Reachability Games and Friends: A Journey Through the Lens of Memory and Complexity

Thomas Brihaye
UMONS – Université de Mons, Belgium

Aline Goeminne
F.R.S.-FNRS & UMONS – Université de Mons, Belgium

James C. A. Main
F.R.S.-FNRS & UMONS – Université de Mons, Belgium

Mickael Randour
F.R.S.-FNRS & UMONS – Université de Mons, Belgium

Abstract

Reachability objectives are arguably the most basic ones in the theory of games on graphs (and beyond). But far from being bland, they constitute the cornerstone of this field. Reachability is everywhere, as are the tools we use to reason about it. In this invited contribution, we take the reader on a journey through a zoo of models that have reachability objectives at their core. Our goal is to illustrate how model complexity impacts the complexity of strategies needed to play optimally in the corresponding games and computational complexity.

2012 ACM Subject Classification Software and its engineering → Formal methods; Theory of computation → Logic and verification; Theory of computation → Solution concepts in game theory

Keywords and phrases Games on graphs, reachability, finite-memory strategies, complexity

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.1

Category Invited Talk

Funding This work has been supported by the Fonds de la Recherche Scientifique – FNRS under Grant n° T.0027.21 (PDR RatBeCoSi) and Grant n° T.0188.23 (PDR ControlleRS).
Aline Goeminne: Postdoctoral Researcher of the Fonds de la Recherche Scientifique – FNRS.
James C. A. Main: Research Fellow of the Fonds de la Recherche Scientifique – FNRS, member of the TRAIL Institute.
Mickael Randour: Research Associate of the Fonds de la Recherche Scientifique – FNRS, member of the TRAIL Institute.

Acknowledgements The authors are grateful to Patricia Bouyer-Decitre for carefully reading through previous versions of this paper and providing valuable comments.

1 Introduction

Games on Graphs. We consider infinite-duration multi-player turn-based games played on graphs [34], often called *arenas*. The set of vertices of the graph is partitioned between the players. Players interact by moving a pebble from vertex to vertex, ad infinitum, following edges of the graph. The game starts in a given vertex, and the owner of the current vertex decides where to send the pebble next. The infinite path thereby created is called a *play*.

Objectives and Strategies. Each player has an *objective function*, which is essentially a measure of the utility of plays: players aim to optimise their utility. Players choose their actions (i.e., where to move the pebble) according to a *strategy*.

A key question in the field is to understand how complex strategies need to be for players to play optimally, when possible. In full generality, strategies may look at the current history – which may be arbitrarily long as we consider infinite-duration games – to pick an action. They may also use randomness, that is, prescribe a probability distribution over possible actions. However, in the simplest settings, this is superfluous. In particular, players may often resort to *pure memoryless* strategies, which simply associate a single action to each vertex, independently of what happened before.

When Are Complex Strategies Needed? Games on graphs and related models, such as Markov decision processes or stochastic games (both involving probabilistic arenas), have been studied extensively for decades, for a plethora of objectives [34]. An interesting line of work tries to understand, classify, and characterise broad classes of games and objectives for which strategies of restricted classes of complexity suffice to play optimally. One can notably mention [42] for (pure) memoryless (resp. [11] for finite-memory) strategies in two-player zero-sum games, [43] for (pure) memoryless (resp. [12] for finite-memory) strategies in two-player zero-sum *stochastic* games, [53] for finite-memory strategies when considering *combinations of objectives*, or [63] for *multi-player* games.

The Different Flavours of Complexity. As discussed above, the recent years have seen a surge in research to understand when (specific classes of) complex strategies are needed. Yet, one may take a step back and ask what does it really mean for a strategy to be complex? And what is the role of this complexity? With respect to memory, for example, we know that different models co-exist: see, e.g., the discussion on chromatic vs. chaotic memory in a recent survey [13]. Similarly, the various ways one may allow strategies to use randomness has a huge impact on the (expressive) power of strategies [56].

Formalising the nature of complexity is not an easy feat, and subject to almost-philosophical debates. What we aim to do in this work is a small step along the way: we will take a walk together among various game models and objectives, of (arguably) increasing complexity, and try to understand how this complexity percolates through the corresponding strategies, as well as on the computational standpoint. It is easy to get lost in the zoo of models and objectives that exist in the literature; to avoid that, we will focus on the familiar yet surprisingly interesting *reachability* objectives.

Reachability. In its simplest form, a reachability objective [34] simply asks to visit once any vertex belonging to a given target set. This objective is often considered to be the most elementary, and the main tool to solve reachability games, *attractors*, has made its way in virtually all algorithms on games on graphs. That is because, in one way or another, strategies for most game objectives do involve going from a point A to a point B as part of their inner mechanisms. This small part is exactly a reachability objective.

One may also note that the prevalence of reachability is not limited to games on graphs. Consider for example the effort put in understanding the reachability problem in Petri nets [54, 28], or in more distant models, the Skolem problem [74].

Outline. Our journey will go through reachability, generalised reachability (i.e., multiple target sets), shortest path (i.e., minimise the costs up to the first visit), repeated reachability (aka Büchi) objectives... and consider various models such as multi-player or stochastic games. As stated before, our goal is to illustrate how and when complexity arises, and what is the role of that complexity in strategies. To do so, we will work our way through increasingly complex models and objectives, grouped with respect to the resulting complexity of the corresponding strategies.

Our presentation is high-level and far from exhaustive: we do not aim to provide a fully-filled table of all possible combinations of models and objectives. Instead, we pick and choose particular settings that exemplify interesting phenomena. As such, this work should not be seen as a definitive overview of reachability games, but as a particular (and hopefully interesting) path through this crowded zoo.

2 Preliminaries

Notation. We let $\mathbb{R}$ (resp. $\mathbb{N}$, $\mathbb{Z}$) denote the set of real (resp. non-negative integer, integer) numbers and $\overline{\mathbb{R}} = \mathbb{R} \cup \{+\infty, -\infty\}$ (resp. $\overline{\mathbb{N}} = \mathbb{N} \cup \{+\infty\}$, $\overline{\mathbb{Z}} = \mathbb{Z} \cup \{+\infty, -\infty\}$). For any integer $k \geq 1$, we let $[\![k]\!] = \{1, \dots, k\}$.

Games. Let (V, E) be a directed graph where V is a finite set of vertices and $E \subseteq V \times V$ is a set of edges. An (n-player) *arena* (on (V, E)) is a tuple $\mathcal{A} = ((V_i)_{i \in [\![n]\!]}, E)$ where $[\![n]\!]$ is a set of n players and $(V_i)_{i \in [\![n]\!]}$ is a partition of V (between the different players). We assume that our arena is deadlock-free, i.e., for all $v \in V$, there exists $v' \in V$ such that $(v, v') \in E$. For $i \in [\![n]\!]$, we write $\mathcal{P}_i$ to denote the ith player.

