' is a formula of the smallest set of formulas containing φ which is closed under subformulas and the unfolding of φ. Here we just give an overview of the main intuitions behind this concept:

1. An example for a base case (s, p) is given by a formula $\varphi = p$. In case $s \models p$ holds, $\exists$ is winning if we add φ to $V_\Box$. This way $\forall$ has no outgoing edge and loses. If $s \not\models p$, we add p to $V_\Diamond$ and $\forall$ wins because $\exists$ has no move. Other base cases are treated analogously.

2. For modal operators we consider the $\Box$-operator where the $\Diamond$-operator is handled similarly. First of all we consider the position $(s, [a]\varphi)$ and derive for each a-reachable state t a new transition: $(s, [a]\varphi) \to (t, \varphi)$ if $s \xrightarrow{a} t$. A position $(s, [a]\varphi)$ is winning for $\exists$ if for all a-reachable states t i.e. $s \xrightarrow{a} t$ we have that $t \models \varphi$. Therefore the derived positions should belong to $\forall$ since in case one state t exists which does not satisfy φ the universal player should have the opportunity to select this path. Dually, $(s, \langle a\rangle\varphi)$ should belong to $\exists$.

3. For similar reasons as described in (2) disjunctions belong to $\exists$ and conjunctions to $\forall$.

4. For the fixpoint operators we follow the intuition behind a fixpoint and define transitions from $(s, \eta X.\varphi(X))$ to $(s, \varphi(\eta X.\varphi(X)))$. Since there exists only one transition it does not matter to which player such a position belongs.

5. Finally, we need to assign priorities (ranks) to the positions. For a given formula φ starting with ν we assign an even priority to the corresponding position where for formulas starting with μ we use odd numbers. In order to ensure this, alternation depths with respect to the bound variables by the η-operators are used [BW18].

It is not hard to see that this procedure yields the same winning-conditions for $\exists$ and $\forall$ as described for our fixpoint game (see [FLV10] and [EJ91]). Before we compare both games we derive an instance of such a game.

Example 7.5.16

We consider the formula $\varphi = \nu x_2.((\mu x_1.(p \vee \Diamond x_1)) \wedge \Box x_2)$ given in Example 7.5.2.

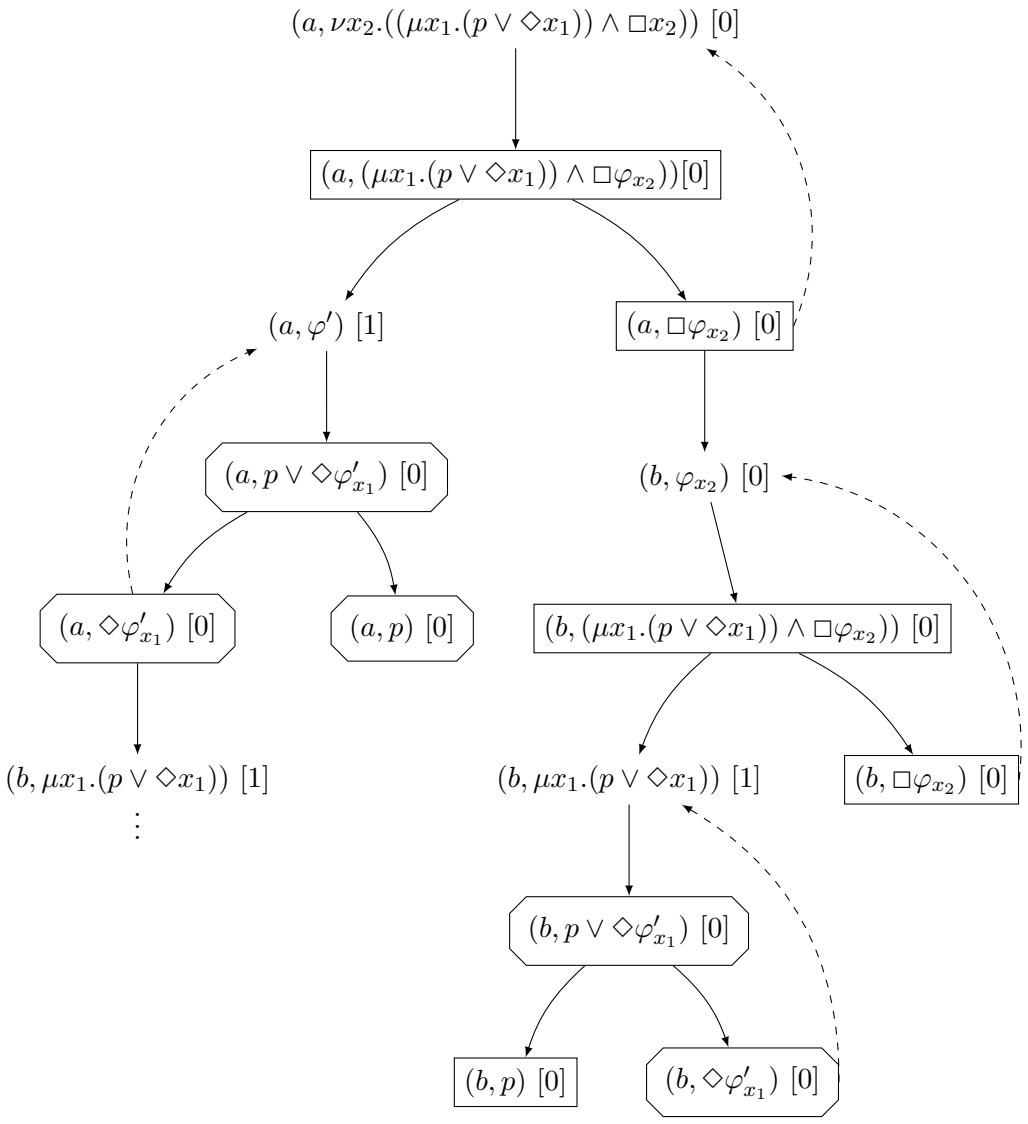


Figure 7.9: The game board derived from a μ-calculus formula [BW18].

The positions and transitions illustrated in Figure 7.9 are constructed based on the previously described transformation rules. For simplification we denote $\varphi' = \mu x_1.(p \vee \Diamond x_1)$. We use rectangles for the positions of $\forall$, chamfered rectangles

for $\exists$ and no shape for the remaining unfolding steps. The priorities are denoted with $[\_]$ and for fixpoints we denote with φ_{x_2} (φ'_{x_1}) the whole formula with respect to the unfolding rule. Note, that both alternation depths (for μ, x_1 and ν, x_2) are 1 since x_2 does not occur in φ' and therefore we get the ranks 1 and 0 respectively.

We are now ready to compare the formula based game with the game over the corresponding system of equations presented in Example 7.5.2. The positions in the fixpoint game depicted in Figure 7.6 refers to the same (sub)formulas and thus it results in an equivalent game board where the fixpoint game does not consider the intermediate positions based on the whole inductive structure of the formula. More precisely, in both versions we move from the initial position to a $\forall$-position which includes three options: going back to the initial position, moving to φ' with respect to a (i.e. to the corresponding equation for the variable x_1), or moving to φ with respect to b.

Next, we consider the last position of those three options. In our game $\forall$ has the opportunity to cycle on this ν-index or move to φ' (i.e. to the corresponding equation for the variable x_1), but this time with respect to b. The game version obtained by the previously described transformation rules enables exactly the same options and therefore $\exists$ has a winning strategy. More precisely, $\forall$ avoids cycling on even priorities (i.e. ν-indizes) and $\exists$ decides for a finite path ending in the position (b, p) or equivalently $(\emptyset, \emptyset)$ in Figure 7.6, where $\forall$ has no further option and therefore he loses the play.

Since a formula can be transformed into an equivalent equational system where each equation is derived from a (sub)formula given by a nested fixpoint operator [CKS92; HSC16], it is not a big surprise to observe that $\exists$ obtains equivalent winning strategies in both game versions. (A detailed discussion with respect to the equivalence of the semantics can be found in [BK+18]). Although, the game board from the system of equations yields a more compact representation, involving alternation depths improves the runtime if one wants to compute the winning strategies (see [Jur00; BK+19a]).

Analogous to the two-player games presented in Chapter 3,5 and 6, winning strategies play a significant role in the context of parity games. Consider a system E of fixpoint equations as described for the modal μ-calculus in Section 7.4.1. For a subset $b \subseteq S$ where S is the state space of some underlying Kripke system, it holds that the states in b satisfy the property characterized by E if and only if $\exists$

has a winning strategy for a game starting with position (b, m) where m denotes the number of fixpoint equations in E. Therefore, the question arises how to compute the winning strategies.

7.6 Winning Strategies and Progress Measures

As mentioned at the beginning of this chapter and considered in the previous section, our game is a parity game (see Definition 2.3.16) and game theoretical semantics provide a comprehensible alternative for logical semantics (see Section 7.4.1).

As a consequence, one is interested whether $\exists$ has a winning strategy or not. Winning strategies are functions that map each winning position to possible moves, which in turn result in winning positions regardless of the opposing player's actions. In parity games, *progress measures* are witnesses for such winning strategies [Jur00].

In this section we just briefly consider progress measures along the lines of [Jur00], influenced by [HSC16] and introduce a general notion of progress measure for fixpoint games over continuous lattices presented in [BK+19a]. We will show how a complete progress measure characterizes the winning positions for the two players.

Given an ordinal α we denote by $[\alpha]^m_\star = \{\beta \mid \beta \leq \alpha\}^m \cup \{\star\}$, the set of ordinal vectors with entries smaller or equal than α, with an added bound $\star$.

Definition 7.6.1: Progress measure

Let L be a continuous lattice and let E be a system of m equations over L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$. Given an ordinal λ, a λ-*progress measure* for E is a function $R\colon B_L \to \underline{m} \to [\lambda]^m_\star$ such that for all $b \in B_L$, $i \in \underline{m}$, either $R(b)(i) = \star$ or there exists $\boldsymbol{l} \in \mathbf{E}(b, i)$ such that for all $(b', j) \in \mathbf{A}(\boldsymbol{l})$ it holds

- if $\eta_i = \mu$ then $R(b)(i) \succ_i R(b')(j)$;
- if $\eta_i = \nu$ then $R(b)(i) \succeq_i R(b')(j)$

A progress measure maps any basis element of the lattice and index $i \in \underline{m}$ to an m-tuple of ordinals, with one component for each equation. Components relative to μ-equations roughly measure how many unfolding steps for the equation would be needed to reach an under-approximation l_i above b, and thus, for $\exists$, to win the game. Components relative to ν-equations, as in the original work of [Jur00], are less relevant, as we will see.

Intuitively, whenever $R(b)(i) \neq \star$, the progress measure R provides an evidence of the existence of a winning strategy for $\exists$ in a play starting from (b, i). The tuple

$\boldsymbol{l}$, whose existence is required by the definition, is a move of player $\exists$ such that for any possible answer of $\forall$, the progress measure will not increase with respect to $\preceq_i$, and it will strictly decrease in the case of μ-equations. Since $\prec_i$ is well-founded, this ensures that we cannot cycle on a μ-equation. Also note that whenever the current index is i, all indices lower than i are irrelevant (expressed by the orders $\succeq_i$ resp. $\succ_i$), which is related to the fact that the highest index which is visited infinitely often is the only relevant index for determining the winner of the game. This idea is formalized in the following lemma.

Lemma 7.6.2: Progress Measures are Strategies

Let L be a continuous lattice and let E be a system of m equations over L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$ with solution $\boldsymbol{u}$. For any $b \in B_L$ and $i \in \underline{m}$, if there exists some ordinal λ and a λ-progress measure R such that $R(b)(i) \preceq_i (\lambda, \ldots, \lambda)$, then $b \sqsubseteq u_i$.

Proof: We show that $\exists$ has a winning strategy from (b, i). The strategy consists in choosing a move $\boldsymbol{l} \in \mathbf{E}(b, i)$ such that for all $(b', j) \in \mathbf{A}(\boldsymbol{l})$, it holds

- $R(b)(i) \succ_i R(b')(j)$, if $\eta_i = \mu$
- $R(b)(i) \succeq_i R(b')(j)$, if $\eta_i = \nu$

which exists by definition of progress measure.

Now, observe that player $\exists$ can always make its turn. Therefore either the play stops because $\forall$ runs out of moves, hence $\exists$ win. Otherwise, the play is infinite, and, if we denote by h the largest index occurring infinitely often, then $\eta_h = \nu$, hence $\exists$ wins. In fact, assume by contradiction that $\eta_h = \mu$. Consider the sequence of turns of the play starting from the point where all indexes repeat infinitely often and take the m-tuples of ordinals $R(b')(h)$ corresponding to the positions (b', i) where $\exists$ plays. For any two successive elements, say (b', i) and (b'', j), by construction

$$R(b')(i) \succeq_i R(b'')(j)$$

Observing that for $i \leq j$ it holds $\boldsymbol{\alpha} \succeq_i \boldsymbol{\alpha'}$ implies $\boldsymbol{\alpha} \succeq_j \boldsymbol{\alpha'}$, we deduce that

$$R(b')(i) \succeq_h R(b'')(j)$$

i.e., the sequence is decreasing.

Moreover, since $\eta_h = \mu$, whenever $i = h$, the sequence strictly decreases, i.e., $R(b')(i) \succ_h R(b'')(j)$. This contradicts well-foundedness of $\prec_h$. □

The above lemma, in a sense, says that progress measures provide sound characterizations of the solution. However, in general, they are not complete, since whenever $R(b)(i) = \star$ we cannot derive any information on (b, i), i.e., if $\boldsymbol{u}$ is the solution of the system, we cannot conclude that $b \not\sqsubseteq u_i$. This motivates the following definition.

Definition 7.6.3: Complete Progress Measures

Let L be a continuous lattice and let E be a system of equations over L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$ with solution $\boldsymbol{u}$. A λ-progress measure $R\colon B_L \to \underline{m} \to [\lambda]^m_\star$ is called *complete* if for all $b \in B_L$ and $i \in \underline{m}$, if $b \sqsubseteq u_i$ then $R(b)(i) \preceq_i (\lambda, \ldots, \lambda)$.

Observe that in search of a complete progress measure, in principle, we would have to try all ordinals as a bound. One can take as bound the height λ_L of the lattice L [HSC16], which provides a generalization of the *small progress measure* in [Jur00].

The existence of such a small progress measure allows to express a complete progress measure as a least fixpoint, thus providing a technique for computing the progress measure and solving the corresponding system of equations. For more details we refer to [BK+19a] and the discussion in the next section.

7.7 Conclusion and Discussion

Our work combines the theory of program analysis and verification [CC77] with domain theory. Both areas are based on lattice theory and in particular on complete lattices where in the second case continuous lattices have been introduced by Scott as a semantic domain for the λ-calculus [Sco72]. Continuous lattices, which received this name due to their intimate connection with continuous functions, have since found many further applications in the semantics of programming languages [GH+03; AJ94].

Our contributions presented in this thesis can be categorized as follows:

▷ First of all, we developed a game-theoretical characterization for systems of fixpoint equations over continuous lattices. Our game generalizes the parity games for the μ-calculus [BW18; Sti95] since it extends the game-theoretical approach to continuous lattices.

▷ Secondly, we established a generalization of progress measures which represent witnesses for winning strategies in parity games.

As already discussed in detail in Section 7.5.3, regarding μ-calculus model checking our game version refers to the equational representation of system properties, which

is inspired by the work in [HSC16].

The modal μ-calculus is an expressive temporal logic, which originates on an unpublished work of Scott and Bakker and the further development by Kozen [Koz83]. The whole chapter refers to the very detailed overview given by [BW18], which introduces the problem of efficient model checking, which involves the solution of nested fixpoint equations, see, e.g., [BC+97; Sei96; CKS92]. The paper [CKS92] introduced the notion of a hierarchical system of fixpoint equations, on which our paper is based as well. The main focus of this chapter is the reduction of the model checking problem into the question of finding winning strategies for parity games as first described in [EJ91]. A very satisfying technique for solving parity games was proposed in [Jur00] resulting in an algorithm which is exponential only in half of the alternation depth.

Since the fixpoint equations derived from a modal μ-calculus formula φ are based on the nested μ, ν-operators within φ as well as the other games presented in [BW18; Sti95] are inductively derived from φ, the corresponding game boards show conceptual similarities. However, the construction of our game is rather simple in that sense, that it can be directly played on the single equations using the indices of the equations, where the other game boards are based on several rules regarding the whole inductive structure of φ. Since quantitative parity games (or model checking [HK97]) as for the probabilistic μ-calculi [Mio12; FGK10; MM07] also closely follow the inductive structure of the formula, it is an interesting question whether the conceptual simplicity of our game can lead to a new perspective on existing games. Quantitative logics are already mentioned in Chapter 5 and also play a significant role in model checking based on "non-binary" truth values (see, e.g., [KL07; EKN12; GL+05; Fit91]), which is considered in our paper [BK+19a], but not in this thesis.

Furthermore, in [BK+19a] we present techniques to compute the generalized version of a small progress measure which is introduced by Jurdziński' [Jur00] and studied in a coalgebraic context in [HSC16]. Our results are based on the fact that the set of progress measures forms a lattice since $[\alpha]^m_\star = \{\beta \mid \beta \leq \alpha\}^m \cup \{\star\}$ is a lattice too. This allows to formulate progress measure equations which leads to a monotone function over progress measures and therefore enables the computation of a complete progress measure via smallest fixpoint computations. Afterwards, given a system of fixpoint equations E we introduce so-called *selections* which provide for each element b of the basis and a function f_i a subset of the moves $\mathbf{E}(b, i)$ that are sufficient to *cover* b in all possible ways. Build up on the observation that selections describe a disjunctive normal form, we introduce a logical characterization of the

existential player's moves and study different ideas how progress measures can be (effectively) computed working also in the scope of infinite lattices [BK+19a; MS17; GS11].

The work in [BKP20] of my coauthors [BK+19a] extends our results in two different dimensions. First of all, they get rid of the restriction to continuous lattices via the observation that an equational system over any complete lattice L can be encoded via a corresponding system over a lattice $\mathcal{P}(B_L)$ by means of a Galois insertion. Secondly, this observation leads to algorithmic developments including up-to techniques inspired by the work in [BG+18; Hir98; PS11]. More precisely, given a monotone function $f : L \to L$ they consider a *compatible* function $u : L \to L$ i.e. $u \circ f \sqsubseteq f \circ u$ with the property:

$$l \sqsubseteq f(u(l)) \Rightarrow l \sqsubseteq \nu f$$

Given a system E of m fixpoint equations, the integration of compatible up-to functions as abstractions enables the simplification of the computations.

Next, a *local* algorithm is presented which addresses questions whether two states are bisimilar. The aim of such an algorithm is to compute the information which is needed to correctly verify such statements. The *on-the-fly* algorithm combines backtracking techniques with pruning procedures and yields a generalization of the computation presented in [SS98] according to fixpoint games. Furthermore, they combine their algorithm with up-to techniques inspired by [Hir99] to improve the computation in practice.

On the one hand, they generalize the results of [BK+19a] partially presented in this chapter to complete lattices and integrate ideas of abstraction theory [CC77], but on the other hand the construction $\mathcal{P}(B_L)$ increases the domain of interest and the games might become larger (i.e. more moves are available).

For solving fixpoint equations over infinite lattices (e.g. the real numbers), it still remains open how to compute or approximate an exact solution. However the use of up-to techniques seems to be a promising approach since the techniques described in [BKP20] are sound, when their overapproximations are not guaranteed to be close enough for non-continuous functions.

Another open question is given by the observation, that a parity game over a finite graph can be easily converted into a system of boolean equations whose solution characterizes the winning positions for the players. Since our game is a standard parity game, possibly played on an infinite graph, the standard conversion would lead to infinitely many equations. Systems of equations of this kind are considered,

e.g., in [Mad97]. An interesting question is under which conditions an infinite parity game can be converted into finitely many equations on an (infinite) powerset lattice.

8 Conclusion and Future Work

The primary objective of this thesis was to complete the coalgebraic framework for state-based systems with logical and game-theoretical approaches. To summarize this thesis we provide an overview of the contributions and start with the four research areas extended via the work presented in this thesis:

▷ Coalgebraic modal logic and games over **Set** (Chapter 3, Chapter 4 and Appendix A.1)

▷ Games and logic for behavioural distances over **Set** (Chapter 5)

▷ Coalgebraic modal logic and games beyond **Set** (Chapter 6)

▷ Parity Games (Chapter 7)

For each item above, we start with a discussion of the main contributions and finally close each topic with a brief outlook. Note, that for each topic a detailed conclusion can be found in the corresponding chapter(s).

Coalgebraic Modal Logic and Games over Set: Specifications are given in terms of logical formulas and in many scenarios several (probabilistic) aspects such as resources, concurrency, or security have to be considered at the same time [CK+11]. Therefore, the composition of different types of branchings and their corresponding modalities is of crucial interest where *coalgebraic modal logic* does not only provide a generalization of modal logic in terms of soundness and expressiveness, the theory of coalgebras also offers compositionality results (i.e. for branchings derived from the combination of functors) [CK+11]. Coalgebraic modal logic is a generalization where modalities are expressed via *predicate liftings* [HJ98; Jac01; Pat04]. In a nutshell, *predicate liftings* are converters, which transform a subset of X into a subset of FX where F specifies the branching type of a transition system.

In Chapter 3 we present two game-theoretical characterizations in terms of *predicate liftings* where we extend the results in [Sch08]. We connect the existence of a expressive set of *monotone* modalities with the anti-symmetry of the lifted order $\leq^F$ with $\leq = \{(0,0),(0,1),(1,1)\}$ (see Proposition 3.2.17). To consider $\leq$ is quite natural in a qualitative setting: either a state satisfies a modal formula or it does not. Given a functor F, which can be a composition of several functors, due to

Proposition 3.2.17 to get a sound and complete game, one only needs to verify if $\leq^F$ is an anti-symmetric preorder and $(Fp\colon FX \to F2)_{p\colon X\to 2}$ is jointly injective.

Inspired by Cleaveland [Cle90], the game-based approach based on $\leq^F$ provides general techniques to derive a distinguishing logical formula of two non-bisimilar states automatically. This approach relies on so-called *cone modalities* (see Definition 3.4.11) which are modalities purely based on $\leq^F$. The usage of *cone modalities* avoids to iterate over infinitely many modalities as given in the probabilistic settings. Besides, the *meanings* of *cone modalities* can be transformed to those of the standard modal operators (e.g. necessarily, everywhere, or probably) which is described in Section 3.4.4, where we provide some encoding techniques into more familiar modal logics. Therefore, *cone modalities* also offer an automatism that does not require the user to specify a set of modal operators in advance.

Moreover, in Chapter 4 we present a prototype implementation of the algorithms presented in Chapter 3 named T-Beg. A huge advantage of the coalgebraic framework is that generic algorithms enable a high degree of abstraction: a part of the architecture of T-Beg was already predefined by the coalgebraic approach before we even started to think of a suitable design. More precisely, Algorithm 3.1 is encapsulated in a separate class implied by the parameterization in terms of the functor and a concrete transition system.