Players take turns choosing the next vertex depending on which player controls the current vertex. This forms an infinite path in the arena called a *play*. Formally, a play in $\mathcal{A}$ is an infinite sequence of vertices $\rho = \rho_0 \rho_1 \dots$ such that for all $\ell \in \mathbb{N}$, $(\rho_\ell, \rho_{\ell+1}) \in E$. A history is a finite prefix of a play $h = h_0 \dots h_\ell$; we denote h_ℓ by $\mathsf{last}(h)$. The set of plays (resp. histories) is denoted by Plays (resp. Hist) and Hist_i is the set of histories ending in a vertex of $\mathcal{P}_i$. For $v \in V$, $\mathrm{Plays}(v)$, $\mathrm{Hist}(v)$, $\mathrm{Hist}_i(v)$ denote respectively the sets of plays, histories and histories ending in a vertex of $\mathcal{P}_i$ that begin in v. In an arena, it is common to fix an initial vertex $v_0 \in V$ and consider only plays and histories starting in v_0.

A (n-player) *game* $\mathcal{G} = (\mathcal{A}, (\mathrm{Obj}_i)_{i \in [\![n]\!]})$ is an arena equipped with a profile of *objective functions* $\mathrm{Obj} = (\mathrm{Obj}_i)_{i \in [\![n]\!]}$. For $i \in [\![n]\!]$, $\mathrm{Obj}_i : \mathrm{Plays} \to \overline{\mathbb{R}}$ measures the utility of each play of $\mathcal{A}$; $\mathcal{P}_i$ aims to optimise Obj_i. If $\mathcal{P}_i$ wants to maximise (resp. minimise) Obj_i, we say that Obj_i is a *gain* (resp. *cost*) *function* and we denote it by Gain_i (resp. Cost_i). In what follows, unless otherwise stated, we assume that the considered objective functions are gain functions.

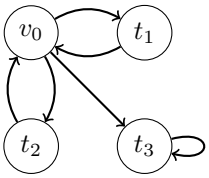
A two-player game $\mathcal{G} = (\mathcal{A}, (\mathrm{Obj}_1, \mathrm{Obj}_2))$ is *zero-sum* if, for all $\rho \in \mathrm{Plays}$, we have that[1] $\mathrm{Obj}_1(\rho) + \mathrm{Obj}_2(\rho) = 0$. Zero-sum games are often abusively denoted $\mathcal{G} = (\mathcal{A}, \mathrm{Obj}_1)$, as Obj_2 can be deduced from Obj_1. Given a play ρ, whenever Obj_1 represents a Boolean objective, i.e., Obj_1 has codomain $\{0, 1\}$, we say that $\mathcal{P}_1$ wins (resp. loses) if $\mathrm{Obj}_1(\rho) = 1$ (resp. 0). The Boolean designation is abusive for $\mathcal{P}_2$, as their gains range in $\{0, -1\}$.

Reachability Objective Functions. We now define the (generalised) reachability objective functions. Quantitative variants are discussed and defined in Section 3.3.

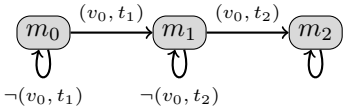
Let $\mathrm{T} \subseteq V$ be a target (set), the *reachability objective (for* T*)* consists in visiting T. Formally, we have that $\mathsf{Reach}[\mathrm{T}](\rho) = 1$, if there exists a position $\rho_\ell \in \mathrm{T}$, otherwise $\mathsf{Reach}[\mathrm{T}](\rho) = 0$. Let $\mathrm{T}_1, \dots \mathrm{T}_k \subseteq V$ be target sets. The *generalised reachability objective (for* $\mathrm{T}_1, \dots \mathrm{T}_k$*)* consists in visiting all the sets T_j in any order. Formally we have that $\mathsf{GReach}[\mathrm{T}_1, \dots, \mathrm{T}_k](\rho) = \prod_{j=1}^{k} \mathsf{Reach}[\mathrm{T}_j](\rho)$.

▶ Remark 1. We refer to games of the form $\mathcal{G} = (\mathcal{A}, (\mathsf{Reach}[\mathrm{T}_i])_{i \in [\![n]\!]})$ as *reachability games*. In the two-player zero-sum case, we speak of zero-sum reachability games if $\mathcal{P}_1$ has a reachability objective, although $\mathcal{P}_2$ does not have a reachability objective. We proceed similarly for the other considered objective functions.

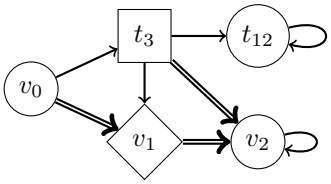
[1] If Obj_1 can take infinite values, we apply the convention $(+\infty) + (-\infty) = 0$.



(a) A one-player generalised reachability game where $\mathrm{T}_j = \{t_j\}$ for all $j \in [\![3]\!]$.



(b) An illustration of a Mealy machine update scheme. The next-move function of the machine is partly given by $\mathsf{nxt}(m_0, v_0) = t_1$, $\mathsf{nxt}(m_1, v_0) = t_2$ and $\mathsf{nxt}(m_2, v_0) = t_3$.



(c) A three-player reachability game with an NE that is not an SPE. Double arrows denote a (memoryless) strategy profile.

Figure 1 A one-player game (Figure 1a) with a Mealy machine encoding a strategy that is winning from v_0 (Figure 1b) and a game illustrating the difference between NEs and SPEs (Figure 1c).

Strategies. A *(pure) strategy* σ_i of $\mathcal{P}_i$, $i \in [\![n]\!]$, provides the next action of $\mathcal{P}_i$ based on what occurred previously. Formally, $\sigma_i \colon \mathrm{Hist}_i \to V$ is a function such that for all $h = h_0 \dots h_\ell \in \mathrm{Hist}_i$, $(h_\ell, \sigma_i(h)) \in E$. We denote by Σ_i the set of strategies of $\mathcal{P}_i$. We say that a play ρ is *played according to* σ_i if, whenever $\rho_\ell \in V_i$, we have that $\sigma_i(\rho_0 \dots \rho_\ell) = \rho_{\ell+1}$. We differentiate two classes of strategies: *memoryless* strategies and *finite-memory* strategies. A *memoryless* strategy σ_i only depends on the last vertex of the history, i.e., for all $h, h' \in \mathrm{Hist}_i$, if $\mathsf{last}(h) = \mathsf{last}(h')$, then $\sigma_i(h) = \sigma_i(h')$.

A strategy is *finite-memory* if it can be encoded by a finite-state Mealy machine. Formally, such a machine (for $\mathcal{P}_i$) is a tuple $\mathfrak{M} = (\mathrm{M}, m_0, \mathsf{up}, \mathsf{nxt})$ where M is a finite set of states, m_0 is an initial memory state, $\mathsf{up} \colon \mathrm{M} \times E \to \mathrm{M}$ is a memory update function and $\mathsf{nxt} \colon \mathrm{M} \times V_i \to V$ is a next-move function. Intuitively, the strategy encoded by a Mealy machine prescribes moves through the nxt function based on the current memory state and game vertex, and the memory state is updated at each round via the up function based on the traversed edge. In the following, the size of a memory for a strategy refers to an upper bound on the number of states of some Mealy machine that encodes the strategy.

Mealy machines combine memoryless strategies: a next-move function assigns a memoryless strategy to each memory state. In particular, memoryless strategies are finite-memory strategies that can be represented by Mealy machines with a single memory state.

▶ Remark 2. In some presentations, the update function of a Mealy machine is defined using vertices rather than edges. Memory bounds for strategies may vary depending on the model, e.g., when the objective function depends on edge-specific information (e.g., the shortest path games of Section 4.2). The vertex-update model is less compact in general.

A *strategy profile* $\sigma = (\sigma_i)_{i \in [\![n]\!]}$ assigns a strategy to each player. To highlight the role of $\mathcal{P}_i$, we sometimes write $\sigma = (\sigma_i, \sigma_{-i})$, where σ_{-i} denotes the strategy profile of the players other than $\mathcal{P}_i$. Once a strategy profile σ is fixed together with an initial vertex v_0, there exists a unique play starting from v_0 and played according to all strategies of σ. We denote this unique play by $\langle \sigma \rangle_{v_0}$ (or just $\langle \sigma \rangle$ when v_0 is clear from the context).

▶ **Example 3.** Let us consider the one-player game played on the arena given in Figure 1a with the objective $\mathsf{GReach}[\mathrm{T}_1, \mathrm{T}_2, \mathrm{T}_3]$ where $\mathrm{T}_j = \{t_j\}$ for all $j \in [\![3]\!]$ and the initial vertex v_0. We provide in Figure 1b a Mealy machine $\mathfrak{M}$ that encodes a strategy σ_1 with a memory of size 3. The three memory states are m_0, m_1 and m_2 and the memory update function is illustrated as follows: *(i)* a label $(v, v') \in E$ (resp. $\neg(v, v')$) on a transition $(m, m') \in \mathrm{M} \times \mathrm{M}$ represents $\mathsf{up}(m, (v, v')) = m'$ (resp. $\mathsf{up}(m, e) = m'$ for all $e \in E \setminus \{(v, v')\}$) and *(ii)* an unlabelled transition (m, m') means $\mathsf{up}(m, e) = m'$ for all $e \in E$. We consider the next-move function partly defined as $\mathsf{nxt}(m_{j-1}, v_0) = t_j$ for all $j \in [\![3]\!]$. The outcome of the strategy σ_1 is obtained as follows: the play begins in v_0 and the initial memory state is m_0, thus the first move prescribed by σ_1 is $\mathsf{nxt}(m_0, v_0) = t_1$. Once transition (v_0, t_1) is crossed, the

memory state is updated to m_1. Since from t_1 the only outgoing edge enters v_0, the current vertex is v_0 again and the memory state does not change. This time, from v_0, we move to $\mathsf{nxt}(m_1, v_0) = t_2$ and the memory state is updated to m_2. Continuing this way results in the outcome $\langle \sigma_1 \rangle = v_0 t_1 v_0 t_2 v_0 t_3^\omega$ which visits all targets.

Solution Concepts. Let $\mathcal{G} = (\mathcal{A}, (\mathrm{Gain}_i)_{i \in [\![n]\!]})$ be a game with an initial vertex v_0.

Given $\alpha \in \overline{\mathbb{R}}$, a strategy σ_i is said to *ensure* α (from v_0) if for all strategy profiles of the opponents σ_{-i}, we have that $\mathrm{Gain}_i(\langle \sigma_i, \sigma_{-i} \rangle) \geq \alpha$. A strategy σ_i is said to be *optimal (for* $\mathcal{P}_i$*)* if there exists $\alpha \in \overline{\mathbb{R}}$ such that σ_i ensures α and for all $\beta > \alpha$, no strategy of $\mathcal{P}_i$ ensures β. If such an α exists, it is called the value of vertex v_0 in the two-player zero-sum game $\mathcal{G}_i = (((V_i, V \setminus V_i), E), \mathrm{Gain}_i)$, called the *coalition game* for $\mathcal{P}_i$, in which $\mathcal{P}_i$ is opposed to the coalition of the other players.

In a two-player zero-sum (or one-player) game with Boolean objective functions, we say that σ_1 is a *winning strategy (for* $\mathcal{P}_1$*)* if σ_1 ensures 1. In this case, σ_1 is necessarily optimal.

A strategy profile σ is a *Nash equilibrium* (NE) if no player has an incentive to deviate unilaterally in order to increase his gain, i.e., for all $i \in [\![n]\!]$, for all $\sigma_i' \in \Sigma_i$, $\mathrm{Gain}_i(\langle \sigma_i', \sigma_{-i} \rangle) \leq \mathrm{Gain}_i(\langle \sigma \rangle)$. By essence, NEs do not take into account the sequential nature of games played on graphs. This implies existence of incredible threats in some NEs (see Example 4). The concept of *subgame perfect equilibrium* (SPE) proposes a classical remedy to this problem.

To formally define SPEs, we introduce the concept of subgame. Given a history $hv \in \mathrm{Hist}(v_0)$, the game $\mathcal{G}_{\restriction hv}$ with initial vertex v is called a *subgame* of $\mathcal{G}$ and is such that $\mathcal{G}_{\restriction hv} = (\mathcal{A}, (\mathrm{Gain}_{i \restriction hv})_{i \in [\![n]\!]})$ with, for all $i \in [\![n]\!]$ and for all plays $\rho \in \mathrm{Plays}(v)$, $\mathrm{Gain}_{i \restriction hv}(\rho) = \mathrm{Gain}_i(h\rho)$. Given a strategy of $\mathcal{P}_i$ in $\mathcal{G}$, $\sigma_{i \restriction hv}$ denotes its restriction in the subgame $\mathcal{G}_{\restriction hv}$ and is defined by, for all histories $h' \in \mathrm{Hist}_i(v)$, $\sigma_{i \restriction hv}(h') = \sigma_i(hh')$.

A strategy profile σ is an SPE in $\mathcal{G}$, if for all histories $hv \in \mathrm{Hist}(v_0)$, $\sigma_{\restriction hv} = (\sigma_{i \restriction hv})_{i \in [\![n]\!]}$ is an NE in $\mathcal{G}_{\restriction hv}$. Notice that, in particular, an SPE is an NE.

▶ **Example 4.** Let us consider the three-player reachability game played on the arena depicted in Figure 1c with initial vertex v_0. $\mathcal{P}_1$, $\mathcal{P}_2$ and $\mathcal{P}_3$ control circle, square and diamond vertices respectively and their targets are $\mathrm{T}_1 = \mathrm{T}_2 = \{t_{12}\}$ and $\mathrm{T}_3 = \{t_3\}$. A (memoryless) strategy profile σ is depicted by double arrows. We can prove that σ is an NE such that no player visits his target. However, it is not an SPE: $\mathcal{P}_2$ has an incentive to change his strategy in the subgame $\mathcal{G}_{\restriction v_0 t_3}$ by visiting $t_{12} \in \mathrm{T}_2$ instead of v_2. We define another strategy profile τ such that *(i)* $\tau_1(v_0) = t_3$ and $\tau_1(v) = v$ if $v \in \{v_2, t_{12}\}$, *(ii)* $\tau_2(t_3) = t_{12}$ and *(iii)* $\tau_3(v_1) = v_2$. This strategy profile is an SPE (thus also an NE) such that all players reach their target.

Studied Problems. We now define decision problems that arise naturally in our setting. In a one-player or two-player zero-sum setting, we study how well a player can perform.