In future we would like to adapt our algorithms in two dimensions. First, we want to improve the polynomial runtime behaviour and slacken the requirements via the techniques presented in [PT87; DM+17]. Second, we believe that our *bisimulation games* can be extended to polyadic predicate liftings [Sch08].

Behavioural Distances over Set: In practice, notions of behavioural equivalence are rather too strict since implementations do not always fully match the requirements and one needs to move to a quantitative setting [DLT08; AFS09].

Therefore, *pseudometrics* and *non-expansive* functions enter the scene, where *pseudometrics* over the state space X (i.e. $d : X \times X \to [0, 1]$) correspond to behavioural distances and *non-expansive* functions are mappings between metric spaces which do not increase the distance. The work in [BB+14] presents a coalgebraic treatment of the material based on the *Kantorovich* and *Wasserstein* liftings, which are standard concepts in terms of probabilistic transition systems originating from transportation theory [BW06; Vil09].

Inspired by the work in [BB+14] we improved the Kantorovich lifting via a parametrization based on a set Λ of evaluation maps. Moreover, we present a real-valued coalgebraic modal logic and give a Hennessy-Milner theorem for the

coalgebraic setting. Furthermore, we also provide a quantitative game in terms of real-valued evaluation maps and present a quantitative version of the classical triad mentioned also in the introduction (see Figure 1.3). Compared to the qualitative setting, monotonicity is replace by *local non-expansiveness* which is already mentioned in [TR98].

Further, behavioural distances could be studied in terms of the *Wasserstein* lifting which is used to set up a game for probabilistic systems as presented in [WS+18b] and also serves as an upper bound for the distance based on the *Kantorovich* lifting [BB+14; Ker16]. In analogy to the results presented in Section 3.4, efficient computation techniques (see, e.g. [CBW12; BB+19]) would enable the generation of spoiler strategies.

In order to lift the level of abstraction – moving to settings beyond **Set** – an approach based on *fibrations* is presented in [KK+19] which generalizes our games but does not consider modal logics. Therefore, the authors in [KR20] consider modal logics based on *fibrations* and *dual adjunctions* which allows to handle **Set** as well as **Meas**. However, we address this topic in Chapter 6.

Coalgebraic Modal Logic beyond Set: The coalgebraic notion of behavioural equivalence (see Definition 2.3.17) is a quite general notion since it also works beyond the standard category **Set**. In Chapter 6 we address the non-trivial question if we can setup logical and game-theoretical characterizations which capture different notions of coalgebraic behavioural equivalence, e.g. bisimulation as well as language equivalence.

Inspired by the work in [PT99; HJS07; JSS15] we move to *Kleisli* categories where for example for non-deterministic automata coalgebraic behavioural equivalence coincides with language equivalence. This phenomenon is achieved since the side-effect (i.e. the non-determinism) is hidden to an outside observer.

We work with *indexed categories* where Jacobs noticed that *predicate liftings* are nothing but indexed morphisms. We extend the ideas based on *indexed categories* [Jac10] to a *Kleisli extension* of predicate liftings which to our knowledge is new. Furthermore, we generalize the logical notion and game version presented in Section 3.3.1 so that we can capture trace equivalence as well as bisimulation.

Summarizing, we provide a different contribution compared to the games obtained for a fibrational framework [KK+19] which do not address trace semantics or modal logics but put games for bisimulation and behavioural metrics under one roof.

For categories beyond **Set** there are still some open questions how coalgebraic behaviour equivalence can be characterized via logical or game-theoretical semantics.

But all the frameworks (i.e. indexed categories, dual adjunctions and fibrations) contribute in answering this *non-trivial* question [KK+19; KR20].

Parity Games: The motivation to study parity games is given by the fact that any modal μ-calculus model checking problem can be reduced to the question whether the existential player $\exists$ has a winning strategy for the game derived from the formula [Sti95; EJ91]. Inspired by the work in [HSC16] we present a *simple* parity game based on the equational representation of a μ-calculus formula. The winner of a game is determined by the highest index occurring infinitely often and in case this index is a ν-index, the existential player $\exists$ is winning. Otherwise, the universal player $\forall$ is the winner. Furthermore, we generalize such a game-theoretical approach to *continuous lattices*, which are anchored in the origin of *domain theory* introduced by Scott [Sco72; GH+03]. Continuous lattices are defined based on the so-called *way-below* relation, which provides the right notion to generalize the property of any element in a powerset lattice. More precisely, each subset is determined by the elements that belong to it and a generalization of this property enables the approximation in other settings, for instance for fixpoint equation systems over the real interval $[0, 1]$ (see [MS17]).

Only briefly addressed in this thesis, in our general framework a theory of *progress measures* in analogy to Jurdzisńki's work is already established in [BK+19a], where *progress measures* serve as witnesses for winning strategies [Jur00; HSC16].

Summarizing the results presented in this thesis, our generalization provides a new basis to study further approaches in terms of computation techniques for *progress measures*. Several ideas are included in [BK+18] where one is based on the incorporation of SMT solvers.

An extension of our work is presented in [BKP20], which mainly considers *up-to* techniques and *local* algorithms which combines backtracking and pruning techniques.

Another interesting question is given by *quantitative* parity games [Mio12; FGK10]: how can the game presented in this thesis and the work in [BKP20] lead to a new perspective on such games?

A Additional Material

The results in Section A.1 and Section A.2 are inspired by the reviews of Calco Early-Ideas 2017 [KM17b], CMCS 2020 [KMS20b] and the joint work with Lutz Schröder.

A.1 Games for Non-Weak Pullback preserving Functors

We consider games for neighbourhood frames where the requirement of a weak pullback preserving functor as introduced for the game in Chapter 3 is not satisfied by the functor which characterizes the *neighbourhood* branching, where such models originate from [MON70; Sco70].

Motivation for Neighbourhood Frames

For normal logic the modal operator $\Box$ satisfies the following axiom

$$\Box(\varphi_1 \wedge \varphi_2) \leftrightarrow \Box\varphi_1 \wedge \Box\varphi_1$$

which is valid in any Kripke model. Concerning a modal operator of non-normal modal logic the axiom $\Box\varphi_1 \wedge \Box\varphi_1 \rightarrow \Box(\varphi_1 \wedge \varphi_2)$ might not be valid any more [HK04]. For such scenarios neighbourhood structures and semantics enter the scene [Che80; Pac17] and monotonic neighborhood structures provide the right models if the properties of the modal operators are reduced to monotonicity [HK04; Han03]:

$$\varphi_1 \rightarrow \varphi_2 \text{ implies } \Box\varphi_1 \rightarrow \Box\varphi_2$$

Moreover, non-normal (monotone) modal logics are generalized by the so-called *coalition* logic which is used in *Social Choice theory* to analyze selection procedures with the goal to model the ability of a group of agents [Pau01]. The semantics of such logics are given by *effectivity* functions $\mathcal{P}N \rightarrow \mathcal{P}\mathcal{P}S$ [MP82] and by *strategic games* where S represents possible options (states) depending on the setting and N denotes the agents involved in the social choice process. An effectivity function $e : \mathcal{P}N \rightarrow \mathcal{P}\mathcal{P}S$ is given by any function e which is *outcome monotonic* [Pau01]:

$$\text{For all } C \subseteq N, X \subseteq Y \subseteq S : X \in e(C) \Rightarrow Y \in e(C)$$

An effectivity function assigns to each group of agents (i.e. $C \subseteq N$) a neighbourhood function on S and $e(C)$ are the neighbourhoods or outcomes for which C (i.e. the group/agent) is *effective* [Han03].

A strategic game form $G = (N, \{\Sigma_i \mid i \in N\}, o, S)$ contains non-empty sets of strategies or actions Σ_i for every player $i \in N$ and an outcome function $o : \prod_{i \in N} \Sigma_i \to S$ which maps each tuple of player strategies to a state $s \in S$. Intuitively, in a strategic game each player decides for one of his possible strategies and all decisions together produce the outcome of the game [Pau01; Han03].

In addition, such effectivity functions needs to be playable, which means that a function has a strategic game form (the transformation of a strategic game form into an effectivity function is described in [Pau01; Han03]). All playable functions are given by the following structure:

$$S \to (\mathcal{P}N \to \mathcal{P}\mathcal{P}S)$$

Therefore, a *coalation* model is based on several *monotonic* neighbourhood frames $E_C : S \to \mathcal{P}\mathcal{P}S$ each for every subset of agents (i.e. $C \subseteq N$)[Pau01]. Since a notion of bisimulation is used to answer the question if two models (states) can be viewed as *equal* [Pau01] we explain in the next section that our game in Definition 3.3.1 characterizes this notion of bisimulation, although the underlying functor is not weak pullback preserving.

A Game for Neighbourhood Frames

Based on the work in [BK11] we require in Lemma 3.2.7 that a functor preserves weak pullbacks to ensure that the lifted order of a partial order $\leq$ and $\pi_i \colon \leq \to 2$ the usual projections with $i \in \{1, 2\}$

$$\leq^F = \{(F\pi_1(t), F\pi_2(t)) \mid t \in F \leq\}$$

is transitive. But since we are only interested in lifting a very specific order

$$\leq = \{(0,0), (0,1), (1,1)\}$$

the question arises if transitivity holds without the assumption of weak-pullback preservation. Especially, since the footnote of Proposition 13 in [BK11] leaves this point open.

Next, we show that the monotone neighbourhood functor $\mathcal{M}$ (see Example 3.4.6) serves as an example for such a functor, where the lifted order is transitive and anti-symmetric. Moreover, this enables to prove the correctness and soundness of our

game for monotone neighbourhood frames without the assumption of weak pullback preservation.

First, we start with the coalgebraic characterization of monotone neighbourhood frames (mNHF) and models (mNHM) (see [HK04]).

Definition A.1.1: Monotone Neighbourhood Functor [HK04]

The monotone neighbourhood functor $\mathcal{M}$ is defined based on the composition of the contravariant powerset functor $\mathcal{Q}$ with itself. For this, we have for any set X and any function $f : X \to Y$ [DM+18]:

$$\begin{aligned} \mathcal{M}X &= \{X' \in \mathcal{Q}\mathcal{Q}X \mid X' \text{ is upward closed under } \subseteq\} \\ \mathcal{M}f(X') &= \{Y' \subseteq Y \mid f^{-1}[Y'] \in X'\} \qquad \text{for all } X' \subseteq \mathcal{Q}X. \end{aligned}$$

A monotone neighbourhood frame (mNHF) is given by a pair (X, α) where $\alpha : X \to \mathcal{M}X$.
A monotone neighbourhood model (mNHM) is given by a pair (X, α) where $\alpha : X \to \mathcal{M}X \times \mathcal{P}P$ and P is a set of propositions. An mNHM is based on an mNHF equipped with a valuation function $v : X \to \mathcal{P}P$, captured by the extension of the functor $\mathcal{M}$ to $\mathcal{M} \times \mathcal{P}P$.

The proof of Theorem 3.3.2 requires that F has a separating set of monotone predicate liftings and preserves weak pullbacks, which guarantees that $\leq^F$ is transitive [BK11].

In [DM+18] the authors already present a monotone unary and separating predicate lifting for the monotone neighbourhood functor:

$$\lambda_X(X') = \{U \in \mathcal{M}X \mid X' \in U\}$$

Remember that the set of separating sets is monotone iff the lifted order is anti-symmetric and $(Fp\colon FX \to F2)_{p\colon X \to 2}$ is jointly injective (see Proposition 3.2.17). Therefore, to show that our game with one coalgebra presented in Definition 3.3.1 is sound and complete for mNHF, it suffices to show that the lifted order $\leq^{\mathcal{M}}$ is again a partial order (reflexive, transitive and also anti-symmetric). Thus, it is left to show that $\leq^{\mathcal{M}}$ is transitive.

Regarding the definition of $\leq^{\mathcal{M}}$ we consider the lifting of the usual projections $\pi_i :\leq \to \{0, 1\}$ with $i \in \{1, 2\}$. Here, we want to emphasize, that $\mathcal{M} \leq$ contains only 20 elements, since all the other $(2)^{2^3} - 20$ elements in $\mathcal{Q}\mathcal{Q} \leq$ are not upward closed. Note, that except the upward-closed element $\emptyset$ all other elements include $\leq$. Thus,

already 127 subsets of $\mathcal{P} \leq$ can not be upward closed (see Figure A.1).

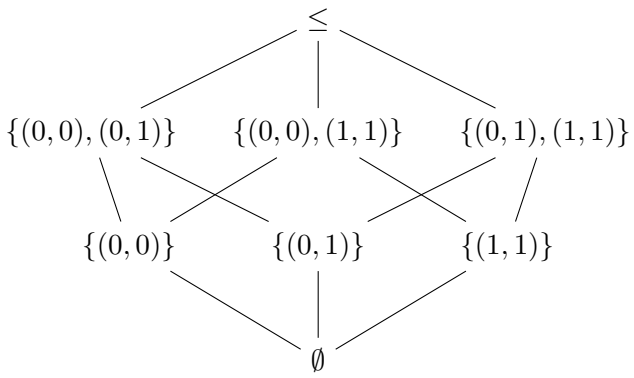


Figure A.1: There are 20 upward-closed subsets of $\mathcal{M} \leq$.

Next, the definition of $\leq^{\mathcal{M}} \subseteq \mathcal{M}2 \times \mathcal{M}2$ is based on the elements as follows:

$$\leq^{\mathcal{M}} = \{(\mathcal{M}\pi_1(U), \mathcal{M}\pi_2(U)) \mid U \in \mathcal{M} \leq\}$$

where $\mathcal{M}\pi_i(U) : \mathcal{M} \leq \to \mathcal{M}\{0,1\}$ is defined as follows:

$$\mathcal{M}\pi_i(U) = \{X \subseteq \{0,1\} \mid \pi_i^{-1}[X] \in U\} \text{ [MV12]}$$

with

$$\begin{aligned}\mathcal{M}\{0,1\} =& \{\emptyset, \{\emptyset, \{0\}, \{1\}, \{0,1\}\}, \{\{0\}, \{0,1\}\},\\ & \{\{1\}, \{0,1\}\}, \{\{0\}, \{1\}, \{(0,1)\}\}, \{\{0,1\}\}\}\end{aligned}$$

In addition, one can easily define the preimages of the projections $\pi_i : \leq \to \{0,1\}$, since the powerset of $\{0,1\}$ contains only 4 elements.

$\pi_1^{-1}[\emptyset] = \emptyset$	$\pi_2^{-1}[\emptyset] = \emptyset$
$\pi_1^{-1}[\{0\}] = \{(0,0),(0,1)\}$	$\pi_2^{-1}[\{0\}] = \{(0,0)\}$
$\pi_1^{-1}[\{1\}] = \{(1,1)\}$	$\pi_2^{-1}[\{1\}] = \{(0,1),(1,1)\}$
$\pi_1^{-1}[\{0,1\}] = \{(0,0),(0,1),(1,1)\}$	$\pi_2^{-1}[\{0,1\}] = \{(0,0),(0,1),(1,1)\}$

Given that one can easily compute for each $U \in \mathcal{M} \leq$ the resulting tuple. Based on all those computations one can verify that the lifted order is transitive and anti-symmetric. We only show a few cases and summarize the result in Figure A.2:

1. $U = \{\{(0,0),(0,1),(1,1)\}\}$:

$$\begin{aligned} & & \mathcal{M}\pi_1(U) = & & \{\{0,1\}\} \\ & \leq^{\mathcal{M}} & \mathcal{M}\pi_2(U) = & & \{\{0,1\}\} \end{aligned}$$

2. $U = \{\{(0,1),(1,1)\},\{(0,0),(0,1),(1,1)\}\}$:

$$\begin{array}{lll} & \mathcal{M}\pi_1(U) = & \{\{0,1\}\} \\ \leq^{\mathcal{M}} & \mathcal{M}\pi_2(U) = & \{\{1\},\{0,1\}\} \end{array}$$

3. $U = \{\{(0,0),(0,1)\},\{(0,0),(1,1)\},\{(0,0),(0,1),(1,1)\}\}$:

$$\begin{array}{lll} & \mathcal{M}\pi_1(U) = & \{\{0\},\{0,1\}\} \\ \leq^{\mathcal{M}} & \mathcal{M}\pi_2(U) = & \{\{0,1\}\} \end{array}$$

4. $U = \{\{(0,0)\},\{(0,1)\},\{(0,0),(0,1)\},\{(0,0),(1,1)\},\{(0,1),(1,1)\},$ $\{(0,0),(0,1),(1,1)\}\}$:

$$\begin{array}{lll} & \mathcal{M}\pi_1(U) = & \{\{0\},\{0,1\}\} \\ \leq^{\mathcal{M}} & \mathcal{M}\pi_2(U) = & \{\{0\},\{1\},\{0,1\}\} \end{array}$$

5. $U = \{\emptyset,\{(0,0)\},\{(0,1)\},\{(1,1)\},\{(0,0),(0,1)\},\{(0,1),(1,1)\},$ $\{(0,0),(1,1)\},\{(0,0),(0,1),(1,1)\}\}$:

$$\begin{array}{lll} & \mathcal{M}\pi_1(U) = & \{\emptyset,\{0\},\{1\},\{0,1\}\} \\ \leq^{\mathcal{M}} & \mathcal{M}\pi_2(U) = & \{\emptyset,\{0\},\{1\},\{0,1\}\} \end{array}$$

Finally, we derive the following partial order $\leq^{\mathcal{M}}$ illustrated in Figure A.2 and conclude that the lifted order satisfies the requirements necessary for Theorem 3.3.2. Therefore – and because $\mathcal{M}$ has a unary separating predicate lifting – our game in Definition 3.3.1 is suitable for mNHF.

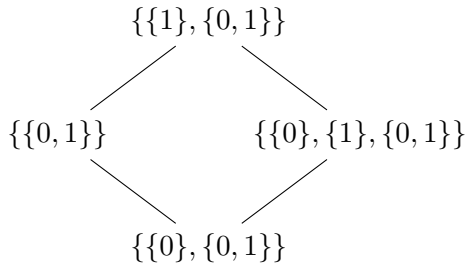


Figure A.2: The lifted order $\leq^{\mathcal{M}}$ is transitive, reflexive and anti-symmetric (i.e. a partial order).

A.2 Our Coalgebraic Game over Two Coalgebras

We present a variant of our game where the setting is given by two coalgebras with disjoint state spaces.

We noticed some problem while playing the game in Definition 3.3.1 over two mNHF frames, which also holds for functors which preserve weak pullbacks.

The problem occurs in Step 3 of the game (see Definition 3.3.1) where the spoiler can choose between p_1, p_2. In case p_1 includes some state, such that no behaviourally equivalent state is present in the state space of the other coalgebra, the duplicator has no opportunity to include a corresponding state into p_2. The states given in the initial situation can be behaviourally equivalent, but the duplicator has no winning-strategy, since the spoiler can choose p_1 and the state, where no behaviourally equivalent state is present in p_2 and therefore the duplicator will lose at Step 4 of the game.

Before we start to explain this problem on a concrete example, we introduce the notion of monotone bisimulation, which coincides with coalgebraic behavioural equivalence for mNHF [HK04]:

Definition A.2.1: Monotone Bisimulation [HK04]

Let $(X_1, \alpha_1), (X_2, \alpha_2)$ be two mNHF. A non-empty relation $Z \subseteq X_1 \times X_2$ is a *monotone bisimulation* between (X_1, α_1) and (X_2, α_2) if for all $x_1 \in X_1$ and $x_2 \in X_2$ such that $(x_1, x_2) \in Z$, the following two conditions are satisfied:

$$\textbf{forth} : \forall C_1 \in \alpha_1(x_1).\exists C_2 \in \alpha_2(x_2) \quad \text{s.t. } (\forall c_2 \in C_2.\exists c_1 \in C_1 : (c_1, c_2) \in Z).$$
$$\textbf{back} : \forall C_2 \in \alpha_2(x_2).\exists C_1 \in \alpha_1(x_1) \quad \text{s.t. } (\forall c_1 \in C_1.\exists c_2 \in C_2 : (c_1, c_2) \in Z).$$

Now, the problem becomes clear for the initial pair (s_1, s_2) of Example A.2.2, where two mNHF frames are given [HK04]: if the spoiler chooses $s_1, p_1 = \{t_1, u_1\}$ then p_2 can not include a state which is monotone bisimilar to u_1. This results in a winning-strategy for the spoiler although s_1 is monotone bisimilar to s_2 [HK04].

Example A.2.2: [HK04]

Let $X = \{s_1, t_1, u_1, v_1\}, Y = \{s_2, t_2\}$ and two coalgebras $\alpha_1 : X \to \mathcal{M}X$, $\alpha_2 : Y \to \mathcal{M}Y$ defined as follows be given:

$$\alpha_1(s_1) = \uparrow \{\{t_1\}, \{u_1, v_1\}\} \qquad \alpha_2(s_2) = \uparrow \{\{t_2\}\}$$
$$\alpha_1(t_1) = \uparrow \emptyset \qquad \alpha_2(t_2) = \uparrow \emptyset$$
$$\alpha_1(u_1) = \uparrow \{\{u_1\}\}$$
$$\alpha_1(v_1) = \uparrow \emptyset$$

The relation $Z = \{(s_1, s_2), (t_1, t_2), (v_1, t_2)\}$ is a monotone bisimulation [HK04].

As previously mentioned, this problem also occurs if we play the game over two F-coalgebras where F preserves weak pullbacks and therefore we slightly adapt Definition 3.3.1 of the game. This variation has been proposed by Lutz Schröder to simplify our game during the work presented in Section 3.3.2. Here we show that this idea solves our problem previously described. The difference is that we play over two coalgebras and **S** has to work with p_2 in Step 3, where in Definition 3.3.1 **S** can freely choose between p_1 and p_2.

Definition A.2.3: Game over two F-coalgebras

Let two coalgebras $\alpha_1 : X \to FX$, $\alpha_2 : Y \to FY$ with two non-empty and disjoint sets X, Y be given, where F has a separating set of predicate liftings and the lifted order $\leq^F$ is a partial order:

- **Initial situation:** $(x, y) \in X \times Y$.
- **Step 1:** **S** chooses $s \in \{x, y\}$ and a predicate $p_1 \colon X \to 2$ if $s = x$ ($p_1 \colon Y \to 2$ if $s = y$).
- **Step 2:** **D** takes $t \in \{x, y\} \backslash \{s\}$ and has to answer with a predicate $p_2 \colon Y \to 2$ if $s = x$ ($p_2 \colon X \to 2$ if $s = y$) satisfying

$$Fp_1(\alpha(s)) \leq^F Fp_2(\alpha(t))$$

- **Step 3:** **S** proceeds with p_2 and chooses some state $x' \in X$ ($x' \in Y$) with $p_2(x') = 1$.
- **Step 4:** **D** chooses some state $y' \in X$ ($x' \in X$) with $p_1(y') = 1$.

The game continues with (x', y').