▶ **Problem 1** (Threshold-ensuring strategy existence (TSE) problem)**.** *Given a two-player zero-sum (or one-player) game* $\mathcal{G} = (\mathcal{A}, \mathrm{Gain}_1)$ *with initial vertex* v_0 *and a threshold* $\alpha \in \overline{\mathbb{R}}$*, does there exist a strategy* σ_1 *of* $\mathcal{P}_1$ *such that* σ_1 *ensures* α*?*

We say that a strategy *solves* the problem if it provides a positive answer to the problem. When studying a two-player zero-sum (or one-player) game with a Boolean objective function, by *solving the game*, we mean deciding the existence of a winning strategy for $\mathcal{P}_1$, i.e., deciding the TSE problem for a threshold $\alpha = 1$.

Regarding equilibria, SPEs (and therefore NEs) are guaranteed to exist in all games with ω-regular Boolean objectives [69]; these games subsume all Boolean objectives considered in Section 3. SPEs exist also in all games with continuous gain functions [40]. These games

include shortest path games (see Section 3.3) with positive weights on the edges through an appropriate transformation [17]. Furthermore, we have seen in Example 4 that an NE where no player visits his target may coexist with another NE such that all players visit their targets, i.e., there is not a unique equilibrium gain profile in a given game. Therefore, instead of studying the existence of equilibria, which is often guaranteed, the following decision problem is considered.

▶ **Problem 2** (Constrained equilibrium existence (CEE) problem)**.** *Given an n-player game $\mathcal{G} = (\mathcal{A}, (\mathrm{Gain}_i)_{i\in[\![n]\!]})$ with initial vertex v_0 and a threshold tuple $\alpha \in \overline{\mathbb{R}}^n$, does there exist an equilibrium σ (NE or SPE) such that for all $i \in [\![n]\!]$, $\mathrm{Gain}_i(\langle\sigma\rangle) \geq \alpha_i$?*

We say that a strategy profile *solves* the CEE problem if it provides a positive answer to it. When studying an n-player game with Boolean objective functions, we only consider threshold tuples in $\{0,1\}^n$.

SAT, QBF and Generalised Reachability. We consider two classical problems: the *Boolean satisfiability (SAT) problem* and the *quantified Boolean formula (QBF) problem*, and their relations with generalised reachability. Although generalised reachability in the two-player zero-sum setting is discussed in Section 3.1.3, we present a hardness argument and an algorithm to solve such games below, as we refer to these throughout Section 3.1.

The QBF problem asks whether a fully quantified formula (in conjunctive normal form) is true or false. To be more specific, let us consider finitely many variables $x_1, \ldots, x_m$. A literal is a variable x_i or its negation $\neg x_i$. A clause C is a finite disjunction of literals. Let Q denote a quantifier, i.e., $Q \in \{\exists, \forall\}$. The QBF problem asks whether a formula of the form $\psi = Q_1 x_1\ Q_2 x_2\ \ldots\ Q_m x_m\ C_1 \wedge\ C_2 \wedge \ldots \wedge C_k$ is true.

It is classical to view the QBF problem as reachability in an And-Or graph (exponential in the size of the formula) [60]. Equivalently, one can also solve the QBF problem via a two-player zero-sum game (linear in the size of the formula) with a generalised reachability objective [35, 36]. We describe a generic construction to derive a graph from the formula ψ. For each quantifier Q_i and corresponding bound variable x_i, we introduce three vertices Q_i, x_i and $\neg x_i$ with edges from Q_i to the other two vertices and, if $i < m$, edges from x_i and $\neg x_i$ to Q_{i+1}. The resulting graph is represented in the small dashed rectangle $\mathbf{QBF_1}$ of Figure 2. We postpone the explanation of the whole figure $\mathbf{QBF_2}$; it is used in a specific reduction in Section 3.1.3. We say that vertex Q_i is an existential (resp. universal) vertex if Q_i is an existential (resp. universal) quantifier. For each clause C_ℓ of the formula, we introduce a target T_ℓ consisting of the vertices that are literals occurring in C_ℓ.

When reducing the QBF instance given by ψ to a two-player zero-sum generalised reachability game, we assign existential and universal vertices in the graph above to $\mathcal{P}_1$ and $\mathcal{P}_2$ respectively. The objective of $\mathcal{P}_1$ is given by $\mathsf{GReach}[\mathrm{T}_1, \ldots, \mathrm{T}_k]$. Intuitively, the players, through their strategies, select the assignments of the variables bound to their quantifiers in the order of quantification and a clause holds if its matching target is visited. It follows that there is a winning strategy for $\mathcal{P}_1$ if and only if there is a way to assign existentially quantified variables such that, no matter how universally quantified variables are assigned, all clauses hold, i.e., if ψ holds [35, 36]. As the QBF problem is PSPACE-complete [60], the above reduction directly implies that solving two-player zero-sum games with generalised reachability objectives is PSPACE-hard.

One can prove that the problem of solving generalised reachability games is actually PSPACE-complete, following the lines of [35, 36]. It can be solved via an alternating polynomial time algorithm. The algorithm relies on the fact that there exists a winning strategy for $\mathcal{P}_1$ if and only if there is one that visits all targets within at most $k \cdot |V|$ steps. To

decide the winner, one simulates plays by choosing the next state existentially or universally depending on the player and until the previous bound is reached. Although the step counter only requires space logarithmic in $k \cdot |V|$, the algorithm does not use logarithmic space overall as it must keep track of the set of visited targets along a play.

The SAT problem, which is known to be NP-complete [26], can be seen as the restriction of the QBF problem where all the quantifiers are existential. The SAT problem can be reduced to solving a one-player generalised reachability game via the previously described reduction. It follows that solving one-player generalised reachability games is NP-hard [35, 36].

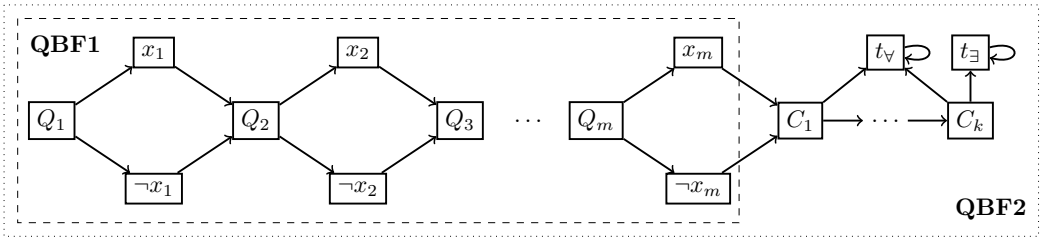


Figure 2 A graph derived from the formula $\psi = Q_1 x_1\ Q_2 x_2\ \ldots\ Q_m x_m\ C_1 \wedge\ C_2 \wedge \ldots \wedge C_k$. For $\ell \in [\![k]\!]$, all the literals of the clause C_ℓ are encoded in a target T_ℓ. For instance, given the clause $x_1 \vee \neg x_2 \vee x_3$, the associated target is $\{x_1, \neg x_2, x_3\}$.

3 The Structure and Properties of Memory

In this section, we overview strategies in games with reachability-related objectives. First, we consider Boolean objectives. We present (generalised) reachability in Section 3.1. In Section 3.2, we discuss games in which targets have to be visited infinitely often. Finally, in Section 3.3, we consider a quantitative variant of reachability, where we aim at reaching the target(s) with minimal cost. The problems in each section are presented in increasing order of complexity required by strategies that solve the problems. The complexity of a strategy refers not only to its memory size but also to the variety of the behaviours it encodes.

3.1 Boolean Reachability

We focus here on (generalised) reachability objectives. The section is divided into three parts. Section 3.1.1 is dedicated to problems where memoryless strategies suffice. In Section 3.1.2, we consider problems where it suffices to use a memory of polynomial size that encodes a (fixed) sequence representing the order in which targets are to be seen. Finally, in Section 3.1.3, we deal with problems where a memory that branches based mainly on the set of visited targets suffices. In this latter case, an exponential memory size may be required.

3.1.1 Memoryless Strategies

There are two simple situations where no memory is needed: (*i*) one-player games with reachability objectives and (*ii*) two-player zero-sum reachability games. In both settings, despite the Boolean nature of the objectives, a natural winning strategy consists in reaching the target as soon as possible, i.e., by minimising the length of the path from the initial vertex to the target.

One-player Reachability Games. A one-player game with a reachability objective is nothing but a connectivity problem in a graph (see [73] and [1] for surveys). If there is a path from an initial vertex to the target, there is a *simple path*, thus no memory is needed. Several algorithms have exploited this idea in order to solve the shortest-path(s) problem.[2]

As pointed in [1], the problem of finding a path from one vertex to another in a directed graph is the first problem that was identified as being complete for a natural subclass of P; it was shown to be NL-complete [49].

Two-player Zero-sum Reachability Games. Let $\mathcal{P}_1$ be the player with the reachability objective. In this two-player zero-sum framework, although it is not sufficient for $\mathcal{P}_1$ to exhibit a simple path from the initial vertex to the target, the resolution has the same flavour as the one-player case. Indeed, if $\mathcal{P}_2$, whose objective is to avoid the target, can force a loop before the target is reached, then $\mathcal{P}_2$ can avoid it forever. In particular, if $\mathcal{P}_1$ has a winning strategy, he can force a visit to the target via a simple path no matter the behaviour of $\mathcal{P}_2$. In other words, as shown in [66, 44], memoryless strategies are sufficient for $\mathcal{P}_1$ (the same holds for $\mathcal{P}_2$).

It can be shown that solving zero-sum reachability games is P-hard, even in acyclic graphs [48]. Furthermore, the alternating algorithm previously described for generalised reachability objectives runs in alternating logarithmic space when there is only one target. This implies the problem is in P, i.e., deciding the TSE problem for two-player zero-sum reachability games is P-complete.

The classical algorithm to solve two-player zero-sum reachability games is the attractor algorithm [66]. Starting from the target, it computes, layer by layer, the set of vertices from which $\mathcal{P}_1$ can win. In step j of the computation, the algorithm computes the vertices from which $\mathcal{P}_1$ can force a visit to the target in at most j steps.

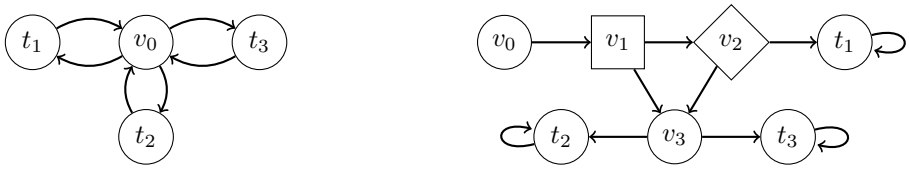
3.1.2 Sequential-memory Strategies

We now consider two situations where a memory of polynomial size is necessary: (i) one-player generalised reachability games and (ii) NEs in n-player reachability games. In both cases, the structure of the memory encodes the (fixed) order in which the targets are to be visited, and its role is to remember previously visited targets. This explains the *sequential-memory* designation: there is only one maximal chain of sets of visited targets compatible with the strategy (profile). In the case of NEs, additional memory is needed to ensure stability, via the so-called classical mechanism of *punishment*.

One-player Generalised Reachability Games. Solving a one-player game with a generalised reachability objective equates to finding a path that visits all targets in the underlying graph. The game illustrated by Figure 3a, with three targets $\mathrm{T}_j = \{t_j\}$ for $j \in [\![3]\!]$, provides a simple situation where memoryless strategies are not sufficient. Indeed, starting from v_0, any memoryless strategy visits only one target. To ensure a visit to all targets, one should first order them, for instance $(\mathrm{T}_1, \mathrm{T}_2, \mathrm{T}_3)$.

In the present example, any order works. However, this is not the case in the game illustrated in Figure 1a, in which target $\mathrm{T}_3 = \{t_3\}$ must be the last to be reached. Once the order is fixed, the memory indicates the next target to be reached (via a memoryless strategy). In general, a memory of size k, encoded via a Mealy machine similar to the one in Figure 1b, is sufficient to win in a one-player game with objective $\mathsf{GReach}[\mathrm{T}_1, \ldots, \mathrm{T}_k]$.

[2] Let us cite Bellman-Ford algorithm [38, 6], Dijkstra's algorithm [31], the Floyd–Warshall algorithm [37, 72, 64] (also used to compute the transitive closure of a graph), and the closely related Kleene's algorithm used to convert a deterministic finite automaton into a regular expression [51].



(a) A one-player generalised reachability game where a sequential-memory strategy suffices to win.

(b) A three-player reachability game where punishment memory is needed to ensure NE stability.

Figure 3 Examples where memoryless strategies do not suffice.

From the discussion about SAT in Section 2, we already know that solving one-player generalised reachability games is NP-hard. In fact, the problem is NP-complete: the alternating algorithm proposed in [35, 36] to solve two-player zero-sum generalised reachability games (cf. Section 2) uses only non-determinism, not alternation, when applied to one-player games, providing the desired result.

Nash Equilibria in n-player Reachability Games. We now consider the CEE problem for NEs. Strategy profiles that solve this problem in n-player reachability games may require memory, the role of which is two-fold. The first role matches that of a sequential memory. Consider the arena illustrated by Figure 3a, interpreted as a three-player game where $\mathcal{P}_1$ controls all vertices and the targets of the players are given by $\mathrm{T}_i = \{t_i\}$ for $i \in [\![3]\!]$. Memory is necessary to obtain an NE from v_0 that visits all targets, as described in the one-player generalised reachability setting.