The winning conditions stay the same but we repeat it quickly: Duplicator wins if the

game runs for ever or spoiler has no choice at Step 3. Spoiler wins if the duplicator has no option at Step 2 or Step 4.

Theorem A.2.4

If $\leq^F$ is a partial order and F has a separating set of predicate liftings, then $x \sim y$ iff $\mathbf{D}$ has a winning strategy for the initial situation (x, y).

Similar to the proof of Theorem 3.3.2 this proof splits into three parts. On the one hand we need to show that the relation

$$\mathcal{W}' = \{(u, v) \in (X \times Y) \cup (Y \times X) \mid \mathbf{D} \text{ has a winning strategy for } (u, v)\}$$

is symmetric. Note, that here we do not require the relation to be reflexive or transitive, since at the initial situation we only consider state pairs of type $(x, y) \in X \times Y$ where X and Y are disjoint sets.

But for the third part of the proof, we need to derive an equivalence relation $\mathcal{W} \subseteq (X \cup Y) \times (X \cup Y)$ from $\mathcal{W}'$ i.e. the reflexive and transitive closure of $\mathcal{W}'$. Therefore, we have to show, that in case $(x_1, y), (y, x_2) \in \mathcal{W}'$ then $\mathbf{D}$ has a winning-strategy for the initial situation $(x_1, x_2) \in X \times Y$ of a game where X, α_1 and Y, α_2 are both initialized with α_1 and $Y := X$ (or with Y, α_2 if $x_1, x_2 \in Y$). Note, that in this game version, a special case is given by $\alpha_1 = \alpha_2$ since the players have to play with the predicates of their underlying system α_1 or α_2.

(Alternatively, in such a case one can restrict to α_1 and play the game only over one system, but then we need to adapt Step 2 to enable games over reflexive pairs. This yields the game version described in Definition 3.3.1 with a small modification: the move of the Spoiler in Step 3 is restricted to the predicate p_2. The proofs are analogous to the proofs of Theorem 3.3.2 where the case $\mathbf{S}$ *chooses* p_1 in Step 3 can be omitted.)

Summarizing, the goal is to add the reflexive and transitive pairs based on $\mathcal{W}'$ to $\mathcal{W}$ where we need to show that $\mathbf{D}$ has a winning strategy for all these pairs.

Secondly, a part of the proof is to establish a winning strategy for $\mathbf{D}$ whenever $x \sim y$ (soundness).

The last part is to construct two witnesses (i.e. coalgebra homomorphisms):

$$f : X \to (X \cup Y)/\mathcal{W}, g : Y \to (X \cup Y)/\mathcal{W}$$

with $f(x') = g(y')$ for each $(x', y') \in \mathcal{W}$ which implies that $f(x) = g(y)$ holds for each $(x, y) \in \mathcal{W}'$. And finally, we show that this way we obtain a coalgebra $\beta : (X \cup Y)/\mathcal{W} \to F((X \cup Y)/\mathcal{W})$ and hence $x \sim y$ (correctness).

Lemma A.2.5

Given two coalgebras $\alpha_1 \colon X \to FX$, $\alpha_2 \colon Y \to FY$ and $\leq^F$ is a partial order. Then

$$\mathcal{W}' = \{(u, v) \in X \times Y \text{ or } Y \times X \mid \mathbf{D} \text{ has a winning strategy for } (u, v)\}$$

is a symmetric relation.
For any pair (u, u) where u is either in X or Y, $\mathbf{D}$ has a winning strategy for the initial situation (u, u) with $\alpha_2 := \alpha_1$ if $u \in X$ (or $\alpha_1 := \alpha_2$ if $u \in Y$).
In addition, if $(x, y), (y, z) \in \mathcal{W}'$ then $\mathbf{D}$ has a winning-strategy for the initial situation (x, z) with $\alpha_2 := \alpha_1$ if $x, z \in X$ (or $\alpha_1 := \alpha_2$ if $x, z \in Y$).

Proof:

- $\mathbf{D}$ *has a winning strategy for any reflexive pair:* (u, u) where u is either in X or Y.

  Both parameters of the game are initialized with the same coalgebra and therefore $\mathbf{D}$ can copy any move of $\mathbf{S}$ and either the game never terminates or $\mathbf{S}$ has no option in Step 3.

- $\mathcal{W}'$ *is symmetric:* $(x, y) \in \mathcal{W}'$ implies $(y, x) \in \mathcal{W}'$.

  If there is a winning strategy for (x, y) there must always also be a winning strategy for (y, x), since $\mathbf{S}$ can choose either x or y.

- *Note* if $(x, y), (y, z) \in \mathcal{W}'$, then $\mathbf{D}$ has a winning-strategy for (x, z) with $\alpha_1 = \alpha_2$.

  Assume that in Step 1 $\mathbf{S}$ chooses x and p_1 (the case where $\mathbf{S}$ chooses y is analogous, taking into account that $\mathcal{W}'$ is symmetric). We know by $(x, y) \in \mathcal{W}'$ that $\mathbf{D}$ has an answer, hence he chooses p_2, for which $Fp_1(\alpha_1(x)) \leq^F Fp_2(\alpha_2(y))$.

  For $\mathbf{S}$ has to work with p_2 and y, we know by $(y, z) \in \mathcal{W}'$ that $\mathbf{D}$ has an answering move, by choosing p_3 such that $Fp_2(\alpha_2(y)) \leq^F Fp_3(\alpha_1(z))$.

  Hence $\mathbf{D}$ makes the choice of p_3 in Step 2. Now we have that

  $$Fp_1(\alpha_1(x)) \leq^F Fp_2(\alpha_2(y)) \leq^F Fp_3(\alpha_1(z))$$

  and, by transitivity of $\leq^F$, $Fp_1(\alpha_1(x)) \leq^F Fp_3(\alpha_1(z))$. Thus, in case the game is initialized two times with the same coalgebra we can derive from p_3 a suitable move for the duplicator since $Y := X$ (or $X := Y$).

(Here, transitivity holds by assumption since we have shown, that the lifted order $\leq^F$ can be transitive although the functor does not preserve weak pullbacks cf. Figure A.2).

Note, that in this game version for Step 3 **S** has to work with p_2 in case of (x, y) and with p_3 in case of (y, z):

Assume that in Step 3 **S** chooses p_3, z' with $p_3(z') = 1$. Again, by $(y, z) \in \mathcal{W}'$, there is an answer of **D** who chooses y' with $p_2(y') = 1$ and $(y', z') \in \mathcal{W}'$.

From $(x, y) \in \mathcal{W}'$ we know that if **S** chooses p_2, y' in Step 3, there is an answer by **D** who chooses x' with $p_1(x') = 1$ and $(x', y'), (y', x') \in \mathcal{W}'$. This state x' is hence finally chosen by **D** in Step 4.

Since we now have $(x', y'), (y', z') \in \mathcal{W}'$, we can continue this strategy for **D** forever.

In addition, if $(x, y), (y, x') \in \mathcal{W}'$ and $(x', y') \in \mathcal{W}'$ with $x, x' \in X$ and $y, y' \in Y$ one can derive similarly that $(x, y') \in \mathcal{W}'$. □

We proceed with the proof of Theorem A.2.4:

Proof: $x \sim y \Rightarrow$ **D** *has winning strategy:* We show that whenever $x \sim y$, then **D** can always answer the steps of **S** and we end up in a pair $x' \sim y'$, from which this strategy continues.

Whenever $x \sim y$, there exists a coalgebra $\beta \colon Z \to FZ$ and two coalgebra homomorphisms $f \colon X \to Z$, $g \colon Y \to Z$ such that $f(x) = g(y)$.

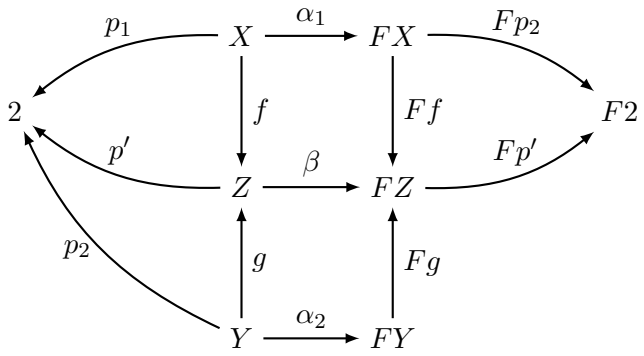
We assume that **S** chooses state x (the other case is analogous) and a predicate $p_1 \colon X \to 2$. **D** has now to react with a predicate p_2. This is constructed based on $p' \colon Z \to 2$ defined as follows:

1. $p'(z) = 1$ if there exists an x' with $p_1(x') = 1$ and $f(x') = z$.

2. $p_2 = p' \circ g$

Note that it does not matter if $p_1(x') = 1$ (which implies $p'(f(x')) = 1$) and no y exists such that $g(y) = z = f(x')$ since spoiler has to work with p_2 in Step 3.

In other words, $p_2(y) = 1$ for $y \in Y$ whenever there exists $x' \in X$ such that $g(y) = f(x')$ and $p_1(x') = 1$. Note, that analogous to Theorem 3.3.2 $p' \colon Z \to 2$ is the least predicate such that $p' \circ f \geq p_1$ (i.e., $p' \leq p$ for all p satisfying $p \circ f \geq p_1$).



Since $p_1 \leq p' \circ f$, we know by Lemma 3.2.8 that $Fp_1 \leq^F F(p' \circ f)$ holds. In addition, we have $p_2 = p' \circ g$.

Since f, g are homomorphisms, we obtain for $f(x) = g(y)$:

$$(Ff \circ \alpha_1)(x) = (\beta \circ f)(x) = (\beta \circ g)(y) = (Fg \circ \alpha_2)(y)$$

which implies

$$\begin{aligned} (Fp' \circ Ff \circ \alpha_1)(x) &= (Fp' \circ Fg \circ \alpha_2)(y) \\ \Rightarrow (Fp' \circ Ff \circ \alpha_1)(x) &= (Fp_2 \circ \alpha_2)(y) \\ \Rightarrow (Fp_1 \circ \alpha_1)(x) \leq^F (Fp' \circ Ff \circ \alpha_1)(x) &= (Fp_2 \circ \alpha_2)(y) \end{aligned}$$

By construction of p_2 there are two cases we have to distinguish:

1. p_2 is the constant 0-predicate, since $p_1(u) = 1$ holds only for states $u \in X$ ($u \in Y$), where no $y' \in Y (x' \in X)$ with $f(u) = g(y')$ ($f(x') = g(u)$) exists. **S** has no option at Step 3 of the game and thus **D** wins.

2. p_2 is not the constant 0-predicate and **S** now chooses a state y' with the constraints described in Step 3, i.e., $p_2(y') = 1 = p' \circ g(y')$ with $g(y') = z'$. By construction of p' this means that there must exist a x' such that $p_1(x') = 1$ with $f(x') = z'$. Therefore and by $p_2 = p' \circ g$, we have $f(x') = z' = g(y')$ and $p'(f(x')) = p'(g(y')) = 1$. Thus, **D** can choose this state x' with $f(x') = g(y')$ and $p_1(x') = 1$. In this case $x' \sim y'$ holds and the game can continue.

D *has a winning strategy* $\Rightarrow x \sim y$*:* We already know by Lemma A.2.5 that

$$\mathcal{W}' = \{(x, y) \in (X \times Y) \cup (Y \times X) \mid \textbf{D} \text{ has a winning strategy for } (x, y)\}$$

is symmetric and in case $(x, y), (y, z) \in \mathcal{W}'$ there also exists a winning-strategy for (x, z) where α_1 and α_2 are equal. Thus we obtain an equivalence relation $\mathcal{W}$ which

includes all the pairs, where the duplicator has a winning-strategy (no matter if we play on two different systems or over one system).

We define two functions $f\colon X \to Z$, $g\colon Y \to Z$ with $Z = (X \cup Y)/\mathcal{W}$ and $f(x) = [x]_{\mathcal{W}}$ and $[y]_{\mathcal{W}} = g(y)$.

Note, that we need the reflexive pairs to get a class represented by a state z in Z, if for a state $x \in X$ (Y) there exists no state in $y \in Y$ (X) such that $(x, y) \in \mathcal{W}'$. Such a state will be mapped to that z by f (respectively g).

It suffices to show that $\beta(g(y)) := Fg(\alpha_2(y))$ and $\beta(f(x)) := Ff(\alpha_1(x))$ with $\beta(f(x)) = \beta(g(y))$ are well defined, since then we have two coalgebra homomorphisms that witness the behavioural equivalence of x, y.

Assume that we have a winning strategy for (x, y) (i.e., $(x, y) \in \mathcal{W}'$) or in other words $f(x) = g(y)$, but $Ff(\alpha_1(x)) \neq Fg(\alpha_2(y))$. Then we know, by the assumption that the functor F has a separating set of predicate liftings (respectively the equivalent condition in [Sch08]), that some $p\colon Z \to 2$ exists such that $Fp(Ff(\alpha_1(x))) \neq Fp(Fg(\alpha_2(y)))$.

We now show, by contradiction, that **D** does not have a winning strategy for (x, y): **S** chooses $p_1 = p \circ f$ and we obtain

$$Fp_1 \circ \alpha(x) \neq F(p \circ g) \circ \alpha(y),$$

since $Fp_1 = Fp \circ Ff$. Since the preorder $\leq^F$ on $F2$ is antisymmetric due to Proposition 3.2.17 at least one of the following two overlapping cases will occur:

- $Fp_1 \circ \alpha_1(x) \not\leq^F F(p \circ g) \circ \alpha_2(y)$
- $F(p \circ g) \circ \alpha_2(y) \not\leq^F Fp_1 \circ \alpha_1(x)$

Here we only consider the first case, since for the second case the argument is analogous. **S** picks x and **D** can not play p_2 such that $p_2 \leq p \circ g$, since in this case we would get $Fp_2 \leq^F F(p \circ g)$ and $F(p \circ g)$ is by assumption not good enough for **D**: combining this with the condition of Step 2 we obtain

$$(Fp_1 \circ \alpha_1)(x) \leq^F (Fp_2 \circ \alpha_2)(y) \leq^F (F(p \circ g) \circ \alpha_2)(y)$$

which, with transitivity of $\leq^F$, is a contradiction to the first case above.

Hence $p_2 \not\leq p \circ g$, which implies that some $y' \in Y$ exists such that $p_2(y') = 1$ and $(p \circ g)(y') = 0$. So **S** picks p_2 and that y'. **D** then picks some $x' \in X$ with $p_1(x') = 1$. If $(x', y') \in \mathcal{W}$ (i.e. $f(x') = g(y')$) it follows from the construction of $p_1 = p \circ f$ that $1 = p_1(x') = p(f(x')) = p(g(y'))$ holds. But this is again a contradiction to $(p \circ g)(y') = 0$. Hence $(x', y') \notin \mathcal{W}$ and **D** does not have a winning strategy. □

We are now ready to play our game on the Example from [HK04].

Example A.2.6

This example demonstrates how to play our game over two monotone neighbourhood frames given in Example A.2.2.
The initial situation of the game is (s_1, s_2):

1. **S** chooses s_1 and a predicate $p_1 : X \to 2$ where $p_1(t_1) = p_1(u_1) = p_1(v_1) = 1$ and $p_1(s_1) = 0$.

2. **D** has to work with s_2 and defines a predicate $p_2 : Y \to 2$ where $p_2(t_2) = 1$ and $p_2(s_2) = 0$ and obtains:

$$\mathcal{M}p_1(\alpha_1(s_1)) = \{\{1\}, \{0,1\}\} \leq^{\mathcal{M}} \{\{1\}, \{0,1\}\} = \mathcal{M}p_2(\alpha_2(s_2))$$

3. **S** has to proceed with p_2 and chooses t_2 with $p_2(t_2) = 1$.

4. **D** chooses t_1 with $p_1(t_1) = 1$.

In the next round, **S** has to choose between t_1 and t_2 and we have $\alpha_1(t_1) = \emptyset = \alpha_1(t_2)$. No matter what **S** will do, **D** can answer with the constant 0-predicate in Step 2. Therefore, **S** will have no option at Step 3 and **D** wins the game.
Assume, that **S** plays in Step 1 with s_1 and $p_1(u_1) = 1$ and $p_1(x) = 0$ for all $x \in X \setminus \{u_1\}$. Since $\mathcal{M}p_1(\alpha_1(s_1)) = \{\{0\}, \{0,1\}\}$ **D** can answer with the constant 0-predicate p_0 in Step 2. We get $\mathcal{M}p_0(\alpha_2(s_2)) = \{\{0\}, \{0,1\}\}$ because $p_0^{-1}[\{0,1\}] = \{s_2, t_2\} \in \alpha(s_2)$ and $p_0^{-1}[\{0\}] = \{s_2, t_2\} \in \alpha(s_2)$ where

$$\mathcal{M}p_0(\alpha_2(s_2)) = \{X \subseteq \{0,1\} \mid p_0^{-1}[X] \in \uparrow \{\{t_2\}\}\}$$

Therefore, **S** will have no option at Step 3 and **D** again wins the game.

For a single mNHF one can also compute the winning strategies and construct distinguishing formulas for non-bisimilar state pairs. But without any further optimizations the runtime behaviour becomes exponential, since the functor $\mathcal{M}$ is not separable by singletons (see Example 3.4.6) and therefore, one needs to iterate over all possible unions of equivalence classes within the *for-loop* of Algorithm 3.1. In general for two coalgebras $(X, \alpha_1), (Y, \alpha_2)$, Algorithm 3.1 has to be adapted to the construction of $\mathcal{W}$.

Note, that for neighbourhood models one gets similar results based on a preorder lifting $\leq^F$ where each subset of propositions $\rho \subseteq P$ yields a transitive and

antisymmetric cone analogous to the cone in Figure A.2.

A.3 Additional Material for Chapter 3

The following proposition is inspired by the results presented in [Sch08] originating from [Pat04].

Proposition 3.2.19 [KM18]

The logic $\mathcal{L}^\kappa(\Lambda)$ is sound, that is given a coalgebra $\alpha\colon X \to FX$ and $x, y \in X$, $x \sim y$ implies that $[\![\varphi]\!]_\alpha(x) = [\![\varphi]\!]_\alpha(y)$ for all formulas φ.

Whenever F is κ-accessible and Λ is separating for F, the logic is also expressive: whenever $[\![\varphi]\!]_\alpha(x) = [\![\varphi]\!]_\alpha(y)$ for all formulas φ we have that $x \sim y$.

Proof:

Soundness: Assume that $x \sim y$, that is there exists a coalgebra morphism $f\colon X \to Y$ between α and a coalgebra $\beta\colon Y \to FY$ such that $\beta \circ f = Ff \circ \alpha$. We show that for all φ, $[\![\varphi]\!]_\alpha = [\![\varphi]\!]_\beta \circ f$ by structural induction over φ. The only interesting case is the modality ($\varphi = [ev]\psi$ with $ev \in \Lambda$) and here we obtain

$$\begin{aligned}[\![[ev]\psi]\!]_\alpha &= ev \circ F[\![\psi]\!]_\alpha \circ \alpha = ev \circ F([\![\psi]\!]_\beta \circ f) \circ \alpha \\ &= ev \circ F[\![\psi]\!]_\beta \circ Ff \circ \alpha = ev \circ F[\![\psi]\!]_\beta \circ \beta \circ f = [\![[ev]\psi]\!]_\beta \circ f\end{aligned}$$

where the second equality is due to the induction hypothesis. Now it follows easily that $[\![\varphi]\!]_\alpha(x) = [\![\varphi]\!]_\beta(f(x)) = [\![\varphi]\!]_\beta(f(y)) = [\![\varphi]\!]_\alpha(y)$.

Expressiveness: We define a logical indistinguishability relation on X:
$x \equiv y \iff \forall\varphi(x \models \varphi \iff y \models \varphi)$. Let $f\colon X \to X/\equiv$ be the function that maps every $x \in X$ to its equivalence class $[x]_\equiv$. In particular $f(x) = f(y)$.

Furthermore we define $\beta\colon (X/\equiv) \to F(X/\equiv)$ as $\beta([x]_\equiv) = Ff(\alpha(x))$. It is sufficient to show that β is well-defined, since then $\beta \circ f = Ff \circ \alpha$ and we have shown that $x \sim y$.

Hence we have to show that for $x, y \in X$ with $x \equiv y$ it always holds that $Ff(\alpha(x)) = Ff(\alpha(y))$. Since Λ is separating for F (cf. Definition 3.2.13) it is sufficient to show that

$$(ev \circ Fp \circ Ff \circ \alpha)(x) = (ev \circ Fp \circ Ff \circ \alpha)(y) \tag{A.1}$$

for all $ev \in \Lambda$ and all $p\colon (X/\equiv) \to 2$. Since F is κ-accessible there exists $Z \subseteq X$, $|Z| < \kappa$ such that $\alpha(x), \alpha(y) \in FZ \subseteq FX$.

Now, for a given $p\colon (X/\equiv) \to 2$ define $q\colon X \to 2$ with $q = p \circ f$. Given $z_0, z_1 \in Z$ with $q(z_0) = 0$, $q(z_1) = 1$, we know that $f(z_0) \neq f(z_1)$ and hence $z_0 \not\equiv z_1$. So there exists a formula φ_{z_0,z_1} that distinguishes z_0, z_1, that is $z_0 \not\models \varphi_{z_0,z_1}$ and $z_1 \models \varphi_{z_0,z_1}$ (this can be ensured since we have negation).

Now consider the following formula:

$$\varphi = \bigwedge_{\substack{z_0 \in Z \\ q(z_0)=0}} \Big(\bigvee_{\substack{z_1 \in Z \\ q(z_1)=1}} \varphi_{z_0,z_1} \Big)$$

Note that since $|Z| < \kappa$ the conjunction and disjunctions obey the cardinality restrictions. For every $z_1 \in Z$ with $q(z_1) = 1$ a formula φ_{z_0',z_1}, which is satisfied by z_1, occurs in every disjunction and hence $[\![\varphi]\!](z_1) = 1$. On the other hand for every $z_0 \in Z$ with $q(z_0) = 0$ there is a disjunction consisting only of formulas $\varphi_{z_0,z_1'}$, which are not satisfied by z_0, and hence $[\![\varphi]\!](z_0) = 0$. Summarizing, we obtain that $[\![\varphi]\!]$ and q agree on Z.

We will now proceed to show that (A.1) holds for a given ev and p: set $\psi = [ev]\varphi$ and we obtain

$$\begin{aligned} & (ev \circ Fp \circ Ff \circ \alpha)(x) = (ev \circ F(p \circ f) \circ \alpha)(x) = (ev \circ Fq \circ \alpha)(x) \\ = \;& (ev \circ F[\![\varphi]\!] \circ \alpha)(x) = [\![\psi]\!](x) \end{aligned}$$

Here we use the fact that whenever $t \in FZ$ and $h|_Z = g|_Z$ for $h, g\colon X \to 2$ we have $Fh(t) = Fg(t)$ (since $h \circ \iota = g \circ \iota$, where $\iota\colon Z \to X$ is the embedding of Z into X). In our case $t = \alpha(x)$.