The second role of memory is to ensure the stability of the equilibrium. Consider the three-player reachability game on the arena of Figure 3b. $\mathcal{P}_1$, $\mathcal{P}_2$ and $\mathcal{P}_3$ control circle, square and diamond vertices respectively and the target of $\mathcal{P}_i$ is $\mathrm{T}_i = \{t_i\}$. An NE from v_0 such that $\mathcal{P}_1$ wins requires memory. The only play that satisfies this requirement is $v_0v_1v_2t_1^\omega$. However, $\mathcal{P}_1$ cannot ensure the stability of this outcome with a memoryless strategy: from v_3, $\mathcal{P}_1$ has to remember if $\mathcal{P}_2$ (resp. $\mathcal{P}_3$) tried to deviate to v_3 and has to choose t_3 (resp. t_2) to avoid $\mathcal{P}_2$ (resp. $\mathcal{P}_3$) from having a profitable deviation. Let σ_1 denote the latter strategy of $\mathcal{P}_1$ and σ_2 (resp. σ_3) the memoryless strategy of $\mathcal{P}_2$ (resp. $\mathcal{P}_3$) defined by $\sigma_2(v_1) = v_2$ (resp. $\sigma_3(v_2) = t_1$). One can check that $(\sigma_1, \sigma_2, \sigma_3)$ is an NE.

In general, outcomes of NEs can be characterised using the values of the coalition games $\mathcal{G}_i$ for each $i \in [\![n]\!]$. A play ρ is an outcome of an NE if and only if, for each player i who loses along ρ, the value of ρ_ℓ in $\mathcal{G}_i$ for $\mathcal{P}_i$ is 0 for all $\ell \in \mathbb{N}$. Any NE outcome satisfies this characterisation as otherwise a losing player would have a profitable deviation by playing his winning strategy in his coalition game $\mathcal{G}_i$ from a vertex with a value of 1 in $\mathcal{G}_i$. Conversely, the following strategy profile is an NE with outcome ρ: all players follow ρ and play an optimal strategy in $\mathcal{G}_i$ if $\mathcal{P}_i$ deviates from ρ, i.e., the coalition *punishes* $\mathcal{P}_i$. This behaviour ensures the stability of the equilibrium. This construction is inspired by the proof of the folk theorem in repeated games [39, 59]. In the NE in the game of Figure 3b defined above, from v_3, $\mathcal{P}_1$ punishes $\mathcal{P}_2$ (resp. $\mathcal{P}_3$) if $\mathcal{P}_2$ (resp. $\mathcal{P}_3$) deviates to v_3 by using his optimal strategy in $\mathcal{G}_2$ (resp. $\mathcal{G}_3$).

Using the characterisation above, we can show that, from any NE outcome, we can derive an NE outcome with a simple structure and the same visited targets. These outcomes are such that the path between two consecutive targets is simple and the last target is followed by a lasso hc^ω such that hc is a simple history. It follows that such outcomes have a representation of polynomial size.

From these simple outcomes, we can design a strategy profile solving the CEE problem for NEs with memory size quadratic in the number of players (and independent of the size of the arena) as follows [55]. Memory states keep track of the next target vertex and who last controlled a vertex of the (simple) path to it. If this path is exited by $\mathcal{P}_i$, then the memory detects $\mathcal{P}_i$ deviated and is updated to its $\mathcal{P}_i$ punishment state; the coalition of other players punish $\mathcal{P}_i$ using a *memoryless* punishing strategy. We remark that if $\mathcal{P}_i$ deviates without exiting the path to the next target, then it is not necessary for the other players to punish $\mathcal{P}_i$ and they continue to try progressing along the path to the next target. This *punishment memory* is not required for players who see their target, as there can be no profitable deviations for them. In particular, if all players see their target, then no punishment memory is needed and a memory size matching the number of players is sufficient as in one-player generalised reachability games. Furthermore, it is known that in every n-player reachability game there exists an NE with memory at most linear in the size of the game [19].

The CEE problem for NEs is NP-complete [14, 18]. The characterisation of NE outcomes and the existence of polynomial-size lasso outcomes above imply that the CEE problem for NEs in n-player reachability games is in NP. Hardness can be recovered from the NP-completeness of the TSE problem in one-player generalised reachability games. The player wins the one-player generalised reachability game with objective $\mathsf{GReach}[\mathrm{T}_1, \ldots, \mathrm{T}_n]$ if and only if there exists an NE such that all players win in the n-player reachability game on the same arena such that $\mathcal{P}_i$, $i \in [\![n]\!]$, has T_i as target.

3.1.3 Branching-memory Strategies

We now consider two situations where a memory of exponential size is necessary: *(i)* two-player zero-sum generalised reachability games and *(ii)* SPE in n-player reachability games. Again, the memory is used to remember the set of visited targets. In the two-player setting with objective $\mathsf{GReach}[\mathrm{T}_1, \ldots, \mathrm{T}_k]$, the memory of $\mathcal{P}_1$ needs to adapt to the choices of $\mathcal{P}_2$. This leads to an exponential blow up in the memory as the whole lattice of subsets of $[\![k]\!]$ has to be remembered. This differs from the one-player setting discussed in Section 3.1.2, where only a chain of the lattice was sufficient. Regarding SPE, players may have to adapt their behaviour depending on the different subgames. These phenomena motivate the name of *branching memory*.

Two-player Zero-sum Generalised Reachability Games. Solving a two-player zero-sum game with a generalised reachability objective is more involved than one could expect [36]. From the discussion about QBF in Section 2, we know that the problem is PSPACE-complete. To exhibit a game where an exponential memory is needed for $\mathcal{P}_1$, we rely on the reduction from the QBF problem to two-player zero-sum generalised reachability games presented in Section 2 (involving the graph depicted in $\mathbf{QBF}_1$ from the small dashed rectangle of Figure 2). We consider the two-player game derived, via this reduction, from the formula:

$$\psi = \forall y_1 \forall y_2 \ldots \forall y_k \; \exists z_1 \exists z_2 \ldots \exists z_k \; \bigwedge_{i=1}^{k} (y_i \vee z_i) \wedge (\neg y_i \vee \neg z_i). \tag{1}$$

We first note that ψ holds. Thus, $\mathcal{P}_1$ (the existential player) has a winning strategy to visit all the targets. To win, $\mathcal{P}_1$ has to assign, to each variable z_i, the negation of the assignment of y_i. A *branching* memory of size 2^k is thus necessary to react to all potential assignments of $\mathcal{P}_2$. In particular, the order in which $\mathcal{P}_1$ visits the targets cannot be fixed a priori.

Notice that all the clauses of the formula ψ from (1) are of size 2. From [4], we know that such a QBF instance can be solved in linear time.[3] At first glance, this could look paradoxical with the memory requirement of $\mathcal{P}_1$ above. But as zero-sum generalised reachability games are determined, i.e., one of the two players has a winning strategy from any initial vertex, we can adopt the point of view of $\mathcal{P}_2$. From [4, Theorem 2], we can deduce that when a formula with at most two literals per clause is not satisfied, then $\mathcal{P}_2$ has a winning strategy in the corresponding game with memory of size at most 2.

In the general setting of two-player zero-sum generalised reachability games, both players may require memory of exponential size. Tight memory bounds are provided in [35, 36]. Regarding computational complexity, we note that PSPACE-hardness is only known when targets contain at least three vertices. The particular case where each target is composed of only one vertex (resp. two vertices) is solvable in polynomial time (resp. is still an open problem) [36].

Subgame Perfect Equilibria in n-player Reachability Games. Since SPEs are NEs, strategy profiles that solve the CEE problem for SPEs in n-player reachability games are no less complex than those needed for NEs. In particular, punishment memory may be necessary. Furthermore, due to the fact that a player should act rationally in all subgames, *branching memory* is needed to choose the next action, and may be more sophisticated than for NEs. It may be exponential for reasons similar to two-player zero-sum generalised reachability games.