Similarly we have $(ev \circ Fp \circ Ff \circ \alpha)(y) = [\![\psi]\!](y)$ and since x, y are logically indistinguishable (A.1) follows. □

The following Corollary A.3.1 refers to the construction of a distinguishing formula presented in Corollary 3.4.18. For the case $I(x, y) > 1$ the size of the conjunctions may be minimized working with $C \subset X \setminus P$ instead of $X \setminus P$ as described in Corollary 3.4.18. Remember P is the move derived from the winning strategy $T(x, y) = (s, P)$ of the spoiler.

In the scope of LTS optimizations are already taken into account by the work of Cleaveland since the recursive construction is based only on the a-successors of the underlying states x, y where $a \in A$ is splitting for (x, y) [Cle90]. In the general setting, one needs to remove all the states from $X \setminus P$, which we denote with P', such that $F\chi_{P \cup P'}(\alpha(t)) = F\chi_P(\alpha(t))$ for each $t \in \{x, y\}$.

Corollary A.3.1: A Simplified Construction

We use the construction of $\varphi_{x,y}$ as described in Corollary 3.4.18 with the only modification that for $i = I(x,y) > 1$ the formula φ is replaced by

$$\varphi' = \bigwedge_{y' \in C} \varphi_{x',y'}$$

for some $x' \in P$ and $C = X \setminus (P \cup P')$ such that $F\chi_{P \cup P'}(\alpha(y)) = F\chi_P(\alpha(y))$ and $F\chi_{P \cup P'}(\alpha(x)) = F\chi_P(\alpha(x))$ with P' maximal. Then this yields a formula $\varphi_{x,y}$ such that $x \vDash \varphi_{x,y}$ and $y \nvDash \varphi_{x,y}$.

Proof: Remember that we define $\varphi_{x,y} = [ev]\varphi'$ with $v = F\chi_P(\alpha(x))$, $ev = \uparrow v$ in case $T(x,y) = (x,P)$ (the other case works analogous). Furthermore, we know that $ev(F\chi_P(\alpha(y))) = 0$ because of $T(x,y) = (x,P)$ and ev is a cone modality.
We show $x \vDash \varphi_{x,y}$: By definition no state $y' \in C$ satisfies φ'. And by the proof of Proposition 3.4.15 we know that all $y' \in X \setminus P$ have been separated from $x' \in P$ in an iteration $j < I(x,y)$ and $y' \in C \Rightarrow y' \in X \setminus P$. Based on Lemma 3.4.16 we know that each $z \in P$ satisfies φ' since we have $x' \vDash \varphi'$ by construction and $z, x' \in P$ with $I(x',z) > j$.

Therefore, we have $P \subseteq [\![\varphi']\!] \subseteq P \cup P'$. Since $v = F\chi_P(\alpha(x)) = F\chi_{P \cup P'}(\alpha(x))$ we get by Lemma 3.2.8 $v \leq^F F\chi_{[\![\varphi']\!]}(\alpha(x)) \leq^F v$ which implies $v = F\chi_{[\![\varphi']\!]}(\alpha(x))$ (by antisymmetry of $\leq^F$). Finally, we derive

$$\uparrow v(F\chi_{[\![\varphi']\!]}(\alpha(x))) = 1$$

.
We conclude that $y \nvDash \varphi_{x,y}$: We have that $v = F\chi_{[\![\varphi']\!]}(\alpha(x))$ and $F\chi_{P \cup P'}(\alpha(y)) = F\chi_P(\alpha(y)) \not\geq^F v$ and thus we have

$$\uparrow v(F\chi_{[\![\varphi']\!]}(\alpha(y))) = 0$$

□

A.4 Proofs for Chapter 6

Lemma 6.2.6

The bifibration property is preserved by the composition of functors.

Proof: Suppose a given bifibration $\mathbf{D}^{\mathrm{op}} \xrightarrow{\Phi} \mathbf{Cat}$ and a functor $\mathbf{C} \xrightarrow{F} \mathbf{D}$, then we need to show that the composition $\Psi = \Phi \circ F$ is again a bifibration. Let

$C \xrightarrow{f} C' \in \mathbf{C}$. It is straightforward to show that $\Psi C \xrightarrow{\exists_{Ff}} \Psi C'$ with $\Phi F = \Psi$ is the left adjoint to the reindexing functor $(Ff)^*$. □

Lemma 6.3.7

For any $\mathbb{U} \subseteq \mathcal{M}_{\mathbb{F}}X$ we find that $\lambda^a_X(\mathbb{U}) = \{\bar{p} \in \mathcal{M}_{\mathbb{F}}(A \times X + 1) \mid \gamma_X^A \bar{p}(a) \in \mathbb{U}\}$ and $\lambda^s_X(\mathbb{U}) = \{\bar{p} \in \mathcal{M}_{\mathbb{F}}(A \times X + 1) \mid \bar{p}(\bullet) = s\}$.

Proof: For any $\mathbb{U} \subseteq \mathcal{M}_{\mathbb{F}}X$ we find that

$$\begin{aligned}
\lambda^a_X(\mathbb{U}) &= {\gamma_X}^{-1}\sigma^a_{\mathcal{M}_{\mathbb{F}}X}(\mathbb{U}) \\
&= {\gamma_X}^{-1}\{(p,s) \in (\mathcal{M}_{\mathbb{F}}X)^A \times \mathbb{F} \mid p(a) \in \mathbb{U}\} \\
&= \left\{\bar{p} \in \mathcal{M}_{\mathbb{F}}(A \times X + 1) \mid \gamma_X \bar{p} \in \{(p,s) \in (\mathcal{M}_{\mathbb{F}}X)^A \times \mathbb{F} \mid p(a) \in \mathbb{U}\}\right\} \\
&= \left\{\bar{p} \in \mathcal{M}_{\mathbb{F}}(A \times X + 1) \mid \gamma_X^A \bar{p}(a) \in \mathbb{U}\right\}.
\end{aligned}$$

Similarly, in the context of termination, we find (for each $s \in \mathbb{F}$):

$$\begin{aligned}
\lambda^s_X(\mathbb{U}) &= {\gamma_X}^{-1}\sigma^s_{\mathcal{M}_{\mathbb{F}}X}(\mathbb{U}) \\
&= {\gamma_X}^{-1}\{(p,s) \mid p \in (\mathcal{M}_{\mathbb{F}}X)^A\} \\
&= \left\{\bar{p} \in \mathcal{M}_{\mathbb{F}}(A \times X + 1) \mid \gamma_X \bar{p} \in \{(p,s) \mid p \in (\mathcal{M}_{\mathbb{F}}X)^A\}\right\} \\
&= \{\bar{p} \in \mathcal{M}_{\mathbb{F}}(A \times X + 1) \mid \bar{p}(\bullet) = s\}.
\end{aligned}$$

□

Lemma 6.3.8

For any $\mathbb{U} \subseteq \mathcal{M}_{\mathbb{F}}X$ and $X \xrightarrow{\alpha} \mathcal{M}_{\mathbb{F}}(A \times X + 1)$, we have $|\alpha|^{-1}\lambda^a_X(\mathbb{U}) = \{p \in \mathcal{M}_{\mathbb{F}}X \mid p \xrightarrow{a} \hat{p}(a) \implies \hat{p}(a) \in \mathbb{U}\}$ and $|\alpha|^{-1}\lambda^s_X(\mathbb{U}) = \{p \in \mathcal{M}_{\mathbb{F}}X \mid s = \sum_{x \in X}\ p(x) \cdot \alpha(x)(\bullet)\}$. Moreover, the set $\Lambda = \{\lambda^a \mid a \in A\} \cup \{\lambda^s \mid s \in \mathbb{F}\}$ is separating with respect to $A \times X + 1$.

Proof: Let $\mathbb{U} \subseteq \mathcal{M}_{\mathbb{F}}X$ and a given LWA $X \xrightarrow{\alpha} \mathcal{M}_{\mathbb{F}}(A \times X + 1)$. Then we derive

$$\begin{aligned}
|\alpha|^{-1}\lambda^a_X\mathbb{U} &= |\alpha|^{-1}\left\{\bar{p} \in \mathcal{M}_{\mathbb{F}}(A \times X + 1) \mid \gamma_X^A \bar{p}(a) \in \mathbb{U}\right\} \\
&= \left\{p \in \mathcal{M}_{\mathbb{F}}X \mid |\alpha|p \in \left\{\bar{p} \in \mathcal{D}(A \times X + 1) \mid \gamma_X^A \bar{p}(a) \in \mathbb{U}\right\}\right\} \\
&= \left\{p \in \mathcal{M}_{\mathbb{F}}X \mid \gamma_X^A(\mu_{A \times X + 1}\mathcal{M}_{\mathbb{F}}\alpha(p))a \in \mathbb{U}\right\} \\
&= \left\{p \in \mathcal{M}_{\mathbb{F}}X \mid p \xrightarrow{a} \hat{p}(a) \implies \hat{p}(a) \in \mathbb{U}\right\}.
\end{aligned}$$

Similarly, we have a modality to handle termination that can be derived as follows:

$$\begin{aligned}|\alpha|^{-1}\lambda^s_X\mathbb{U} &= |\alpha|^{-1}\{\bar{p} \in \mathcal{M}_{\mathbb{F}}(A \times X + 1) \mid \bar{p}(\bullet) = s\} \\ &= \{p \in \mathcal{M}_{\mathbb{F}}X \mid |\alpha|p \in \{\bar{p} \in \mathcal{M}_{\mathbb{F}}(A \times X + 1) \mid \bar{p}(\bullet) = s\}\} \\ &= \{p \in \mathcal{M}_{\mathbb{F}}X \mid s = \sum_{x \in X} p(x) \cdot \alpha(x)(\bullet)\}.\end{aligned}$$

Now it remains to show that Λ is separating w.r.t. $A \times X + 1$. So let $\bar{p}, \bar{p}' \in \mathcal{M}_{\mathbb{F}}(A \times X + 1)$ with $\bar{p} \neq \bar{p}'$. Then we identify the following two cases:

1. Let $\bar{p}(\bullet) \neq \bar{p}'(\bullet)$. Consider an empty predicate $\emptyset$ with $r = \bar{p}(\bullet)$. Then we find $\bar{p} \in \lambda^r_X\emptyset$; however, $\bar{p}' \notin \lambda^r_X\mathbb{U}$ (for any $\mathbb{U} \subseteq \mathcal{M}_{\mathbb{F}}X$) since the definition of λ^r is independent of $\mathbb{U}$ and, moreover, $\bar{p}'(\bullet) \neq r$.

2. Let $\bar{p}(a, x) \neq \bar{p}'(a, x)$, for some $a \in A, x \in X$. Consider the function

$$p(x') = \begin{cases}\bar{p}(a, x), & \text{if } x' = x \\ 0, & \text{otherwise}\end{cases}$$

   together with a predicate $\mathbb{U} = \{p\}$. Clearly, $\bar{p} \in \lambda^a_X\mathbb{U}$; however, $\bar{p}' \notin \lambda^a_X\mathbb{U}$ because $p(x) = \bar{p}(a, x) \neq \bar{p}'(a, x)$.

□

Lemma A.4.1

Given the predicate lifting σ^a_X and $\sigma^{\downarrow}_X$ introduced in Lemma 6.3.4. Both indexed morphisms preserve finite meets.

Proof: Clearly, σ^a_X (for each $X \in \mathbf{Set}$) preserves finite meet because

$$\begin{aligned}(b, p) \in \sigma^a_X(\bigcap_{i \in I} U_i) &\iff b \in 2 \wedge pa \in \bigcap_{i \in I} U_i \\ &\iff b \in 2 \wedge \forall_{i \in I}\ pa \in U_i \\ &\iff \bigcap_{i \in I}(b, p) \in \sigma^a_X U_i.\end{aligned}$$

Lastly, $\sigma^{\downarrow}_X$ preserves finite meet because

$$\sigma^{\downarrow}_X(\bigcap_{i \in I} U_i) = 1 \times X^A = \bigcap_{i \in I}(1 \times X^A) = \bigcap_{i \in I} \sigma^{\downarrow}_X U_i.$$

□

The proof that λ^a_X, λ^s in the setting of LWA preserve finite meets is similar to the Boolean case.

For the rest of the proofs we have the following situation:

Given an equivalence relation $\equiv \subseteq |C| \times |C|$, we have a reflective subcategory $\mathbf{B}$ of $\mathbf{C}$ which has all the coequalisers and F preserves $\mathbf{B}$.
Let $C \xrightarrow{\alpha} FC \in \mathbf{C}$ be a coalgebra and let $\equiv \; \underset{\pi_2}{\overset{\pi_1}{\rightrightarrows}} |C|$ be some equivalence relation on $|C|$, i.e., $\equiv_L = \{(c, c') \in |C| \times |C| \mid \forall_{\varphi \in \mathbb{M}_\Lambda}\ c \models \varphi \iff c' \models \varphi\}$. Furthermore, we fix ε, η and ε', η' for the (co)unit of the adjunctions $\mathcal{R} \dashv \mathcal{I}_r$ and $I \dashv |\_|$, respectively. Moreover, due to the given adjoint situations, we have the following correspondences on arrows:

$$\frac{\equiv \; \underset{\pi_2}{\overset{\pi_1}{\rightrightarrows}} |C| \quad \in \mathbf{Set}}{I \equiv \; \underset{\pi_2'}{\overset{\pi_1'}{\rightrightarrows}} C \xrightarrow{\eta_C} \mathcal{I}_r\mathcal{R}C \quad \in \mathbf{C}}$$
$$\mathcal{R}I \equiv \; \underset{\tilde{\pi}_2}{\overset{\tilde{\pi}_1}{\rightrightarrows}} \mathcal{R}C \quad \in \mathbf{B}$$

where π_i' is the transpose of π_i under $I \dashv |\_|$ (i.e., $\pi_i' = \varepsilon'_C \circ I\pi_i$ for $i \in \{1, 2\}$) and $\tilde{\pi}_i$ is the transpose of $\eta_C \circ \pi_i'$ under $\mathcal{R} \dashv \mathcal{I}_r$ (i.e., $\tilde{\pi}_i = \varepsilon_{\mathcal{R}C} \circ \mathcal{R}(\eta_C \circ \pi_i')$ for $i \in \{1, 2\}$). Note that $\mathcal{R}\pi_i' = \tilde{\pi}_i$ because of the counit-unit identities. $\mathbf{B}$ has all the coequalisers, so in particular we can construct the following coequaliser:

$$\mathcal{R}I \equiv \; \underset{\tilde{\pi}_2}{\overset{\tilde{\pi}_1}{\rightrightarrows}} \mathcal{R}C \xrightarrow{f} B \quad \in \mathbf{B}.$$

Let g be the transpose of f under $\mathcal{R} \dashv \mathcal{I}_r$, i.e., $g = \mathcal{I}_r f \circ \eta_C$.

Lemma 6.4.4

Given an equivalence relation $\equiv \subseteq |C| \times |C|$. If there is a reflective subcategory $\mathbf{B}$ of $\mathbf{C}$ having all the coequalizers, then for the transpose g of f given by $\mathcal{R}\mathcal{I} \equiv \; \underset{\tilde{\pi}_2}{\overset{\tilde{\pi}_1}{\rightrightarrows}} \mathcal{R}C \xrightarrow{f} B$ we have that $|g|c = |g|c'$ for each $(c, c') \in \equiv$.

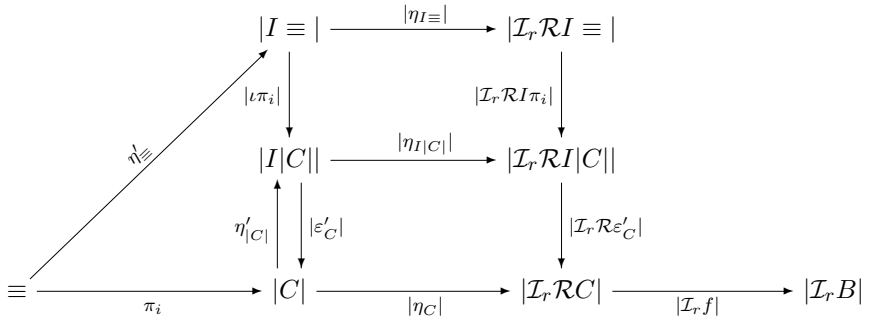
Proof: Next we show that any two equivalent states $(c, c') \in \equiv$ are mapped to same point by $|g|$, i.e.,

$$|\mathcal{I}_r f \circ \eta_C| \circ \pi_1 = |\mathcal{I}_r f \circ \eta_C| \circ \pi_2. \tag{A.2}$$

Recall that $g = \mathcal{I}_r f \circ \eta_C$. For this we first derive

$$\begin{aligned} f \circ \tilde{\pi}_1 = f \circ \tilde{\pi}_2 \implies & \quad f \circ \mathcal{R}\varepsilon'_C \circ \mathcal{R}I\pi_1 = f \circ \mathcal{R}\varepsilon'_C \circ \mathcal{R}I\pi_2 \\ \implies & \quad |\mathcal{I}_r f| \circ |\mathcal{I}_r\mathcal{R}\varepsilon'_C| \circ |\mathcal{I}_r\mathcal{R}I\pi_1| = |\mathcal{I}_r f| \circ |\mathcal{I}_r\mathcal{R}\varepsilon'_C| \circ |\mathcal{I}_r\mathcal{R}I\pi_1| \end{aligned}$$

then plug it into the following commuting diagram in **Set**, where $i \in \{1, 2\}$.



Note that $|\varepsilon'_C| \circ |I\pi_i| \circ \eta'_{\equiv} = \pi_i$ because the naturality of η' gives:

$$|I\pi_i| \circ \eta'_{\equiv} = \eta'_{|C|} \circ \pi_i \implies |\varepsilon'_C| \circ |I\pi_i| \circ \eta'_{\equiv} = |\varepsilon'_C| \circ \eta'_{|C|} \circ \pi_i = \mathrm{id}_{|C|} \circ \pi_i = \pi_i.$$

□

Lemma 6.4.5

Given a coequalizer f under the restrictions described in Lemma 6.4.4 and F preserves $\mathbf{B}$. For its transpose g it holds that $|Fg| \circ |\alpha| \circ \pi_1 = |Fg| \circ |\alpha| \circ \pi_2$ implies $Ff \circ \bar{\mathcal{R}}\alpha \circ \tilde{\pi}_1 = Ff \circ \bar{\mathcal{R}}\alpha \circ \tilde{\pi}_2$.

Proof: Let us see why $|Fg| \circ |\alpha| \circ \pi_1 = |Fg| \circ |\alpha| \circ \pi_2$ implies $Ff \circ \bar{\mathcal{R}}\alpha \circ \tilde{\pi}_1 = Ff \circ \bar{\mathcal{R}}\alpha \circ \tilde{\pi}_2$. Recall that if two arrows are the same, then so are their transpose. Thus, if $|Fg| \circ |\alpha| \circ \pi_1 = |Fg| \circ |\alpha| \circ \pi_2$, then their transposes under the two adjunctions results in the following diagram (see Figure A.3) in $\mathbf{B}$. We first transpose $|Fg| \circ |\alpha| \circ \pi_i$ under $I \dashv |\_|$ and then transpose the obtained transpose under $\mathcal{R} \dashv \mathcal{I}_r$.

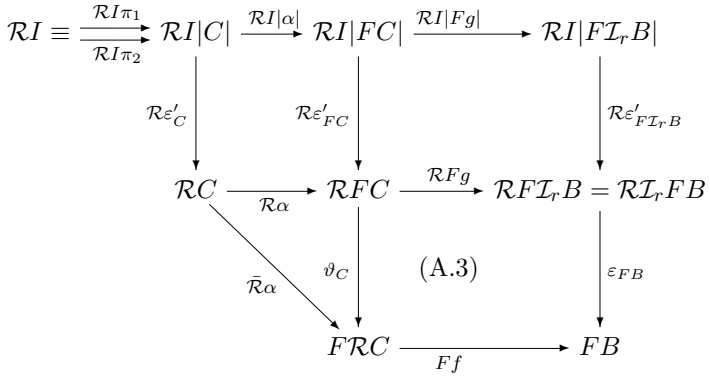


Figure A.3: A commuting diagram.

The commutativity of all squares except (A.3) follows directly from the naturality of the counit ε'. For Square (A.3), again recall that ϑ_C is the transpose of $F\eta_C$, i.e., $\vartheta_C = \varepsilon_{F\mathcal{R}C} \circ \mathcal{R}F\eta_C$ and consider the diagram in Figure A.4 drawn below.

Thus, $|Fg| \circ |\alpha| \circ \pi_1 = |Fg| \circ |\alpha| \circ \pi_2$ implies $Ff \circ \bar{\mathcal{R}}\alpha \circ \tilde{\pi}_1 = Ff \circ \bar{\mathcal{R}}\alpha \circ \tilde{\pi}_2$.

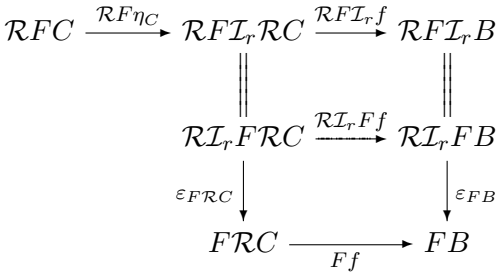


Figure A.4: A commuting diagram due to the (co)unit of the adjunction $\mathcal{R} \dashv \mathcal{I}_r$.

□

Theorem 6.4.10

If $\Phi \xrightarrow{\omega} \tilde{\mathcal{Q}} \circ |\_|$ preserves fibred (co)limits and is injective on objects, Λ is separating for F, and there is a reflective subcategory $\mathbf{B}$ of $\mathbf{C}$ having all the coequalisers and F preserves $\mathbf{B}$, then logical equivalence implies behavioural equivalence.

Proof: Our aim is to construct β by using the universal property of a coequaliser. Thus, we define $\beta(f(c)) = Ff \circ \bar{\mathcal{R}}\alpha(c)$ for $c \in \mathcal{R}C$.

Consider the following commutative diagram in Figure A.5 (without the dashed arrow) in $\mathbf{C}$, where the commutativity of Square (A.5) follows from the commutative diagram drawn in Figure A.6 and by recalling that ϑ_C is the transpose of $F\eta_C$, i.e., $\vartheta_C = \varepsilon_{F\mathcal{R}C} \circ \mathcal{R}F\eta_C$.

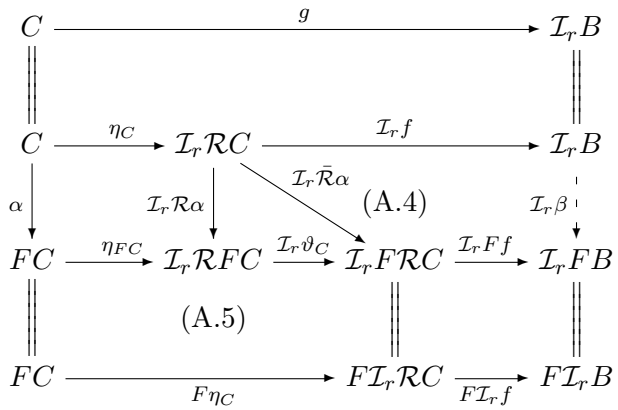


Figure A.5: Construction of β based on the transpose of $f \in \mathbf{B}$.

We already have shown that for any two logical equivalent states c, c' we have $|g|(c) = |g|(c')$ (see Lemma 6.4.4).

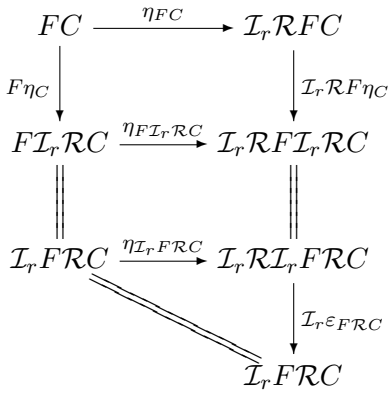
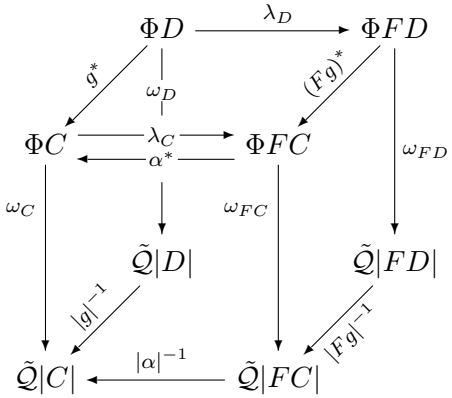


Figure A.6: A commutative diagram due to the (co)unit of the adjunction $\mathcal{R} \dashv \mathcal{I}_r$.

Next, we need to show that $\beta \in \mathbf{B}$ is well defined and therefore we show that by $f \circ \tilde{\pi_1}' = f \circ \tilde{\pi_2}'$ the equation $Ff \circ \bar{\mathcal{R}}\alpha \circ \tilde{\pi}_1 = Ff \circ \bar{\mathcal{R}}\alpha \circ \tilde{\pi}_2$ (see Square A.4 in Figure A.5) holds and based on Lemma 6.4.5 it suffices to proof the equation $|Fg| \circ |\alpha| \circ \pi_1 = |Fg| \circ |\alpha| \circ \pi_2$.

So let $c, c' \in |C|$ such that $c \equiv c'$ and fix $D = \mathcal{I}_rB$. Clearly, the elements $|Fg||\alpha|(c), |Fg||\alpha|(c') \in |FD|$ and we will use the fact F is separating with respect to Λ to conclude that these two elements are the same.
Consider the diagram in (A.6) where the four squares commute (three out of four square commute due to the naturality of ω and one due to the naturality of λ).



(A.6)

Let $U \in \Phi D$ be a predicate. Note that, for any $c_1, c_2 \in |C|$, if $c_1 \in \omega_C(g^*U)$ and $c_2 \notin \omega_C(g^*U)$ then, $c_1 \not\equiv c_2$. To see this pick two points c_1, c_2 such that the antecedent is true. Then, due to the naturality of ω (see (A.6)) we have $c_1 \in |g|^*\omega_D(U)$ and $c_2 \notin |g|^*\omega_D(U)$. Thus, $|g|(c_1) \neq |g|(c_2)$ because otherwise we

would have a contradiction $|g|(c_1) \in \omega_D(U)$ and $|g|(c_1) = |g|(c_2) \notin \omega_D(U)$. Hence, by Equation (A.2) we get $c_1 \not\equiv c_2$.

Moreover, since negations are well-behaved (in Boolean algebras) we find a formula φ_{c_1,c_2} such that $c_1 \not\models_\alpha \varphi_{c_1,c_2}$ and $c_2 \models_\alpha \varphi_{c_1,c_2}$.

Consider $\varphi_U = \bigwedge_{c_1 \notin \omega_C(g^*U)} \bigvee_{c_2 \in \omega_C(g^*U)} \varphi_{c_1,c_2}$ and we claim that

$$\hat{c} \models_\alpha \varphi_U \iff \hat{c} \in \omega_C(g^*U), \qquad \text{for any } \hat{c} \in |C|. \tag{A.7}$$

Suppose $\hat{c} \in \omega_C(g^*U)$. Then we distinguish two cases:

1. If there is no c_1 such that $c_1 \notin \omega_C(g^*U)$. Then, φ_U is equivalent to $\top$, i.e. $[\![\top]\!]$, the terminal object in the fibre category. Clearly, $\hat{c} \in \omega_C([\![\top]\!]) = |C|$ because ω_C preserves fibre limits.

2. If there is some c_1 such that $c_1 \notin \omega_C(g^*U)$. Then, for all such c_1 we have $c_1 \not\equiv \hat{c}$. Thus, there is some distinguishing formula $\varphi_{c_1,\hat{c}}$ such that $\hat{c} \models \varphi_{c_1,\hat{c}}$. I.e.,

$$\hat{c} \in \bigcap_{c_1 \notin \omega_C(g^*U)} \bigcup_{c_2 \in \omega_C(g^*U)} \omega_C([\![\varphi_{c_1,c_2}]\!]_\alpha) \implies \hat{c} \models_\alpha \varphi_U.$$

   The last implication is because ω_C preserves both fibre limits and colimits.

For the converse, suppose $\hat{c} \notin \omega_C(g^*U)$. Then, we distinguish two cases:

1. If there is no c_2 such that $c_2 \in \omega_C(g^*U)$. Then, φ_U is equivalent to $\bot$, i.e., the initial object in the fibre category. Clearly, $c_1 \not\models_\alpha \varphi_U$ because $c_1 \notin \omega_C[\![\bot]\!]$ and $\omega_C[\![\bot]\!] = \emptyset$. The latter is because ω preserves fibred colimits.

2. If there is some c_2 such that $c_2 \in \omega_C(g^*U)$. Then, for all such c_2 there is some distinguishing formula $\varphi_{\hat{c},c_2}$ such that $\hat{c} \not\models_\alpha \varphi_{\hat{c},c_2}$ and $c_2 \models_\alpha \varphi_{\hat{c},c_2}$. I.e., $\hat{c} \notin \bigcup_{c_2 \in \omega_C(g^*U)} \omega_C[\![\varphi_{\hat{c},c_2}]\!]_\alpha$. Clearly,

$$\hat{c} \notin \bigcap_{c_1 \notin \omega_C(g^*U)} \bigcup_{c_2 \in \omega_C(g^*U)} \omega_C[\![\varphi_{c_1,c_2}]\!]_\alpha \implies \hat{c} \notin \omega_c[\![\varphi_U]\!]_\alpha.$$

Thus, (A.7) holds, i.e., $\omega_C[\![\varphi_U]\!]_\alpha = \omega_C(g^*U)$, for any $U \in \Phi D$. And the injectivity of ω_C on objects gives:

$$[\![\varphi_U]\!]_\alpha = g^*U, \quad (\text{for any } U \in \Phi). \tag{A.8}$$

Furthermore, for any $\lambda \in \Lambda$ and $U \in \Phi D$, we derive:

$$\begin{aligned}
|Fg||\alpha|(c) \in \omega_{FD}(\lambda_D U) &\iff c \in |\alpha|^{-1} \circ |Fg|^{-1} \circ \omega_{FD}(\lambda_D U) \\
&\overset{(A.6)}{\iff} c \in \omega_C \circ \alpha^* \circ \lambda_C(g^* U) \\
&\overset{(A.8)}{\iff} c \in \omega_C \circ \alpha^* \circ \lambda_C([\![\varphi_U]\!]_\alpha) \\
&\overset{(\text{Def. } [\lambda])}{\iff} c \models_\alpha [\lambda]\varphi_U \\
&\overset{(c \equiv c')}{\iff} c' \models_\alpha [\lambda]\varphi_U \\
&\iff |Fg||\alpha|(c') \in \omega_{FD}(\lambda_D U).
\end{aligned}$$

Hence $|Fg||\alpha|(c) = |Fg||\alpha|(c')$. □

B Bibliography

[ARP13] Fides Aarts, Joeri de Ruiter, and Erik Poll. "Formal Models of Bank Cards for Free". In: *Sixth IEEE International Conference on Software Testing, Verification and Validation, ICST 2013 Workshops Proceedings, Luxembourg, March 18-22, 2013*. IEEE Computer Society, 2013, pp. 461–468. DOI: 10.1109/ICSTW.2013.60 (cit. on p. 33).

[AJ94] Samson Abramsky and Achim Jung. "Domain Theory". In: *Handbook of Logic in Computer Science*. Ed. by Samson Abramsky, Dov Gabbay, and Thomas Stephen Edward Maibaum. Oxford University Press, 1994, pp. 1–168 (cit. on pp. 225, 269).

[AIS12] Luca Aceto, Anna Ingólfsdóttir, and Jirí Srba. "The algorithmics of bisimilarity". In: *Advanced Topics in Bisimulation and Coinduction*. Ed. by Davide Sangiorgi and Jan J. M. M. Rutten. Vol. 52. Cambridge tracts in theoretical computer science. Cambridge University Press, 2012, pp. 100–172. DOI: 10.1017/CBO9780511792588.004 (cit. on pp. 98, 99).

[AM89] Peter Aczel and Nax Paul Mendler. "A Final Coalgebra Theorem". In: *Category Theory and Computer Science, Manchester, UK, September 5-8, 1989, Proceedings*. Ed. by David H. Pitt, David E. Rydeheard, Peter Dybjer, Andrew M. Pitts, and Axel Poigné. Vol. 389. Lecture Notes in Computer Science. Springer, 1989, pp. 357–365. DOI: 10.1007/BFb0018361 (cit. on p. 48).

[AB+12] Jirí Adámek, Filippo Bonchi, Mathias Hülsbusch, Barbara König, Stefan Milius, and Alexandra Silva. "A Coalgebraic Perspective on Minimization and Determinization". In: *Foundations of Software Science and Computational Structures - 15th International Conference, FOSSACS 2012, Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2012, Tallinn, Estonia, March 24 - April 1, 2012. Proceedings*. Ed. by Lars Birkedal. Vol. 7213. Lecture Notes in Computer Science. Springer, 2012, pp. 58–73. DOI: 10.1007/978-3-642-28729-9_4 (cit. on pp. 201, 202, 208).

[AGT10] Jirí Adámek, H. Peter Gumm, and Vera Trnková. “Presentation of Set Functors: A Coalgebraic Perspective”. In: *Journal of Logic and Computation* 20.5 (2010), pp. 991–1015. DOI: 10.1093/logcom/exn090 (cit. on p. 80).

[AHS09] Jirí Adámek, Horst Herrlich, and George E. Strecker. *Abstract and Concrete Categories - The Joy of Cats*. Dover Publications, 2009. ISBN: 978-0-486-46934-8. URL: http://katmat.math.uni-bremen.de/acc (cit. on pp. 19, 39, 54–56, 59).

[AR94] Jirí Adámek and Jirí Rosický. *Locally Presentable and Accessible Categories*. Vol. 189. London Mathematical Society Lecture Note Series. Cambridge University Press, 1994. URL: https://books.google.de/books?id=iXh6rOd7of0C (cit. on p. 80).

[AFS04] Luca de Alfaro, Marco Faella, and Mariëlle Stoelinga. “Linear and Branching Metrics for Quantitative Transition Systems”. In: *Automata, Languages and Programming: 31st International Colloquium, ICALP 2004, Turku, Finland, July 12-16, 2004. Proceedings*. Ed. by Josep Díaz, Juhani Karhumäki, Arto Lepistö, and Donald Sannella. Vol. 3142. Lecture Notes in Computer Science. Springer, 2004, pp. 97–109. DOI: 10.1007/978-3-540-27836-8_11 (cit. on p. 141).

[AFS09] Luca de Alfaro, Marco Faella, and Mariëlle Stoelinga. “Linear and Branching System Metrics”. In: *IEEE Trans. Software Eng.* 35.2 (2009), pp. 258–273. DOI: 10.1109/TSE.2008.106 (cit. on pp. 17, 136, 141, 142, 158, 274).

[Ash72] Robert B. Ash. *Real Analysis and Probability*. Academic Press, 1972. DOI: 10.1016/C2013-0-06164-6 (cit. on pp. 165, 167, 170, 179).

[Awo06] Steve Awodey. *Category Theory*. Oxford logic guides. Clarendon Press, 2006. ISBN: 978-0-19-856861-2 (cit. on p. 56).

[BB+19] Giorgio Bacci, Giovanni Bacci, Kim G. Larsen, Radu Mardare, Qiyi Tang, and Franck van Breugel. “Computing Probabilistic Bisimilarity Distances for Probabilistic Automata”. In: *30th International Conference on Concurrency Theory, CONCUR 2019, August 27-30, 2019, Amsterdam, the Netherlands*. Ed. by Wan J. Fokkink and Rob van Glabbeek. Vol. 140. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019, 9:1–9:17. DOI: 10.4230/LIPIcs.CONCUR.2019.9 (cit. on p. 275).

[Bae19] John Baez. *Category Theory (Lecture Script)*. Available from author's website. 2019. URL: http://math.ucr.edu/home/baez/qg-winter2016/CategoryTheoryNotes.pdf (cit. on p. 220).

[BK08] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008. ISBN: 978-0-262-02649-9 (cit. on pp. 18, 241).

[BK11] Adriana Balan and Alexander Kurz. "Finitary Functors: From **Set** to **Preord** and **Poset**". In: *Algebra and Coalgebra in Computer Science - 4th International Conference, CALCO 2011, Winchester, UK, August 30 - September 2, 2011. Proceedings*. Ed. by Andrea Corradini, Bartek Klin, and Corina Cîrstea. Vol. 6859. Lecture Notes in Computer Science. Springer, 2011, pp. 85–99. DOI: 10.1007/978-3-642-22944-2_7 (cit. on pp. 76, 77, 132, 133, 278, 279).

[BB+14] Paolo Baldan, Filippo Bonchi, Henning Kerstan, and Barbara König. "Behavioral Metrics via Functor Lifting". In: *34th International Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2014, December 15-17, 2014, New Delhi, India*. Ed. by Venkatesh Raman and S. P. Suresh. Vol. 29. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2014, pp. 403–415. DOI: 10.4230/LIPIcs.FSTTCS.2014.403 (cit. on pp. 18, 136, 137, 140, 145–147, 156, 158, 179, 181, 182, 274, 275).

[BB+18] Paolo Baldan, Filippo Bonchi, Henning Kerstan, and Barbara König. "Coalgebraic Behavioral Metrics". In: *Logical Methods in Computer Science* 14.3 (2018). Selected Papers of the 6th Conference on Algebra and Coalgebra in Computer Science (CALCO 2015). URL: https://lmcs.episciences.org/4827 (cit. on pp. 136–140, 142, 145, 148, 158).

[BK+18] Paolo Baldan, Barbara König, Christina Mika-Michalski, and Tommaso Padoan. *Fixpoint Games on Continuous Lattices*. 2018. URL: https://arxiv.org/abs/1810.11404 (cit. on pp. 239, 243, 248, 258, 263, 266, 276).

[BK+19a] Paolo Baldan, Barbara König, Christina Mika-Michalski, and Tommaso Padoan. "Fixpoint Games on Continuous Lattices". In: *Proc. ACM Symposium on Principles of Programming Languages (POPL)* 3 (2019), 26:1–26:29. DOI: 10.1145/3290339 (cit. on pp. 19, 21–23, 27, 227, 266, 267, 269–271, 276).

[BKP20] Paolo Baldan, Barbara König, and Tommaso Padoan. "Abstraction, Up-To Techniques and Games for Systems of Fixpoint Equations". In: *31st International Conference on Concurrency Theory, CONCUR 2020, September 1-4, 2020, Vienna, Austria (Virtual Conference)*. Ed. by Igor Konnov and Laura Kovács. Vol. 171. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, 25:1–25:20. DOI: 10.4230/LIPIcs.CONCUR.2020.25 (cit. on pp. 271, 276).

[Bal99] Alexandru Baltag. *Truth-as-Simulation: Towards a Coalgebraic Perspective on Logic and Games*. Tech. rep. SEN-R9923. CWI-(Centre for Mathematics and Computer Science), Nov. 1999 (cit. on pp. 13, 17, 181, 263).

[Bal00] Alexandru Baltag. "A Logic for Coalgebraic Simulation". In: *Coalgebraic Methods in Computer Science, CMCS 2000, Berlin, Germany, March 25-26, 2000*. Ed. by Horst Reichel. Vol. 33. Electronic Notes in Theoretical Computer Science. Elsevier, 2000, pp. 42–60. DOI: 10.1016/S1571-0661(05)80343-3 (cit. on p. 125).

[Bar70] Micheal Barr. "Relational algebras". In: *Reports of the Midwest Category Seminar IV*. Vol. 137. lnm. Springer Berlin Heidelberg, 1970. ISBN: 978-3-540-36292-0 (cit. on pp. 76, 77).

[BSd04] Falk Bartels, Ana Sokolova, and Erik de Vink. "A hierarchy of probabilistic system types". In: *Theoretical Computer Science* 327.1 (2004). Selected Papers of CMCS '03, pp. 3–22. ISSN: 0304-3975. DOI: https://doi.org/10.1016/j.tcs.2004.07.019 (cit. on pp. 51, 52, 54).

[Bén85] Jean Bénabou. "Fibered Categories and the Foundations of Naive Category Theory". In: *Journal of Symbolic Logic* 50.1 (1985). published online Cambridge University Press, pp. 10–37. DOI: 10.2307/2273784 (cit. on p. 65).

[BK+19b] Harsh Beohar, Barbara König, Sebastian Küpper, and Christina Mika-Michalski. "Coalgebraic Games in Kleisli Categories". In: *CALCO Early Ideas '19*. 2019 (cit. on pp. 21, 23).

[BK+20] Harsh Beohar, Barbara König, Sebastian Küpper, and Christina Mika-Michalski. *Coalgebraic modal logic and games: an indexed category framework*. unpublished. 2020 (cit. on pp. 19, 21, 23, 65, 222).

[BK+21] Harsh Beohar, Barbara König, Sebastian Küpper, and Christina Mika-Michalski. “Coalgebraic modal logic and games for coalgebras with side effects”. In: *CoRR* abs/2110.09911 (2021). arXiv: 2110.09911. URL: https://arxiv.org/abs/2110.09911 (cit. on p. 23).

[BK+17] Harsh Beohar, Barbara König, Sebastian Küpper, and Alexandra Silva. “Conditional Transition System with Upgrades”. In: *2017 International Symposium on Theoretical Aspects of Software Engineering (TASE)*. Sept. 2017, pp. 1–8. DOI: 10.1109/TASE.2017.8285624 (cit. on p. 222).

[BB07] Patrick Blackburn and Johan van Benthem. “Modal logic: a semantic perspective”. In: *Handbook of Modal Logic*. Ed. by Patrick Blackburn, J. F. A. K. van Benthem, and Frank Wolter. Vol. 3. Studies in logic and practical reasoning. North-Holland/Elsevier, 2007, pp. 1–84. DOI: 10.1016/s1570-2464(07)80004-8 (cit. on p. 15).

[BB+13] Christoph Blume, H. J. Sander Bruggink, Martin Friedrich, and Barbara König. “Treewidth, pathwidth and cospan decompositions with applications to graph-accepting tree automata”. In: *Journal of Visual Language Computing* 24.3 (2013), pp. 192–206. DOI: 10.1016/j.jvlc.2012.10.002 (cit. on p. 41).

[BHR14] Udi Boker, Thomas A. Henzinger, and Arjun Radhakrishna. “Battery transition systems”. In: *The 41st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL ’14, San Diego, CA, USA, January 20-21, 2014*. Ed. by Suresh Jagannathan and Peter Sewell. ACM, 2014, pp. 595–606. DOI: 10.1145/2535838.2535875 (cit. on p. 141).

[BB+12a] Filippo Bonchi, Marcello M. Bonsangue, Michele Boreale, Jan J. M. M. Rutten, and Alexandra Silva. “A coalgebraic perspective on linear weighted automata”. In: *Inf. Comput.* 211 (2012), pp. 77–105. DOI: 10.1016/j.ic.2011.12.002 (cit. on pp. 31, 49–51, 53).

[BB+12b] Filippo Bonchi, Marcello M. Bonsangue, Jan J.M.M. Rutten, and Alexandra Silva. “Brzozowski’s Algorithm (Co)Algebraically”. In: *Logic and Program Semantics - Essays Dedicated to Dexter Kozen on the Occasion of His 60th Birthday*. Ed. by Robert L. Constable and Alexandra Silva. Vol. 7230. Lecture Notes in Computer Science. Springer, 2012, pp. 12–23. DOI: 10.1007/978-3-642-29485-3_2 (cit. on p. 207).

[BG+18] Filippo Bonchi, Pierre Ganty, Roberto Giacobazzi, and Dusko Pavlovic. "Sound up-to techniques and Complete abstract domains". In: *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*. Ed. by Anuj Dawar and Erich Grädel. ACM, 2018, pp. 175–184. DOI: 10.1145/3209108.3209169 (cit. on p. 271).

[BRS08] Marcello Bonsangue, Jan Rutten, and Alexandra Silva. "Coalgebraic Logic and Synthesis of Mealy Machines". In: *Foundations of Software Science and Computational Structures, 11th International Conference, FOSSACS 2008, Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29 - April 6, 2008. Proceedings*. Ed. by Roberto M. Amadio. Vol. 4962. Lecture Notes in Computer Science. Springer, 2008, pp. 231–245. DOI: 10.1007/978-3-540-78499-9_17 (cit. on pp. 36, 38, 125, 131).

[BK05] Marcello M. Bonsangue and Alexander Kurz. "Duality for Logics of Transition Systems". In: *Foundations of Software Science and Computational Structures, 8th International Conference, FOSSACS 2005, Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2005, Edinburgh, UK, April 4-8, 2005, Proceedings*. Ed. by Vladimiro Sassone. Vol. 3441. Lecture Notes in Computer Science. Springer, 2005, pp. 455–469. DOI: 10.1007/978-3-540-31982-5_29 (cit. on p. 184).

[BF16] Wassim Mohamed Boussahel and Georg Frey. "Priced discrete Automata for modeling energy efficient manufacturing systems". In: *13th International Workshop on Discrete Event Systems, WODES 2016, Xi'an, China, May 30 - June 1, 2016*. Ed. by Christos G. Cassandras, Alessandro Giua, and Zhiwu Li. IEEE, 2016, pp. 79–84. DOI: 10.1109/WODES.2016.7497829 (cit. on p. 141).

[BW18] Julian C. Bradfield and Igor Walukiewicz. "The mu-calculus and Model Checking". In: *Handbook of Model Checking*. Ed. by Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem. Springer International Publishing, 2018, pp. 871–919. DOI: 10.1007/978-3-319-10575-8_26 (cit. on pp. 18, 19, 225, 226, 241, 242, 263–265, 269, 270).

[BW05] Franck van Breugel and James Worrell. "A behavioural pseudometric for probabilistic transition systems". In: *Theoretical Computer Science*

331.1 (2005), pp. 115–142. DOI: 10.1016/j.tcs.2004.09.035 (cit. on pp. 136, 137, 162, 163).