In this setting, the CEE problem is PSPACE-complete [16]. The PSPACE membership relies on the following property: if there exists an SPE that solves the problem, then there exists an SPE that solves the problem and has a finite representation. This representation is a set of lassoes of polynomial length, mainly one per element of $[\![n]\!] \times V \times 2^n$, that satisfy some property on the players' gains. This characterisation allows the design of an alternating algorithm that runs in polynomial time. More details can be found in [16].

PSPACE-hardness is due to a reduction from the QBF problem (explained in Section 2). From a quantified formula $\psi = Q_1 x_1\ Q_2 x_2\ \dots\ Q_m x_m\ C_1 \wedge\ C_2 \wedge \dots \wedge C_k$ we build a $(k+2)$-player reachability game on the graph illustrated by the big dashed rectangle ($\mathbf{QBF_2}$) of Figure 2. In addition to the structure of $\mathbf{QBF_1}$, we consider one additional vertex per clause and two more vertices $t_\forall$ and $t_\exists$, linked by the edges depicted in the figure. This is a $(k+2)$-player reachability game with initial vertex Q_1 such that $\mathcal{P}_i$, $i \in [\![k]\!]$ represents the clause C_i, controls the vertex C_i and wants to visit a literal of his clause or vertex $t_\forall$. The other two players are $\mathcal{P}_\exists$ and $\mathcal{P}_\forall$: $\mathcal{P}_\exists$ (resp. $\mathcal{P}_\forall$) controls existential (resp. universal) vertices and wants to reach $t_\exists$ (resp. $t_\forall$). In an equilibrium, $\mathcal{P}_\exists$ cannot win if some clause C_i is not satisfied since $\mathcal{P}_i$ should move to $t_\forall$ from vertex C_i. Moreover, $\mathcal{P}_\exists$ wins if and only if $\mathcal{P}_\forall$ loses. With this construction, it can be shown that ψ is true if and only if there exists an SPE such that all players win except $\mathcal{P}_\forall$. Notice that this reduction is no longer correct if we replace SPE by NE: the subgame rationality imposed by SPEs plays a crucial role. Indeed, let us consider the formula $\psi = \forall x_1 \exists x_2 (x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2)$ and the memoryless strategy profile σ such that $\sigma_\forall(\forall_1) = \neg x_1$, $\sigma_\exists(\exists_2) = x_2$, $\sigma_1(C_1) = C_2$, $\sigma_2(C_2) = C_3$ and $\sigma_3(C_3) = t_\exists$. We can prove that σ is an NE such that all players but $\mathcal{P}_\forall$ win, although ψ is false. In particular, $\mathcal{P}_\forall$ has no incentive to move towards x_1 since, even if $\mathcal{P}_2$ does not visit his target along this outcome, he still chooses to move towards $t_\exists$. However,

[3] The algorithm is based on a smart analysis of the strongly connected components of a directed graph associated with the formula.

this latter behaviour is not rational in the subgames induced by the $\mathcal{P}_\forall$'s choice to go to x_1, since $\mathcal{P}_2$ should choose $t_\forall$ and win. Thus σ is not an SPE. Finally, remark that deciding if there exists an SPE such that all players win is easier; it amounts to deciding if there exists a winning strategy for the player of the one-player generalised reachability game with objective $\mathsf{GReach}[\mathrm{T}_1, \ldots, \mathrm{T}_n]$. That can be done with a nondeterministic polynomial time algorithm.

Thanks to the finite representation of an SPE mentioned before, it can be deduced from [16] that strategy profiles with memory size exponential in the number of players and polynomial in the size of the arena are sufficient to solve the CEE problem for SPEs.

3.2 Repeated Reachability

We now consider the (generalised) Büchi objective functions. Let $\mathrm{T} \subseteq V$ be a target, the *Büchi objective (for* T*)* consists in visiting T infinitely often. Formally we have that $\mathsf{Büchi}[\mathrm{T}](\rho) = 1$, if for all $j \in \mathbb{N}$ there exists a position $\rho_\ell \in \mathrm{T}$ with $\ell > j$, otherwise $\mathsf{Büchi}[\mathrm{T}](\rho) = 0$. Let $\mathrm{T}_1, \ldots \mathrm{T}_k \subseteq V$ be target sets. The *generalised Büchi objective (for* $\mathrm{T}_1, \ldots \mathrm{T}_k$*)* consists in visiting all the sets T_j infinitely often. Formally we have that $\mathsf{GBüchi}[\mathrm{T}_1, \ldots, \mathrm{T}_k](\rho) = \prod_{j=1}^{k} \mathsf{Büchi}[\mathrm{T}_j](\rho)$.

This section follows a line of progression similar to that of Section 3.1.

3.2.1 Memoryless Strategies

As in Section 3.1.1, memoryless strategies are sufficient in two situations: (*i*) one-player games with Büchi objectives and (*ii*) two-player zero-sum Büchi games. These memoryless strategies are essentially reachability strategies for targets that can be visited infinitely often.

3.2.2 Looping-sequential-memory Strategies

Let us now consider (*i*) one-player generalised Büchi games and (*ii*) NEs in n-player Büchi games, the counterparts of the games studied in Section 3.1.2, and (*iii*) two-player generalised Büchi games, the counterpart of the games studied in the first part of Section 3.1.3. For (*i*) and (*ii*), the required memory is similar to the reachability case. Indeed, in the two examples of Figure 3 the (generalised) reachability objectives can be translated into (generalised) Büchi objectives without affecting the memory requirements of the players. Although Büchi objectives appear more complex than reachability objectives, in both cases, the decision problems are simpler and can be solved in polynomial time, whereas the reachability variants are NP-complete. For (*iii*), the decision problem can also be solved in polynomial time, with a memory of polynomial size, even though the generalised reachability variant is PSPACE-hard and requires strategies with exponential memory.

One-player Generalised Büchi Games. Solving a one-player game with a generalised Büchi objective $\mathsf{GBüchi}[\mathrm{T}_1, \ldots, \mathrm{T}_k]$ amounts to finding a reachable strongly connected component (SCC) which contains elements of each target $\mathrm{T}_1, \ldots, \mathrm{T}_k$. This can be done in polynomial time [65]. The structure of the memory encodes a (fixed) order in which the targets have to be repeatedly visited. We speak here of *looping sequential memory* as the reachability scheme has to be repeated. As an example, let us consider the one-player game depicted on Figure 3a with objective $\mathsf{GBüchi}[\{t_1\}, \{t_2\}, \{t_3\}]$. The memory structure of a winning strategy can be derived from the Mealy machine of Figure 1b with an additional transition from m_2 to m_0 when reading edge (v_0, t_3) to complete the loop.

An alternative polynomial time algorithm to solve a one-player generalised Büchi game is to transform it into a one-player (simple) Büchi game of polynomial size and then solve this Büchi game. Given a game with objective $\mathsf{GBüchi}[\mathrm{T}_1, \ldots, \mathrm{T}_k]$, the Büchi game is obtained by

juxtaposing k copies of the original game. We move from the jth copy to the $(j+1)$th copy (when $j \neq k$) or to the first (when $j = k$) whenever T_j is visited within the jth copy. The (simple) Büchi condition on this expanded game consists in visiting infinitely often the first copy of T_1. Details of the construction and its correctness can be found in [5, Theorem 4.56].