[BW06] Franck van Breugel and James Worrell. "Approximating and computing behavioural distances in probabilistic transition systems". In: *Theoretical Computer Science* 360.1-3 (2006), pp. 373–385. DOI: 10.1016/j.tcs.2006.05.021 (cit. on pp. 137, 144, 145, 180, 274).

[BC+97] Anca Browne, Edmund M. Clarke, Somesh Jha, David E. Long, and Wilfredo R. Marrero. "An improved algorithm for the evaluation of fixpoint expressions". In: *Theoretical Computer Science* 178.1–2 (1997), pp. 237–255. DOI: 10.1007/3-540-58179-0_66 (cit. on pp. 233, 270).

[CJ+17] Cristian S. Calude, Sanjay Jain, Bakhadyr Khoussainov, Wei Li, and Frank Stephan. "Deciding parity games in quasipolynomial time". In: *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*. ACM, 2017, pp. 252–263. DOI: 10.1145/3055399.3055409 (cit. on p. 226).

[CPA16] Robert Cartwright, Rebecca Parsons, and Moez A. AbdelGawad. "Domain Theory: An Introduction". In: *CoRR* abs/1605.05858 (2016). arXiv: 1605.05858. URL: http://arxiv.org/abs/1605.05858 (cit. on p. 225).

[CCP18] Valentina Castiglioni, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. "A Logical Characterization of Differential Privacy via Behavioral Metrics". In: *Formal Aspects of Component Software - 15th International Conference, FACS 2018, Pohang, South Korea, October 10-12, 2018, Proceedings*. Ed. by Kyungmin Bae and Peter Csaba Ölveczky. Vol. 11222. Lecture Notes in Computer Science. Springer, 2018, pp. 75–96 (cit. on p. 136).

[CGT16] Valentina Castiglioni, Daniel Gebler, and Simone Tini. "Logical Characterization of Bisimulation Metrics". In: *Proceedings 14th International Workshop Quantitative Aspects of Programming Languages and Systems, QAPL 2016, Eindhoven, The Netherlands, April 2-3, 2016*. Ed. by Mirco Tribastone and Herbert Wiklicky. Vol. 227. EPTCS. 2016, pp. 44–62. DOI: 10.4204/EPTCS.227.4 (cit. on pp. 17, 136).

[CHR10] Pavol Cerný, Thomas A. Henzinger, and Arjun Radhakrishna. "Quantitative Simulation Games". In: *Time for Verification, Essays in Memory of Amir Pnueli*. Ed. by Zohar Manna and Doron A. Peled. Vol. 6200.

Lecture Notes in Computer Science. Springer, 2010, pp. 42–60. DOI: 10.1007/978-3-642-13754-9_3 (cit. on pp. 136, 143).

[CG+14] Konstantinos Chatzikokolakis, Daniel Gebler, Catuscia Palamidessi, and Lili Xu. "Generalized Bisimulation Metrics". In: *CONCUR 2014 - Concurrency Theory - 25th International Conference, Rome, Italy, September 2-5, 2014. Proceedings*. Ed. by Paolo Baldan and Daniele Gorla. Vol. 8704. Lecture Notes in Computer Science. Springer, 2014, pp. 32–46. DOI: 10.1007/978-3-662-44584-6_4 (cit. on pp. 17, 136).

[Che80] Brian F. Chellas. *Modal Logic - An Introduction*. Cambridge University Press, 1980. ISBN: 978-0-51162119-2. DOI: 10.1017/CBO9780511621192 (cit. on p. 277).

[CBW12] Di Chen, Franck van Breugel, and James Worrell. "On the Complexity of Computing Probabilistic Bisimilarity". In: *Foundations of Software Science and Computational Structures - 15th International Conference, FOSSACS 2012, Part of ETAPS 2012, Tallinn, Estonia, March 24 - April 1, 2012. Proceedings*. Vol. 7213. Lecture Notes in Computer Science. Springer, 2012, pp. 437–451. DOI: 10.1007/978-3-642-28729-9_29 (cit. on pp. 182, 275).

[CD08] Xin Chen and Yuxin Deng. "Game Characterizations of Process Equivalences". In: *Programming Languages and Systems. APLAS 2009*. Ed. by G. Ramalingam. Berlin, Heidelberg: Springer, 2008, pp. 107–121. DOI: https://doi.org/10.1007/978-3-540-89330-1_8 (cit. on p. 71).

[CK+11] Corina Cîrstea, Alexander Kurz, Dirk Pattinson, Lutz Schröder, and Yde Venema. "Modal Logics are Coalgebraic". In: *Comput. J.* 54.1 (2011), pp. 31–41. DOI: 10.1093/comjnl/bxp004 (cit. on p. 273).

[CE81] Edmund M. Clarke and E. Allen Emerson. "Design and Synthesis of Synchronization Skeletons Using Branching-Time Temporal Logic". In: *Logics of Programs, Workshop, Yorktown Heights, New York, USA, May 1981*. Ed. by Dexter Kozen. Vol. 131. Lecture Notes in Computer Science. Springer, 1981, pp. 52–71. DOI: 10.1007/BFb0025774 (cit. on p. 241).

[CK91] Edmund M. Clarke and Robert P. Kurshan, eds. *Computer Aided Verification, 2nd International Workshop, CAV '90, New Brunswick, NJ, USA, June 18-21, 1990, Proceedings*. Vol. 531. Lecture Notes in Computer Science. Springer, 1991. ISBN: 3-540-54477-1. DOI: 10.1007/BFb0023712.

[Cle90] Rance Cleaveland. "On Automatically Explaining Bisimulation Inequivalence". In: *Computer-Aided Verification*. Ed. by Edmund M. Clarke and Robert P. Kurshan. Lecture Notes in Computer Science 531. Springer Heidelberg, 1990, pp. 364–372. ISBN: 978-3-540-38394-9. DOI: 10.1007/BFb0023750 (cit. on pp. 17, 21, 72, 96, 116, 125, 126, 274, 291).

[CKS92] Rance Cleaveland, Marion Klein, and Bernhard Steffen. "Faster model checking for the modal Mu-Calculus". In: *Proc. of CAV 1992*. Vol. 663. Lecture Notes in Computer Science. Springer, 1992, pp. 410–422. DOI: 10.1007/3-540-56496-9_32 (cit. on pp. 19, 233, 243, 266, 270).

[CC77] Patrick Cousot and Radhia Cousot. "Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints". In: *Conference Record of the Fourth ACM Symposium on Principles of Programming Languages, Los Angeles, California, USA, January 1977*. ACM, 1977, pp. 238–252. DOI: 10.1145/512950.512973 (cit. on pp. 269, 271).

[CC79] Radhia Cousot and Patrick Cousot. "Constructive versions of Tarski's fixed point theorems." In: *Pacific Journal of Mathematics* 82.1 (1979), pp. 43–57 (cit. on p. 27).

[CRH02] P.J.L. Cuijpers, Michel A. Reniers, and W.P.M.H. (Maurice) Heemels. *Hybrid Transition Systems*. Vol. 0212. Computer science reports. Eindhoven University of Technology, 2002 (cit. on p. 140).

[DP02] Brian A. Davey and Hilary A. Priestley. *Introduction to lattices and order*. Cambridge University Press, 2002. DOI: https://doi.org/10.1017/CBO9780511809088 (cit. on pp. 25, 27, 227).

[dR99] E.P. de Vink and J.J.M.M. Rutten. "Bisimulation for probabilistic transition systems: a coalgebraic approach". In: *Theoretical Computer Science* 221.1 (1999), pp. 271–293. ISSN: 0304-3975. DOI: 10.1016/S0304-3975(99)00035-3 (cit. on p. 49).

[DM+19] Hans-Peter Deifel, Stefan Milius, Lutz Schröder, and Thorsten Wißmann. "Generic Partition Refinement and Weighted Tree Automata". In: *Formal Methods - The Next 30 Years - Third World Congress, FM 2019, Porto, Portugal, October 7-11, 2019, Proceedings*. Ed. by Maurice H. ter Beek, Annabelle McIver, and José N. Oliveira. Vol. 11800. Lecture Notes in Computer Science. Springer, 2019, pp. 280–297. DOI: 10.1007/978-3-030-30942-8_18 (cit. on pp. 101, 134).

[Des99] Josée Desharnais. “Labelled Markov processes”. PhD thesis. McGill University, Montreal, Nov. 1999 (cit. on pp. 17, 136).

[DG+04] Josée Desharnais, Vineet Gupta, Radha Jagadeesan, and Prakash Panangaden. “Metrics for labelled Markov processes”. In: *Theoretical Computer Science* 318.3 (2004), pp. 323–354. DOI: 10.1016/j.tcs.2003.09.013 (cit. on pp. 17, 136, 137, 180).

[DLT08] Josée Desharnais, François Laviolette, and Mathieu Tracol. “Approximate Analysis of Probabilistic Processes: Logic, Simulation and Games”. In: *Fifth International Conference on the Quantitative Evaluaiton of Systems (QEST 2008), 14-17 September 2008, Saint-Malo, France*. IEEE Computer Society, 2008, pp. 264–273. DOI: 10.1109/QEST.2008.42 (cit. on pp. 13, 18, 71, 90–92, 125, 136, 137, 212, 274).

[DMS19] Ulrich Dorsch, Stefan Milius, and Lutz Schröder. “Graded Monads and Graded Logics for the Linear Time - Branching Time Spectrum”. In: *30th International Conference on Concurrency Theory, CONCUR 2019, August 27-30, 2019, Amsterdam, the Netherlands*. Ed. by Wan J. Fokkink and Rob van Glabbeek. Vol. 140. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019, 36:1–36:16. DOI: 10.4230/LIPIcs.CONCUR.2019.36 (cit. on p. 223).

[DM+17] Ulrich Dorsch, Stefan Milius, Lutz Schröder, and Thorsten Wißmann. “Efficient Coalgebraic Partition Refinement”. In: *28th International Conference on Concurrency Theory, CONCUR 2017, September 5-8, 2017, Berlin, Germany*. Ed. by Roland Meyer and Uwe Nestmann. Vol. 85. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017, 32:1–32:16. DOI: 10.4230/LIPIcs.CONCUR.2017.32 (cit. on pp. 17, 21, 97–101, 103, 109, 110, 112, 125, 274).

[DM+18] Ulrich Dorsch, Stefan Milius, Lutz Schröder, and Thorsten Wißmann. “Predicate Liftings and Functor Presentations in Coalgebraic Expression Languages”. In: *Coalgebraic Methods in Computer Science - 14th International Workshop, CMCS 2018, Colocated with ETAPS 2018, Thessaloniki, Greece, April 14-15, 2018, Revised Selected Papers*. Ed. by Corina Cîrstea. Vol. 11202. Lecture Notes in Computer Science. Springer, 2018, pp. 56–77. DOI: 10.1007/978-3-030-00389-0_5 (cit. on pp. 108, 279).

[DI02] Daniel C. DuVarney and S. Purushothaman Iyer. “C Wolf - A Toolset for Extracting Models from C Programs”. In: *Formal Techniques for Networked and Distributed Systems - FORTE 2002, 22nd IFIP WG 6.1 International Conference Houston, Texas, USA, November 11-14, 2002, Proceedings*. Ed. by Doron A. Peled and Moshe Y. Vardi. Vol. 2529. Lecture Notes in Computer Science. Springer, 2002, pp. 260–275. DOI: 10.1007/3-540-36135-9_17 (cit. on p. 131).

[Dwo06] Cynthia Dwork. “Differential Privacy”. In: *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II*. Ed. by Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener. Vol. 4052. Lecture Notes in Computer Science. Springer, 2006, pp. 1–12 (cit. on p. 136).

[DH+01] Matthew B. Dwyer, John Hatcliff, Roby Joehanes, Shawn Laubach, Corina S. Pasareanu, Robby, Hongjun Zheng, and Willem Visser. “Tool-Supported Program Abstraction for Finite-State Verification”. In: *Proceedings of the 23rd International Conference on Software Engineering, ICSE 2001, 12-19 May 2001, Toronto, Ontario, Canada*. Ed. by Hausi A. Müller, Mary Jean Harrold, and Wilhelm Schäfer. IEEE Computer Society, 2001, pp. 177–187. DOI: 10.1109/ICSE.2001.919092 (cit. on p. 131).

[EM42] Samuel Eilenberg and Saunders Maclane. “Natural Isomorphisms in Group Theory.” In: *Proceedings of the National Academy of Sciences of the United States of America* 28 12 (1942), pp. 537–43. DOI: 10.1073/pnas.28.12.537 (cit. on p. 39).

[EKN12] Pantelis E. Eleftheriou, Costas D. Koutras, and Christos Nomikos. “Notions of bisimulation for Heyting-valued modal languages”. In: *Journal of Logic and Computation* 22 (2012), pp. 213–235 (cit. on p. 270).

[Eme85] E. Allen Emerson. “Automata, tableaux, and temporal logics”. In: *Proceedings of Logics of Programs 1985*. Ed. by R. Parikh. Vol. 193. Lecture Notes in Computer Science. Springer, 1985, pp. 79–88. DOI: 10.1007/3-540-15648-8_7 (cit. on p. 263).

[EJ91] E. Allen Emerson and Charanjit S. Jutla. “Tree Automata, Mu-Calculus and Determinacy (Extended Abstract)”. In: *32nd Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 1-4 October*

1991. IEEE Computer Society, 1991, pp. 368–377. DOI: 10.1109/SFCS.1991.185392 (cit. on pp. 18, 19, 225, 247, 264, 265, 270, 276).

[FLT11] Uli Fahrenberg, Axel Legay, and Claus R. Thrane. "The Quantitative Linear-Time–Branching-Time Spectrum". In: *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2011, December 12-14, 2011, Mumbai, India*. Ed. by Supratik Chakraborty and Amit Kumar. Vol. 13. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2011, pp. 103–114. DOI: 10.4230/LIPIcs.FSTTCS.2011.103 (cit. on pp. 17, 136).

[FKP17] Nathanaël Fijalkow, Bartek Klin, and Prakash Panangaden. "Expressiveness of Probabilistic Modal Logics, Revisited". In: *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*. Ed. by Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl. Vol. 80. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017, 105:1–105:12. DOI: 10.4230/LIPIcs.ICALP.2017.105 (cit. on p. 71).

[FGK10] Diana Fischer, Erich Grädel, and Łukasz Kaiser. "Model Checking Games for the Quantitative mu-Calculus". In: *Theory Comput. Syst.* 47.3 (2010), pp. 696–719. URL: http://logic.rwth-aachen.de/~kaiser/quantitative_games_journal.pdf (cit. on pp. 270, 276).

[FV99] Kathi Fisler and Moshe Y. Vardi. "Bisimulation and Model Checking". In: *Correct Hardware Design and Verification Methods, 10th IFIP Advanced Research Working Conference, CHARME '99, Bad Herrenalb, Germany, September 27-29, 1999, Proceedings*. Ed. by Laurence Pierre and Thomas Kropf. Vol. 1703. Lecture Notes in Computer Science. Springer, 1999, pp. 338–341. DOI: 10.1007/3-540-48153-2_29 (cit. on pp. 18, 241).

[Fit91] Melvin Fitting. "Many-valued modal logics". In: *Fundamenta Informaticae* 15 (1991), pp. 235–254 (cit. on p. 270).

[FLV10] Gaëlle Fontaine, Raul Andres Leal, and Yde Venema. "Automata for Coalgebras: An Approach Using Predicate Liftings". In: *Automata, Languages and Programming, 37th International Colloquium, ICALP 2010, Bordeaux, France, July 6-10, 2010, Proceedings, Part II*. Ed. by Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis. Vol. 6199. Lecture Notes in Computer

Science. Springer, 2010, pp. 381–392. DOI: 10.1007/978-3-642-14162-1_32 (cit. on pp. 125, 265).

[FKW17] David de Frutos-Escrig, Jeroen J. A. Keiren, and Tim A. C. Willemse. "Games for Bisimulations and Abstraction". In: *Logical Methods in Computer Science* 13.4 (2017). DOI: 10.23638/LMCS-13(4:15)2017 (cit. on p. 127).

[GK09] Nicola Gambino and Joachim Kock. "Polynomial functors and polynomial monads". In: *Mathematical Proceedings of the Cambridge Philosophical Society* 154 (June 2009), pp. 153–192. DOI: 10.1017/S0305004112000394 (cit. on p. 42).

[GS11] Thomas Martin Gawlitza and Helmut Seidl. "Solving systems of rational equations through strategy iteration". In: *ACM Trans. Program. Lang. Syst.* 33.3 (2011), 11:1–11:48 (cit. on p. 271).

[GH+80] Gerhard Gierz, Karl H. Hofmann, Klaus Keimel, Jimmie D. Lawson, Michael W. Mislove, and Dana S. Scott. *A Compendium of Continuous Lattices*. Springer, 1980. DOI: 10.1007/978-3-642-67678-9.

[GH+03] Gerhard Gierz, Karl H. Hofmann, Klaus Keimel, Jimmie D. Lawson, Michael W. Mislove, and Dana S. Scott. *Continuous Lattices and Domains*. Cambridge University Press, 2003. DOI: 10.1007/s11225-007-9052-y (cit. on pp. 25, 27, 225, 227, 230, 269, 276).

[GZ12] Antoine Girard and Gang Zheng. "Verification of Safety and Liveness Properties of Metric Transition Systems". In: *ACM Trans. Embed. Comput. Syst.* 11.S2 (2012), 54:1–54:23. DOI: 10.1145/2331147.2331164 (cit. on p. 141).

[Gla01] Rob J. van Glabbeek. "The Linear Time - Branching Time Spectrum I. The Semantics of Concrete, Sequential Processes". In: *Handbook of Process Algebra*. Ed. by Jan A. Bergstra, Alban Ponse, and Scott A. Smolka. North-Holland/Elsevier, 2001, pp. 3–99. DOI: 10.1016/b978-044482830-9/50019-9 (cit. on pp. 14, 35, 71, 183).

[GS13] Daniel Gorín and Lutz Schröder. "Simulations and Bisimulations for Coalgebraic Modal Logics". In: *Algebra and Coalgebra in Computer Science - 5th International Conference, CALCO 2013, Warsaw, Poland, September 3-6, 2013. Proceedings*. Ed. by Reiko Heckel and Stefan Milius. Vol. 8089. Lecture Notes in Computer Science. Springer, 2013, pp. 253–266. DOI: 10.1007/978-3-642-40206-7_19 (cit. on pp. 78, 94, 126).

[GO14] Erich Grädel and Martin Otto. "The Freedoms of (Guarded) Bisimulation". In: *Johan van Benthem on Logic and Information Dynamics. Outstanding Contributions to Logic*. Ed. by Alexandru Baltag and Sonja Smets. Vol. 5. Springer, 2014, pp. 3–31. DOI: 10.1007/978-3-319-06025-5_1 (cit. on p. 15).

[Gra66] John W. Gray. "Fibred and Cofibred Categories". In: *Proceedings of the Conference on Categorical Algebra*. Ed. by S. Eilenberg, D. K. Harrison, S. MacLane, and H. Röhrl. Berlin, Heidelberg: Springer Berlin Heidelberg, 1966, pp. 21–83 (cit. on p. 66).

[Gro71] Alexander Grothendieck. "Catégories fibrées et descente". In: *Revêtements étales et groupe fondamental*. Lecture Notes in Mathematics. Springer Berlin Heidelberg, 1971. DOI: 10.1007/BFb0058656 (cit. on p. 65).

[GR02] Alexander Grothendieck and Michele Raynaud. *Revêtements étales et groupe fondamental (SGA 1)*. 2002. arXiv: math/0206203 [math.AG] (cit. on p. 65).

[GL+05] Orna Grumberg, Martin Lange, Martin Leucker, and Sharon Shoham. "Don't Know in the μ-Calculus". In: *Proc. of VMCAI '05*. Ed. by Radhia Cousot. Vol. 3385. Lecture Notes in Computer Science. Springer, 2005, pp. 233–249 (cit. on p. 270).

[GV08] Orna Grumberg and Helmut Veith. "25 Years of Model Checking - History, Achievements, Perspectives". In: *25 Years of Model Checking*. Springer Berlin Heidelberg, 2008. DOI: 10.1007/978-3-540-69850-0 (cit. on p. 241).

[GS01] H. Peter Gumm and Tobias Schröder. "Monoid-labeled transition systems". In: *Coalgebraic Methods in Computer Science, CMCS 2001, a Satellite Event of ETAPS 2001, Genova, Italy, April 6-7, 2001*. Ed. by Andrea Corradini, Marina Lenisa, and Ugo Montanari. Vol. 44-1. Electronic Notes in Theoretical Computer Science. Elsevier, 2001, pp. 185–204. DOI: 10.1016/S1571-0661(04)80908-3 (cit. on pp. 51, 53).

[Han03] Helle Hvid Hansen. "Monotonic Modal Logics". MA thesis. University of Amsterdam, 2003 (cit. on pp. 126, 277, 278).

[HK04] Helle Hvid Hansen and Clemens Kupke. "A Coalgebraic Perspective on Monotone Modal Logic". In: *Proceedings of the Workshop on Coalgebraic Methods in Computer Science, CMCS 2004, Barcelona, Spain, March*

27-29, 2004. Ed. by Jirí Adámek and Stefan Milius. Vol. 106. Electronic Notes in Theoretical Computer Science. Elsevier, 2004, pp. 121–143. DOI: 10.1016/j.entcs.2004.02.028 (cit. on pp. 16, 277, 279, 282, 283, 289).

[HK+17] Helle Hvid Hansen, Clemens Kupke, Johannes Marti, and Yde Venema. "Parity Games and Automata for Game Logic". In: *Dynamic Logic. New Trends and Applications - First International Workshop, DALI 2017, Brasilia, Brazil, September 23-24, 2017, Proceedings*. Vol. 10669. Lecture Notes in Computer Science. Springer, 2017, pp. 115–132. DOI: 10.1007/978-3-319-73579-5_8 (cit. on p. 246).

[HKP07] Helle Hvid Hansen, Clemens Kupke, and Eric Pacuit. "Bisimulation for Neighbourhood Structures". In: *Algebra and Coalgebra in Computer Science, Second International Conference, CALCO 2007, Bergen, Norway, August 20-24, 2007, Proceedings*. Ed. by Till Mossakowski, Ugo Montanari, and Magne Haveraaen. Vol. 4624. Lecture Notes in Computer Science. Springer, 2007, pp. 279–293. DOI: 10.1007/978-3-540-73859-6_19 (cit. on p. 51).

[HJS07] Ichiro Hasuo, Bart Jacobs, and Ana Sokolova. "Generic Trace Semantics via Coinduction". In: *Logical Methods in Computer Science* 3.4 (2007). DOI: 10.2168/LMCS-3(4:11)2007 (cit. on pp. 61, 62, 186, 275).

[HKC18] Ichiro Hasuo, Toshiki Kataoka, and Kenta Cho. "Coinductive predicates and final sequences in a fibration". In: *Mathematical Structures in Computer Science* 28.4 (2018), pp. 562–611. DOI: 10.1017/S0960129517000056 (cit. on p. 183).

[HSC16] Ichiro Hasuo, Shunsuke Shimizu, and Corina Cîrstea. "Lattice-theoretic progress measures and coalgebraic model checking". In: *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2016, St. Petersburg, FL, USA, January 20 - 22, 2016*. Ed. by Rastislav Bodík and Rupak Majumdar. ACM, 2016, pp. 718–732. DOI: 10.1145/2837614.2837673 (cit. on pp. 19, 226, 227, 231, 233, 234, 238, 239, 243, 266, 267, 269, 270, 276).

[Hen61] Leon A. Henkin. "Some Remarks on Infinitely Long Formulas". In: *Journal of Symbolic Logic*. Pergamon Press, 1961, pp. 167–183. DOI: 10.2307/2270594 (cit. on p. 14).

[HM80] Matthew Hennessy and Robin Milner. “On observing Nondeterminism and Concurrency”. In: *Automata, Languages and Programming*. Ed. by Jaco de Bakker and Jan van Leeuwen. Proc. of ICALP ’80, LNCS vol. 85. Springer Berlin Heidelberg, 1980, pp. 299–309. ISBN: 978-3-540-39346-7 (cit. on pp. 15, 71).

[HM85] Matthew Hennessy and Robin Milner. “Algebraic Laws for Nondeterminism and Concurrency”. In: *J. ACM* 32.1 (Jan. 1985), pp. 137–161. ISSN: 0004-5411. DOI: 10.1145/2455.2460 (cit. on pp. 72, 74, 114).

[HJ98] Claudio Hermida and Bart Jacobs. “Structural Induction and Coinduction in a Fibrational Setting”. In: *Information and Computation* 145.2 (1998), pp. 107–152. DOI: 10.1006/inco.1998.2725 (cit. on pp. 17, 183, 202, 273).

[HH90] H. Herrlich and M. Hušek. “Galois connections categorically”. In: *Journal of Pure and Applied Algebra* 68.1 (1990). Special Issue in Honor of B. Banaschewski, pp. 165–180. ISSN: 0022-4049 (cit. on p. 221).

[Hin68] Jaakko Hintikka. “Language-Games for Quantifiers”. In: *Studies in Logical Theory*. Ed. by Nicholas Rescher. Oxford:Blackwell, 1968, pp. 46–72 (cit. on pp. 13, 15, 225).

[Hir98] Daniel Hirschkoff. “Automatically Proving Up-to Bisimulation”. In: *Electronic Notes in Theoretical Computer Science* 18 (1998), pp. 75–89. DOI: 10.1016/S1571-0661(05)80251-8 (cit. on p. 271).

[Hir99] Daniel Hirschkoff. “Mise en oeuvre de preuves de bisimulation”. PhD thesis. Ecole Nationale des Ponts et Chaussées, 1999 (cit. on p. 271).

[HV19] Wilfrid Hodges and Jouko Väänänen. “Logic and Games”. In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Fall 2019. Metaphysics Research Lab, Stanford University, 2019. URL: https://plato.stanford.edu/archives/fall2019/entries/logic-games/ (cit. on pp. 14, 15).

[Hop71] John E. Hopcraft. “An n log n algorithm for minimizing states in a finite automaton.” In: *Theory of machines and computations*. Academic Press, New York, 1971, pp. 189–196 (cit. on pp. 17, 31).

[HU79] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979. ISBN: 0-201-02988-X (cit. on p. 35).

[Hop20] Jonas Hoppe. "Automatic Explanation of Non-Bisimilarity for Probabilistic Transition Systems". Bachelor's thesis. Universität Duisburg-Essen, Fakultät für Ingenieurwissenschaften, Abteilung für Informatik und Angewandte Kognitionswissenschaft, June 2020 (cit. on p. 144).

[HK97] Michael Huth and Marta Kwiatkowska. "Quantitative analysis and model checking". In: *Proc. of LICS '97*. IEEE, 1997, pp. 111–122. DOI: 10.1109/LICS.1997.614940 (cit. on p. 270).

[Jac99] Bart Jacobs. *Categorical Logic and Type Theory*. Ed. by S. Abramsky, S. Artemov, R. A. Shoare, and A. S. Troelstra. 1st. Vol. 141. Studies in Logic and the Foundations of Mathematics. North Holland, Elsevier, Jan. 1999. ISBN: 0444501703 (cit. on pp. 65–67, 188).

[Jac01] Bart Jacobs. "Many-sorted coalgebraic modal logic: a model-theoretic study". In: *RAIRO - Theoretical Informatics and Applications - Informatique Théorique et Applications* 35.1 (2001), pp. 31–59. URL: http://eudml.org/doc/92654 (cit. on pp. 17, 209, 273).

[Jac04] Bart Jacobs. "Trace Semantics for Coalgebras". In: *Electronic Notes in Theoretical Computer Science* 106 (2004). Proceedings of the Workshop on Coalgebraic Methods in Computer Science (CMCS), pp. 167–184. ISSN: 1571-0661. DOI: 10.1016/j.entcs.2004.02.031 (cit. on p. 186).

[Jac10] Bart Jacobs. *Predicate Logic for Functors and Monads*. Available from author's website. 2010. URL: http://www.cs.ru.nl/~bart/PAPERS/predlift-indcat.pdf (cit. on pp. 65, 66, 68, 184, 185, 188, 208, 275).

[JSS15] Bart Jacobs, Alexandra Silva, and Ana Sokolova. "Trace semantics via determinization". In: *Journal of Computer and System Science* 81.5 (2015), pp. 859–879. DOI: 10.1016/j.jcss.2014.12.005 (cit. on pp. 60, 62, 183–187, 193, 195, 197, 198, 220, 222, 275).

[JS09] Bart Jacobs and Ana Sokolova. "Exemplaric Expressivity of Modal Logics". In: *Journal of Logic and Computation* 20.5 (Feb. 2009), pp. 1041–1068. ISSN: 0955-792X. DOI: 10.1093/logcom/exn093 (cit. on pp. 68, 183, 184).

[Jur00] Marcin Jurdziński. "Small Progress Measures for Solving Parity Games". In: *STACS 2000, 17th Annual Symposium on Theoretical Aspects of Computer Science, Lille, France, February 2000, Proceedings*. Ed. by Horst Reichel and Sophie Tison. Vol. 1770. Lecture Notes in Computer

Science. Springer, 2000, pp. 290–301. DOI: 10.1007/3-540-46541-3_24 (cit. on pp. 18, 19, 225, 227, 231, 232, 264, 266, 267, 269, 270, 276).

[KS90] Paris C. Kanellakis and Scott A. Smolka. "CCS Expressions, Finite State Processes, and Three Problems of Equivalence". In: *Inf. Comput.* 86 (1990), pp. 43–68. DOI: 10.1016/0890-5401(90)90025-D (cit. on pp. 31, 97, 98, 104, 110, 112).

[Ker16] Henning Kerstan. "Coalgebraic Behavior Analysis – From Qualitative to Quantitative Analyses". PhD thesis. Universität Duisburg-Essen, Fakultät für Ingenieurwissenschaften, Abteilung für Informatik und Angewandte Kognitionswissenschaft, May 2016 (cit. on pp. 56, 61, 139, 145, 146, 148, 158, 159, 181, 275).

[KM15] Narges Khakpour and Mohammad Reza Mousavi. "Notions of Conformance Testing for Cyber-Physical Systems: Overview and Roadmap (Invited Paper)". In: *26th International Conference on Concurrency Theory (CONCUR 2015)*. Ed. by Luca Aceto and David de Frutos Escrig. Vol. 42. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2015, pp. 18–40. ISBN: 978-3-939897-91-0. DOI: 10.4230/LIPIcs.CONCUR.2015.18 (cit. on pp. 17, 31, 136, 140, 141).

[KK11] Christian Kissig and Alexander Kurz. "Generic Trace Logics". 2011. arXiv: 1103.3239. URL: http://arxiv.org/abs/1103.3239 (cit. on p. 184).

[Kli07] Bartek Klin. "Coalgebraic Modal Logic Beyond Sets". In: *Proceedings of the 23rd Conference on the Mathematical Foundations of Programming Semantics, MFPS 2007, New Orleans, LA, USA, April 11-14, 2007*. Ed. by Marcelo Fiore. Vol. 173. Electronic Notes in Theoretical Computer Science. Elsevier, 2007, pp. 177–201. DOI: 10.1016/j.entcs.2007.02.034 (cit. on p. 184).

[KR16] Bartek Klin and Jurriaan Rot. "Coalgebraic trace semantics via forgetful logics". In: *Logical Methods in Computer Science* 12.4 (2016). DOI: 10.2168/LMCS-12(4:10)2016 (cit. on p. 184).

[KRM17] Urvashi Kodwani, Sonal Rajurkar, and S. Mundada. "Realization of sequential circuit using finite state machine". In: *2017 International Conference on Intelligent Computing and Control Systems (ICICCS)*

(2017), pp. 472–476. DOI: 10.1109/ICCONS.2017.8250767 (cit. on p. 131).

[KK+19] Yuichi Komorida, Shin-Ya Katsumata, Nick Hu, Bartek Klin, and Ichiro Hasuo. "Codensity Games for Bisimilarity". In: *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019, Vancouver, BC, Canada, June 24-27, 2019*. IEEE, 2019, pp. 1–13. DOI: 10.1109/LICS.2019.8785691 (cit. on pp. 18, 125, 137, 180, 181, 183, 185, 209, 223, 275, 276).

[KK14] Barbara König and Sebastian Küpper. "Generic Partition Refinement Algorithms for Coalgebras and an Instantiation to Weighted Automata". In: *Theoretical Computer Science - 8th IFIP TC 1/WG 2.2 International Conference, TCS 2014, Rome, Italy, September 1-3, 2014. Proceedings*. Ed. by Josep Díaz, Ivan Lanese, and Davide Sangiorgi. Vol. 8705. Lecture Notes in Computer Science. Springer, 2014, pp. 311–325. DOI: 10.1007/978-3-662-44602-7_24 (cit. on p. 99).

[KK18] Barbara König and Sebastian Küpper. "A Generalized Partition Refinement Algorithm, Instantiated to Language Equivalence Checking for Weighted Automata". In: *Soft Computing* 22.4 (2018). DOI: 10.1007/s00500-016-2363-z (cit. on pp. 17, 55, 97–100, 102, 103, 110, 187).

[KKM17] Barbara König, Sebastian Küpper, and Christina Mika. "PAWS: A Tool for the Analysis of Weighted Systems". In: *Proceedings 15th Workshop on Quantitative Aspects of Programming Languages and Systems, QAPL@ETAPS 2017, Uppsala, Sweden, 23rd April 2017*. 2017, pp. 75–91. DOI: 10.4204/EPTCS.250.5 (cit. on pp. 100, 127, 134).

[KM17a] Barbara König and Christina Mika. "Bisimulation Games on Coalgebras". In: *CoRR* abs/1705.10165 (2017). arXiv: 1705.10165. URL: http://arxiv.org/abs/1705.10165 (cit. on p. 137).

[KM17b] Barbara König and Christina Mika. "Bisimulation Games on Coalgebras*". In: *CALCO Early Ideas '17*. 2017 (cit. on pp. 22, 277).

[KM18] Barbara König and Christina Mika-Michalski. "(Metric) Bisimulation Games and Real-Valued Modal Logics for Coalgebras". In: *29th International Conference on Concurrency Theory (CONCUR 2018)*. Ed. by S. Schewe and L. Zhang. Vol. 118. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2018, 37:1–37:17. ISBN: 978-3-95977-087-3. DOI:

10.4230/LIPIcs.CONCUR.2018.37 (cit. on pp. 21, 22, 85, 106, 108, 125, 126, 183, 212, 219, 290).

[KMS20a] Barbara König, Christina Mika-Michalski, and Lutz Schröder. *Explaining Non-Bisimilarity in a Coalgebraic Approach: Games and Distinguishing Formulas*. arXiv:2002.11459. 2020. URL: https://arxiv.org/abs/2002.11459.

[KMS20b] Barbara König, Christina Mika-Michalski, and Lutz Schröder. "Explaining Non-Bisimilarity in a Coalgebraic Approach: Games and Distinguishing Formulas". In: *Coalgebraic Methods in Computer Science - 15th International Workshop, CMCS 2020, Colocated with ETAPS 2020, Proceedings*. Ed. by Daniela Petrisan and Jurriaan Rot. Vol. 12094. Lecture Notes in Computer Science. Springer, 2020, pp. 133–154. DOI: 10.1007/978-3-030-57201-3_8 (cit. on pp. 21–23, 33, 96, 128, 129, 277).

[Kot15a] Stefan Kottwitz. *A mindmap showing TeX projects supported by DANTE e.V.* Website. https://www.texample.net/tikz/examples/servers/ visited on January 15, 2021. 2015 (cit. on p. 97).

[Kot15b] Stefan Kottwitz. *Latex Cookbook*. Packt, 2015. ISBN: 9781784395148. URL: https://www.packtpub.com/hardware-and-creative/latex-cookbook (cit. on p. 97).

[Koz83] Dexter Kozen. "Results on the Propositional μ-Calculus". In: *Theoretical Computer Science* 27.3 (1983). Special Issue Ninth International Colloquium on Automata, Languages and Programming (ICALP) Aarhus, Summer 1982, pp. 333–354. DOI: 10.1016/0304-3975(82)90125-6 (cit. on pp. 18, 225, 270).

[KW99] Marcus Kracht and Frank Wolter. "Normal Monomodal Logics Can Simulate All Others". In: *The Journal of Symbolic Logic* 64.1 (1999), pp. 99–138 (cit. on p. 126).

[KL07] Orna Kupfermann and Yoad Lustig. "Latticed Simulation Relations and Games". In: *Proc. of ATVA '07*. Vol. 4672. Lecture Notes in Computer Science. Springer, 2007, pp. 316–330 (cit. on p. 270).

[Kup07] Clemens Kupke. "Terminal Sequence Induction via Games". In: *Logic, Language, and Computation, 7th International Tbilisi Symposium on Logic, Language, and Computation, TbiLLC 2007, Tbilisi, Georgia, October 1-5, 2007. Revised Selected Papers*. Ed. by Peter Bosch, David

Gabelaia, and Jérôme Lang. Vol. 5422. Lecture Notes in Computer Science. Springer, 2007, pp. 257–271. DOI: 10.1007/978-3-642-00665-4_21 (cit. on pp. 125, 183).

[KKP04] Clemens Kupke, Alexander Kurz, and Dirk Pattinson. "Algebraic Semantics for Coalgebraic Logics". In: *Proceedings of the Workshop on Coalgebraic Methods in Computer Science, CMCS 2004, Barcelona, Spain, March 27-29, 2004*. Ed. by Jirí Adámek and Stefan Milius. Vol. 106. Electronic Notes in Theoretical Computer Science. Elsevier, 2004, pp. 219–241. DOI: 10.1016/j.entcs.2004.02.037 (cit. on p. 184).

[KL09] Clemens Kupke and Raul Andres Leal. "Characterising Behavioural Equivalence: Three Sides of One Coin". In: *Algebra and Coalgebra in Computer Science, Third International Conference, CALCO 2009, Udine, Italy, September 7-10, 2009. Proceedings*. Ed. by Alexander Kurz, Marina Lenisa, and Andrzej Tarlecki. Vol. 5728. Lecture Notes in Computer Science. Springer, 2009, pp. 97–112. DOI: 10.1007/978-3-642-03741-2_8 (cit. on p. 75).

[KR20] Clemens Kupke and Jurriaan Rot. "Expressive Logics for Coinductive Predicates". In: *28th EACSL Annual Conference on Computer Science Logic, CSL 2020, January 13-16, 2020, Barcelona, Spain*. Ed. by Maribel Fernández and Anca Muscholl. Vol. 152. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, 26:1–26:18. DOI: 10.4230/LIPIcs.CSL.2020.26 (cit. on pp. 183, 184, 223, 275, 276).

[Küp17] Sebastian Küpper. "Behavioural Analysis of Systems with Weights and Conditions". PhD thesis. Universität Duisburg-Essen, Fakultät für Ingenieurwissenschaften, Abteilung für Informatik und Angewandte Kognitionswissenschaft, Nov. 2017 (cit. on pp. 65, 98–100, 134).

[Kur00] Alexander Kurz. "Logics for coalgebras and applications to computer science". PhD thesis. Universität München, May 2000 (cit. on pp. 49, 75).

[LS89] Kim G. Larsen and Arne Skou. "Bisimulation Through Probabilistic Testing". In: *Conference Record of the Sixteenth Annual ACM Symposium on Principles of Programming Languages, Austin, Texas, USA, January 11-13, 1989*. ACM Press, 1989, pp. 344–352. DOI: 10.1145/75277.75307 (cit. on pp. 32, 36, 92, 125).

[LR19] Fosco Loregian and Emily Riehl. “Categorical notions of fibration”. In: *Expositiones Mathematicae* (2019). ISSN: 0723-0869. DOI: 10.1016/j.exmath.2019.02.004 (cit. on p. 65).

[Mac98] Saunders MacLane. *Categories for the Working Mathematician*. 2nd edition. New York: Springer-Verlag, 1998, pp. ix+317. DOI: 10.1007/978-1-4757-4721-8 (cit. on pp. 56, 59).

[ME45] Saunders MacLane and Samuel Eilenberg. “General Theory of Natural Equivalences”. In: *Transactions of the American Mathematical Society* (1945), pp. 231–294. DOI: https://doi.org/10.2307/1990284 (cit. on p. 39).

[Mad97] Angelika Mader. “Verification of Modal Properties Using Boolean Equation Systems”. PhD thesis. TU München, 1997 (cit. on p. 272).

[MPT08] Ondrej Majer, AHTI-Veikko Pietarinen, and Tero Tulenheimo. “Games and logic in philosophy”. In: vol. 15. 2008. ISBN: 978-1-4020-9373-9. DOI: 10.1007/978-1-4020-9374-6.

[MPT09] Ondrej Majer, AHTI-Veikko Pietarinen, and Tero Tulenheimo, eds. *Games: Unifying Logic, Language, and Philosophy*. Vol. 15. Logic, Epistemology, and the Unity of Science. Springer, 2009. ISBN: 978-1-4020-9373-9. DOI: 10.1007/978-1-4020-9374-6 (cit. on p. 15).

[Mar19] Iza Marfisi-Schottman. “Games in Higher Education”. In: *Encyclopedia of Education and Information Technologies*. Ed. by Arthur Tatnall. Cham: Springer International Publishing, 2019, pp. 1–9. ISBN: 978-3-319-60013-0. DOI: 10.1007/978-3-319-60013-0_35-1 (cit. on p. 13).

[Mar76] George Markowsky. “Chain-complete posets and directed sets with applications”. In: *Algebra Universalis* 6 (1976), pp. 53–68. DOI: 10.1007/BF02485815.

[MV12] Johannes Marti and Yde Venema. “Lax Extensions of Coalgebra Functors”. In: *Coalgebraic Methods in Computer Science - 11th International Workshop, CMCS 2012, Colocated with ETAPS 2012, Tallinn, Estonia, March 31 - April 1, 2012, Revised Selected Papers*. Ed. by Dirk Pattinson and Lutz Schröder. Vol. 7399. Lecture Notes in Computer Science. Springer, 2012, pp. 150–169. DOI: 10.1007/978-3-642-32784-1_9 (cit. on p. 280).

[MS19] Cristina Matache and Sam Staton. “A Sound and Complete Logic for Algebraic Effects”. In: *Foundations of Software Science and Computation Structures - 22nd International Conference, FOSSACS 2019, Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, April 6-11, 2019, Proceedings*. Ed. by Mikolaj Bojanczyk and Alex Simpson. Vol. 11425. Lecture Notes in Computer Science. Springer, 2019, pp. 382–399. DOI: 10.1007/978-3-030-17127-8_22 (cit. on p. 72).

[May07] Merrilea J. Mayo. “Games for Science and Engineering Education”. In: *Commun. ACM* 50.7 (July 2007), pp. 30–35. ISSN: 0001-0782. DOI: 10.1145/1272516.1272536 (cit. on p. 13).

[MM07] Annabelle McIver and Carroll Morgan. “Results on the quantitative μ-calculus qMμ”. In: *ACM Trans. Comp. Log.* 8.1:3 (2007) (cit. on p. 270).

[Mea55] George H. Mealy. “A Method for Synthesizing Sequential Circuits”. In: *Bell System Technical Journal* 34.5 (1955), pp. 1045–1079. DOI: 10.1002/j.1538-7305.1955.tb03788.x (cit. on pp. 33, 34, 36–38).

[MK16] Murtaza Mehdi and Aihab Khan. “DNA Pattern Analysis using FA, Mealy and Moore Machines”. In: *International Journal of Computer Science and Information Security* 14 (Sept. 2016), p. 9 (cit. on pp. 33, 131).

[MR+17] Michael Michalski, Martin Rieth, Andreas Kempf, and Jens H. Krüger. “CoFlaVis: A Visualization System for Pulverized Coal Flames”. In: *Comput. Sci. Eng.* 19.6 (2017), pp. 72–78. DOI: 10.1109/MCSE.2017.3971156 (cit. on p. 31).

[Mil00] Stefan Milius. “Relations in Categories”. MA thesis. Graduate Program in Mathematics and Statistics York University Toronto, Ontario, June 2000 (cit. on p. 200).

[MPS09] Stefan Milius, Thorsten Palm, and Daniel Schwencke. “Complete Iterativity for Algebras with Effects”. In: *Algebra and Coalgebra in Computer Science, Third International Conference, CALCO 2009, Udine, Italy, September 7-10, 2009. Proceedings*. Ed. by Alexander Kurz, Marina Lenisa, and Andrzej Tarlecki. Vol. 5728. Lecture Notes in Computer Science. Springer, 2009, pp. 34–48. DOI: 10.1007/978-3-642-03741-2_4 (cit. on p. 62).

[Mio12] Matteo Mio. "On the Equivalence of Game and Denotational Semantics for the Probabilistic μ-Calculus". In: *Logical Methods in Computer Science* 8.2:07 (2012), pp. 1–21. DOI: 10.2168/LMCS-8(2:7)2012 (cit. on pp. 270, 276).

[MS17] Matteo Mio and Alex Simpson. "Łukasiewicz μ-calculus". In: *Fundamenta Informaticae* 150.3-4 (2017), pp. 317–346. DOI: 10.3233/FI-2017-1472 (cit. on pp. 271, 276).

[Moh09] Mehryar Mohri. "Weighted Automata Algorithms". English. In: *Handbook of Weighted Automata*. Ed. by Manfred Droste, Werner Kuich, and Heiko Vogler. Springer, 2009, pp. 213–254. ISBN: 978-3-642-01491-8. DOI: 10.1007/978-3-642-01492-5_6 (cit. on p. 31).

[MPR96] Mehryar Mohri, Fernando Pereira, and Michael Riley. "Weighted Automata in Text and Speech Processing". In: *IN ECAI-96 WORKSHOP*. Vol. abs/cs/0503077. John Wiley and Sons, 1996, pp. 46–50. URL: http://arxiv.org/abs/cs/0503077 (cit. on p. 31).

[MON70] RICHARD MONTAGUE. "Universal grammar". In: *Theoria* 36.3 (1970), pp. 373–398. DOI: https://doi.org/10.1111/j.1755-2567.1970.tb00434.x (cit. on p. 277).

[Moo56] Edward F. Moore. "Gedanken-Experiments on Sequential Machines". In: *Automata Studies*. Ed. by Claude Shannon and John McCarthy. Princeton, NJ: Princeton University Press, 1956, pp. 129–153 (cit. on p. 36).

[Mos99] Lawrence S. Moss. "Coalgebraic logic". In: *Annals of Pure and Applied Logic - Dedicated to Rohit Parikh on his 60th birthday* 96.1–3 (1999), pp. 277–317. DOI: 10.1016/S0168-0072(98)00042-6 (cit. on pp. 17, 125).

[MP82] Herve Moulin and Bezalel Peleg. "Cores of effectivity functions and implementation theory". In: *Journal of Mathematical Economics* 10.1 (1982), pp. 115–145. ISSN: 0304-4068. DOI: https://doi.org/10.1016/0304-4068(82)90009-X (cit. on p. 277).

[Mul94] Philip S. Mulry. "Lifting theorems for Kleisli categories". In: *Mathematical Foundations of Programming Semantics*. Ed. by Stephen Brookes, Michael Main, Austin Melton, Michael Mislove, and David Schmidt. Berlin, Heidelberg: Springer Berlin Heidelberg, 1994, pp. 304–319. DOI: 10.1007/3-540-58027-1_15 (cit. on pp. 61, 186).

[nca09] ncatlab-anonymous. *Rel - Limits and Colimits*. Revision 15 on July 2014 and notes of the forum `https://math.stackexchange.com/questions/1931577/equalizers-dont-exist-in-rel` visited last on September 2021. 2009. URL: `http://ncatlab.org/nlab/show/Rel#LimitsAndColimit` (visited on 09/30/2021) (cit. on p. 201).

[Nic11] Rocco De Nicola. "Behavioral Equivalences". In: *Encyclopedia of Parallel Computing*. Ed. by David A. Padua. Springer, 2011, pp. 120–127. DOI: `10.1007/978-0-387-09766-4_517` (cit. on p. 35).

[NNH99] Flemming Nielson, Hanne Riis Nielson, and Chris Hankin. *Principles of Program Analysis*. Springer, 1999. DOI: `10.1007/978-3-662-03811-6` (cit. on pp. 225, 241, 244, 245).

[Ott04] Martin Otto. *Elementary proof of the van Benthem-Rosen characterisation theorem*. Tech. rep. 2342. Department of Mathematics, Technische Universität Darmstadt, 2004 (cit. on pp. 15, 71).

[Pac17] Eric Pacuit. *Neighborhood Semantics for Modal Logic*. Springer International Publishing, Jan. 2017, pp. xii, 154. ISBN: 978-3-319-67148-2. DOI: `10.1007/978-3-319-67149-9` (cit. on p. 277).

[PT87] Robert Paige and Robert Endre Tarjan. "Three Partition Refinement Algorithms". In: *SIAM Journal on Computing* 16.6 (1987), pp. 973–989. DOI: `10.1137/0216062` (cit. on pp. 31, 98, 99, 101, 110, 112, 125, 274).

[PS78] Robert Paré and Dietmar Schumacher. "Abstract families and the adjoint functor theorems". In: *Indexed Categories and Their Applications*. Springer, 1978, pp. 1–125. DOI: `10.1007/BFb0061361` (cit. on p. 68).

[Pat03] Dirk Pattinson. "Coalgebraic modal logic: soundness, completeness and decidability of local consequence". In: *Theoretical Computer Science* 309.1 (2003), pp. 177–193. ISSN: 0304-3975. DOI: `https://doi.org/10.1016/S0304-3975(03)00201-9` (cit. on pp. 73, 78, 84, 85, 183, 208).

[Pat04] Dirk Pattinson. "Expressive Logics for Coalgebras via Terminal Sequence Induction". In: *Notre Dame J. Formal Logic* 45.1 (2004). Duke University Press, pp. 19–33. DOI: `10.1305/ndjfl/1094155277` (cit. on pp. 17, 55, 77, 80, 210, 273, 290).

[Pau01] Marc Pauly. "Logic for social software." PhD thesis. University of Amsterdam, Interfacultary Research Institutes, Institute for Logic, Language and Computation (ILLC), 2001 (cit. on pp. 13, 277, 278).

[Pit00] Andrew M. Pitts. "Categorical Logic". In: *Handbook of Logic in Computer Science, Volume 5. Algebraic and Logical Structures*. Ed. by S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum. Oxford University Press, 2000. Chap. 2, pp. 39–128. ISBN: 0-19-853781-6 (cit. on p. 188).

[PS11] Damien Pous and Davide Sangiorgi. "Enhancements of the bisimulation proof method". In: *Advanced Topics in Bisimulation and Coinduction*. Ed. by Davide Sangiorgi and Jan Rutten. Cambridge University Press, 2011 (cit. on p. 271).

[PT99] John Power and Daniele Turi. "A Coalgebraic Foundation for Linear Time Semantics". In: *Conference on Category Theory and Computer Science, CTCS 1999, Edinburgh, UK, December 10-12, 1999*. Ed. by Martin Hofmann, Giuseppe Rosolini, and Dusko Pavlovic. Vol. 29. Electronic Notes in Theoretical Computer Science. Elsevier, 1999, pp. 259–274. DOI: 10.1016/S1571-0661(05)80319-6 (cit. on pp. 55, 56, 62, 63, 65, 222, 275).

[QS82] Jean-Pierre Queille and Joseph Sifakis. "Specification and verification of concurrent systems in CESAR". In: *International Symposium on Programming, 5th Colloquium, Torino, Italy, April 6-8, 1982, Proceedings*. Ed. by Mariangiola Dezani-Ciancaglini and Ugo Montanari. Vol. 137. Lecture Notes in Computer Science. Springer, 1982, pp. 337–351. DOI: 10.1007/3-540-11494-7_22 (cit. on p. 241).

[Ran52] George N. Raney. "Completely Distributive Complete Lattices". In: *Proceedings of the American Mathematical Society* 3.5 (1952), pp. 677–680. ISSN: 00029939, 10886826. URL: http://www.jstor.org/stable/2032165 (cit. on p. 231).

[Roj15] Raúl Rojas. "A Tutorial Introduction to the Lambda Calculus". In: *CoRR* abs/1503.09060 (2015). arXiv: 1503.09060. URL: http://arxiv.org/abs/1503.09060 (cit. on p. 225).

[Röß00] Martin Rößiger. "Coalgebras and Modal Logic". In: *Coalgebraic Methods in Computer Science, CMCS 2000, Berlin, Germany, March 25-26, 2000*. Ed. by Horst Reichel. Vol. 33. Electronic Notes in Theoretical Computer Science. Elsevier, 2000, pp. 294–315. DOI: 10.1016/S1571-0661(05)80353-6 (cit. on p. 17).

[Rut00] Jan J. M. M. Rutten. “Universal coalgebra: a theory of systems”. In: *Theoretical Computer Science* 249.1 (2000), pp. 3–80. DOI: 10.1016/S0304-3975(00)00056-6 (cit. on pp. 16, 46, 48, 49, 51, 52, 72).

[San09] Davide Sangiorgi. “On the origins of bisimulation and coinduction”. In: *ACM Trans. Program. Lang. Syst.* 31.4 (2009), 15:1–15:41. DOI: 10.1145/1516507.1516510 (cit. on pp. 14, 71).

[San11] Davide Sangiorgi. *Introduction to Bisimulation and Coinduction*. Cambridge University Press, 2011. DOI: 10.1017/CBO9780511777110 (cit. on pp. 13, 14, 16, 29, 30, 35, 36).

[Sch08] Lutz Schröder. “Expressivity of Coalgebraic Modal Logic: The Limits and Beyond”. In: *Theoretical Computer Science* 390.2-3 (2008). Foundations of Software Science and Computational Structures, pp. 230–247. DOI: 10.1016/j.tcs.2007.09.023 (cit. on pp. 17, 73, 78, 80, 83, 85, 89, 121, 183, 184, 273, 274, 288, 290).

[SP11] Lutz Schröder and Dirk Pattinson. “Description Logics and Fuzzy Probability”. In: *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*. Ed. by Toby Walsh. IJCAI/AAAI, 2011, pp. 1075–1081. DOI: 10.5591/978-1-57735-516-8/IJCAI11-184 (cit. on p. 137).

[Sch61] Marcel-Paul Schützenberger. “On the Definition of a Family of Automata”. In: *Information and Control* 4.2–3 (1961), pp. 245–270 (cit. on p. 185).

[Sco70] Dana Scott. “Advice on Modal Logic”. In: *Philosophical Problems in Logic: Some Recent Developments*. Ed. by Karel Lambert. Dordrecht: Springer Netherlands, 1970, pp. 143–173. ISBN: 978-94-010-3272-8. DOI: 10.1007/978-94-010-3272-8_7 (cit. on p. 277).

[Sco72] Dana Scott. “Continuous lattices”. In: *Toposes, Algebraic Geometry and Logic*. Ed. by F. W. Lawvere. Lecture Notes in Mathematics. Springer, 1972, pp. 97–136. DOI: 10.1007/BFb0073961 (cit. on pp. 225, 226, 229, 269, 276).

[Sei96] Helmut Seidl. “Fast and simple nested fixpoints”. In: *Information Processing Letters* 59.6 (1996). Elsevier, pp. 303–308. DOI: 10.1016/0020-0190(96)00130-5 (cit. on pp. 19, 225, 233, 243, 270).

[Sok11] Ana Sokolova. “Probabilistic systems coalgebraically: A survey”. In: *Theoretical Computer Science* 412.38 (2011), pp. 5095–5110. DOI: 10.1016/j.tcs.2011.05.008 (cit. on pp. 31, 32).

[SK+18] David Sprunger, Shin-ya Katsumata, Jérémy Dubut, and Ichiro Hasuo. “Fibrational Bisimulations and Quantitative Reasoning”. In: *Coalgebraic Methods in Computer Science - 14th International Workshop, CMCS 2018, Colocated with ETAPS 2018, Thessaloniki, Greece, April 14-15, 2018, Revised Selected Papers*. Ed. by Corina Cîrstea. Vol. 11202. Lecture Notes in Computer Science. Springer, 2018, pp. 190–213. DOI: 10.1007/978-3-030-00389-0_11 (cit. on p. 181).

[Sta09] Sam Staton. “Relating Coalgebraic Notions of Bisimulation”. In: *Algebra and Coalgebra in Computer Science, Third International Conference, CALCO 2009, Udine, Italy, September 7-10, 2009. Proceedings*. Ed. by Alexander Kurz, Marina Lenisa, and Andrzej Tarlecki. Vol. 5728. Lecture Notes in Computer Science. Springer, 2009, pp. 191–205. DOI: 10.1007/978-3-642-03741-2_14 (cit. on p. 73).

[SS98] Perdita Stevens and Colin Stirling. “Practical Model-Checking Using Games”. In: *Tools and Algorithms for Construction and Analysis of Systems, 4th International Conference, TACAS ’98, Part of the European Joint Conferences on the Theory and Practice of Software, ETAPS’98, Lisbon, Portugal, March 28 - April 4, 1998, Proceedings*. Ed. by Bernhard Steffen. Vol. 1384. Lecture Notes in Computer Science. Springer, 1998, pp. 85–101. DOI: 10.1007/BFb0054166 (cit. on p. 271).

[Sti95] Colin Stirling. “Lokal Model Checking Games”. In: *CONCUR ’95: Concurrency Theory, 6th International Conference, Philadelphia, PA, USA, August 21-24, 1995, Proceedings*. Ed. by Insup Lee and Scott A. Smolka. Vol. 962. Lecture Notes in Computer Science. Springer, 1995, pp. 1–11. DOI: 10.1007/3-540-60218-6_1 (cit. on pp. 18, 225, 264, 269, 270, 276).

[Sti99] Colin Stirling. “Bisimulation, Modal Logic and Model Checking Games”. In: *Logic Journal of the IGPL* 7.1 (1999), pp. 103–124. DOI: 10.1093/jigpal/7.1.103 (cit. on pp. 13–15, 71, 73, 74, 90, 125).

[SW91] Colin Stirling and David Walker. “Local Model Checking in the Modal mu-Calculus”. In: *Theoretical Computer Science* 89.1 (1991), pp. 161–177 (cit. on p. 263).

[Tar55] Alfred Tarski. “A lattice-theoretical theorem and its applications”. In: *Pacific Journal of Mathematics* 5 (1955), pp. 285–309. URL: https://projecteuclid.org:443/euclid.pjm/1103044538 (cit. on pp. 27, 36, 232).

[Trn80] Věra Trnková. “General theory of relational automata”. In: *Fundamenta Informaticae* 3 (1980), pp. 189–234 (cit. on pp. 76, 77).

[TR98] Daniele Turi and Jan J. M. M. Rutten. “On the foundations of final coalgebra semantics: non-well-founded sets, partial orders, metric spaces”. In: *Mathematical Structures in Computer Science* 8.5 (1998), pp. 481–540. DOI: 10.1017/S0960129598002588 (cit. on pp. 17, 136, 150, 161, 180, 275).

[Tur37] Alan M. Turing. “Computability and λ-definability”. In: *Journal of Symbolic Logic* 2.4 (1937), pp. 153–163. DOI: 10.2307/2268280 (cit. on p. 225).

[Van13] Peter Vankúš. *Didactic Games in Mathematics*. Faculty of Mathematics, Physics and Informatics, Comenius University Bratislava, Dec. 2013, pp. 8, 18. ISBN: 978-80-8147-007-3. DOI: 10.13140/2.1.3138.9120 (cit. on p. 13).

[Vel17] Jiri Velebil. *Categorical methods in universal coalgebra*. TACL 2017 Summer School Lecture Notes. 2017 (cit. on pp. 58, 59).

[Ven08] Yde Venema. *Lectures on the modal μ-calculus*. Lecture notes, Institute for Logic, Language and Computation, University of Amsterdam. 2008 (cit. on pp. 226, 246, 264).

[Ver06] Anatoly Vershik. “Kantorovich Metric: Initial History and Little-Known Applications”. In: *Journal of Mathematical Sciences* 133 (Mar. 2006), pp. 1410–1417. DOI: 10.1007/s10958-006-0056-3 (cit. on p. 145).

[Vil09] Cédric Villani. “Optimal transport – Old and new”. In: *Grundlehren der mathematischen Wissenschaften*. Vol. 338. Springer Berlin Heidelberg, 2009, pp. xxii+973. DOI: 10.1007/978-3-540-71050-9 (cit. on pp. 139, 145, 156, 274).

[WS21] Paul Wild and Lutz Schröder. “A Quantified Coalgebraic van Benthem Theorem”. In: *Foundations of Software Science and Computation Structures - 24th International Conference, FOSSACS 2021, Held as Part of the European Joint Conferences on Theory and Practice of*

Software, ETAPS 2021, Luxembourg City, March 27 - April 1, 2021, Proceedings. Ed. by Stefan Kiefer and Christine Tasson. Vol. 12650. Lecture Notes in Computer Science. Springer, 2021, pp. 551–571. DOI: 10.1007/978-3-030-71995-1_28 (cit. on p. 180).

[WS+18a] Paul Wild, Lutz Schröder, Dirk Pattinson, and Barbara König. "A van Benthem Theorem for Fuzzy Modal Logic". In: *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*. Ed. by Anuj Dawar and Erich Grädel. ACM, 2018, pp. 909–918. DOI: 10.1145/3209108.3209180 (cit. on pp. 18, 137, 165, 167, 180).

[WS+18b] Paul Wild, Lutz Schröder, Dirk Pattinson, and Barbara König. "A van Benthem Theorem for Quantitative Probabilistic Modal Logic". In: *CoRR* abs/1810.04722 (2018). URL: http://arxiv.org/abs/1810.04722 (cit. on pp. 181, 275).

[Wiß] Thorsten Wißmann. *Personal Communication* (cit. on p. 110).

[WD+18] Thorsten Wißmann, Ulrich Dorsch, Stefan Milius, and Lutz Schröder. "Efficient and Modular Coalgebraic Partition Refinement". In: *CoRR* abs/1806.05654 (2018). URL: http://arxiv.org/abs/1806.05654 (cit. on p. 17).

[WD+20] Thorsten Wißmann, Ulrich Dorsch, Stefan Milius, and Lutz Schröder. "Efficient and Modular Coalgebraic Partition Refinement". In: *Logical Methods in Computer Science* Volume 16, Issue 1 (2020). DOI: 10.23638/LMCS-16(1:8)2020 (cit. on pp. 55, 100, 101, 134).

[WMS21] Thorsten Wißmann, Stefan Milius, and Lutz Schröder. "Explaining Behavioural Inequivalence Generically in Quasilinear Time". In: *32nd International Conference on Concurrency Theory, CONCUR 2021, August 24-27, 2021, Virtual Conference*. Ed. by Serge Haddad and Daniele Varacca. Vol. 203. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, 32:1–32:18. DOI: 10.4230/LIPIcs.CONCUR.2021.32 (cit. on pp. 102, 112, 126).

[Wor05] James Worrell. "On the final sequence of a finitary Set functor". In: *Theoretical Computer Science* 338.1-3 (2005), pp. 184–199. DOI: 10.1016/j.tcs.2004.12.009 (cit. on pp. 54, 55, 80, 98).

[Yan96] Wuu Yang. “Mealy Machines are a Better Model of Lexical Analyzers”. In: *Computer Languages* 22.1 (1996), pp. 27–38. DOI: 10.1016/0096-0551(96)00003-3 (cit. on p. 33).

[Zie98] Wieslaw Zielonka. “Infinite Games on Finitely Coloured Graphs with Applications to Automata on Infinite Trees”. In: *Theoretical Computer Science* 200.1-2 (1998), pp. 135–183. DOI: 10.1016/S0304-3975(98)00009-7 (cit. on p. 247).

C

List of Symbols

Symbol	Description
S	Spoiler, p. 14
D	Duplicator, p. 14
$\mathbb{N}, \mathbb{N}_0$	natural numbers, p. 25
$\mathbb{Q}, \mathbb{R}$	rational and real numbers, p. 25
$\mathbb{Q}_0, \mathbb{R}_0$	rational and real numbers including 0, p. 25
$+$	disjoint union, p. 16
$\sim$	we write $x \sim y$ if x, y are bisimilar. p. 35
2	we write 2 for $\{0, 1\}$, p. 77
$\bullet$	$1 = \{\bullet\}$, usually associated with termination, p. 16
$\hat{p}$	to denote the subset of a characteristic function $p : X \to 2$, p. 78
$[x]_R$	equivalence class $[x]$ for an equivalence relation R, p. 26
X/R	equivalence classes based on an equivalence relation R, p. 26
$\equiv$	to denote an equivalence relation
LTS	Labelled transition system, p. 13
NDA	non-deterministic automata, p. 62
DFA	deterministic finite automata, p. 31
MTS	metric transition system, p. 140
Set	the category of sets and functions, p. 40
Rel	the category of sets and relations, p. 40
$\mathbf{K}\ell(T)$	the Kleisli category induced by a monad (T, η, μ), p. 61
$\mathbf{C}^{\mathrm{op}}$	opposite category, p. 42
Cat	category of small categories, p. 65
$\mathbb{E}(\Phi)$	*category of elements* aka the *Grothendieck construction*, p. 67
$\mathbf{cBA}_{\wedge}$	category of complete Boolean algebras, p. 210
id_C	identity morphism over the object C, p. 25
Id_F	identity natural transformation over a functor F, p. 45
$ID_{\mathbf{C}}$	identity functor over a category $\mathbf{C}$

$\lvert\_\rvert$	forgetful functor $\mathbf{C} \to \mathbf{Set}$, p. 185
$\mathcal{P}$	the powerset functor, p. 43
$\mathcal{P}_f$	the *finite* powerset functor, p. 43
$\mathcal{Q}, \tilde{\mathcal{Q}}$	the contravariant powerset functors, p. 65, 78
$\mathcal{D}$	the distribution functor, p. 43
$\mathcal{M}_{\mathbb{F}}$	multiset functor over a semiring S, p. 60
$\mathcal{M}$	the monotone neighbourhood functor, p. 108
$\dashv$	to denote an adjunction, p. 56
f^*	for Φf with $\Phi : \mathbf{C}^{op} \to \mathbf{Cat}$, p. 65
HM	Hennessy-Milner logic, p. 74
$\Box, \Diamond$	the modal operators of **HM**, p. 15
Λ	a set of predicate liftings, p. 80
ev, ev_F	an evaluation map $F2 \to 2$ for a functor F, p. 79
γ	a quantitative evaluation map $F[0,\top] \to [0,\top]$ for a functor F, p. 146
λ	predicate lifting as natural transformation, p.78
$\mathcal{M}(\Gamma)$	quantitative coalgebraic modal logic, p. 162
$\mathbb{M}_\Lambda$	coalgebraic modal logic beyond **Set**, p. 208
$[\![\varphi]\!]$	semantic of some formula φ, p. 84, 162
$md(\varphi)$	modal depth, p. 162
d^L_α	logical distance, p. 162
d^G_α	game distance, p. 171
$\leq^F$	lifted order, p.76
$d^{\uparrow F}$	the Kantorovich pseudometric, p. 146
$\tilde{F}_\gamma f$	$\gamma \circ Ff$, p. 146
$d_e, d_\ominus$	euclidean metric, directed metric, p. 139
$\xrightarrow{1}$	non-expansive function, p. 139
$\mathcal{B}_\varepsilon(x_i)$	an ε-ball around x_i, p. 165
$\hookrightarrow$	inclusion arrow, p. 202
$\uparrow l$	upward-closure for $l \in L$ where L is a lattice, p. 227
P	partial ordered set, p. 26
$\bigsqcup, \bigsqcap$	join and meet of a subset $X \subseteq P$, p. 26
$\top, \bot$	top and bottom of an lattice L, p. 26
μ, ν	standard notation for the least and greatest fixpoint (respectively), p. 233
η	$\in \{\mu, \nu\}$, p. 233
$\ll$	way-below (relation), p. 228
$\downarrow l$	the sets of elements way below l, p. 229

$\underline{n}$	integer interval $\{1, \ldots, n\}$, p. 230
$(P^n, \preceq)$	lexicographical order with P a partial order, p. 231
$\preceq_i$	truncated lexicographical order, p. 231
$\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$	system of fixpoint equations, p. 233
$sol(E)$	solution of a system of fixpoint equations E, p. 234
$\mathbf{E}(), \mathbf{A}()$	possible moves of $\exists$ and $\forall$, p. 247
T-Beg	Tool for behavioural equivalence games, p. 127
Paws	Tool for the analysis of weighted systems, p. 127

Index