Such an elegant *polynomial-size* construction would not work for a generalised reachability objective. This construction relies on the fact that if the objective $\mathsf{GBüchi}[\mathrm{T}_1, \dots, \mathrm{T}_k]$ is satisfied, we can visit targets in any order as there are vertices of each target that are connected. In particular, it is suitable to first visit T_1, then T_2 and so on. By contrast, recall that enforcing $\mathsf{GReach}[\mathrm{T}_1, \dots, \mathrm{T}_k]$ may require visiting targets in a specific order (see Example 3). The analogue of the construction above for generalised reachability games is to include the sets of previously visited targets in the vertices of the arena, which comes at an exponential price [35, 36].

Equilibria in n-player Büchi Games. Let us now turn to NEs in n-player Büchi games. In this setting, a polynomial-size memory is sufficient [70]. Once more, SCCs play a crucial role to obtain a polynomial algorithm. A deterministic polynomial algorithm is presented in [70]. This algorithm relies on a characterisation of outcomes of NEs as explained for n-player reachability games and on an algorithm tailored for the Büchi objective that finds an adequate reachable SCC in refined arenas of the game.

In contrast to the NE case, memory requirements as well as the exact complexity class of the CEE problem for SPEs in n-player Büchi games are still open. Nevertheless, this latter problem is in NP since it is the case for the more general parity games [15].

Two-player Generalised Büchi Games. Two-player zero-sum generalised Büchi games can also be solved in polynomial time and a memory of polynomial size is sufficient [32, 23]. The construction discussed above for one-player generalised Büchi games can be adapted to the two-player zero-sum setting. As memoryless strategies suffice to win in (simple) Büchi games, it follows from the construction that a polynomial-size memory suffices to win in generalised Büchi games. The role of the memory is to ensure that T_1 is visited, then T_2 until T_k and then restart from T_1.

3.3 Reaching Targets Quickly: Shortest-path Games

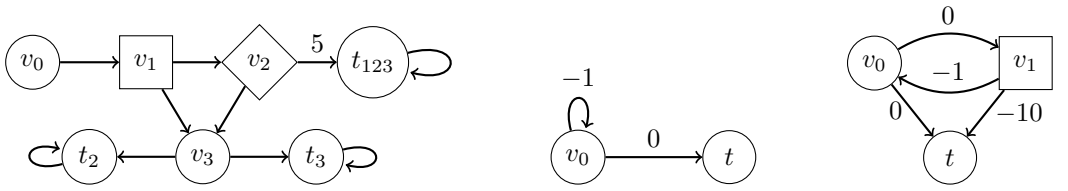
In this section, we focus on reaching a target as soon as possible. In order to formalise this idea, we first need to enrich our arenas with weights on edges. A *weighted arena* is couple $(\mathcal{A}, w)$ where $\mathcal{A}$ is an arena and $w \colon E \to \mathbb{Z}$ is a weight function.

Let $\mathrm{T} \subseteq V$ be a target. We introduce the *shortest-path objective* described by the cost function $\mathsf{SPath}_w[\mathrm{T}]$ that players intend to minimise. It is a quantitative variant of the Boolean reachability objective function. Formally, $\mathsf{SPath}_w[\mathrm{T}] : \mathrm{Plays} \to \overline{\mathbb{Z}}$ is defined by:

$$\mathsf{SPath}_w[\mathrm{T}](\rho) = \begin{cases} \sum_{j=0}^{\ell-1} w(\rho_j, \rho_{j+1}) & \text{if } \ell \text{ is the least index such that } \rho_\ell \in \mathrm{T} \\ \infty & \text{otherwise.} \end{cases}$$

Definitions using gain functions can be recovered for cost functions by reversing inequalities (e.g., solution concepts and the CEE problem).

In Section 3.3.1, we study shortest-path games whose weight function is restricted to non-negative weights, while Section 3.3.2 considers unrestricted weights.



(a) A game where punishment memory is needed even if all targets are visited.

(b) A one-player game where any cost can be ensured using step-counting memory.

(c) A game where a step-counting memory suffices to play optimally.

Figure 4 Weighted arenas for shortest-path games. Unlabelled edges have a weight of 1.

3.3.1 Same Strategies as the Boolean Case: Non-negative Weights

First, we consider the case where $w\colon E \to \mathbb{N}$. In a one-player or two-player zero-sum setting, there is always a memoryless optimal strategy in shortest-path games. The memory needed to solve the natural shortest-path variants of the CEE problem differs only slightly from the Boolean case. Namely, for NEs, punishment memory could be required even when all players visit their target.

Let us consider the three-player shortest-path game played on the arena of Figure 4a with initial vertex v_0. $\mathcal{P}_1$, $\mathcal{P}_2$ and $\mathcal{P}_3$ control circle, square and diamond vertices respectively and their targets are $\{t_{123}\}$, $\{t_2, t_{123}\}$ and $\{t_3, t_{123}\}$. To obtain an NE visiting targets of all players, $\mathcal{P}_1$ needs memory. Indeed, the only possible such outcome is $v_0v_1v_2t_{123}^\omega$ with a cost of 7. However, if $\mathcal{P}_1$ uses a memoryless strategy and chooses t_2 (resp. t_3) from v_3 then, $\mathcal{P}_3$ (resp. $\mathcal{P}_2$) should deviate towards v_3 to obtain a cost of 3. Punishment memory, matching that required in the game illustrated in Figure 3b, is needed.

One could say that nothing differs in terms of computational complexity for the TSE problem in shortest-path one-player or two-player zero-sum games (e.g., [31, 52, 50, 21]). In general, the philosophy of the algorithms remains the same, as the algorithms for the Boolean case already compute shortest paths in some sense.

Let us now turn to the shortest-path variants of the CEE problem. In this case, nothing differs in terms of complexity classes. The NP membership for NEs in n-player shortest-path games can be derived from the arguments used in the Boolean case. As the hardness argument in the Boolean setting of one-player generalised reachability games is not applicable in this case, NP-hardness relies on a reduction from the SAT problem. This reduction is in the same spirit as the construction presented for SPEs but restricted to existential formulas [18]. To obtain PSPACE-completeness for SPEs in n-player shortest-path games, non-trivial adaptations of the Boolean case are needed (specifically for PSPACE membership), see [17].[4]

We now mention another setting where an exponential memory size is needed. In the Boolean setting, the simplest such case is when solving two-player zero-sum generalised reachability games. We present here a multi-dimensional variant of the TSE problem where an exponential size memory is needed even with a single target T.

We consider a two-player arena equipped with a multi-dimensional weight function $w\colon E \to \mathbb{N}^k$. The (multi-dimensional) cost function of $\mathcal{P}_1$ is such that, for all plays $\rho \in \mathrm{Plays}$, $\mathsf{SPath}_w[\mathrm{T}](\rho) = (\mathsf{SPath}_{w_1}[\mathrm{T}](\rho), \dots, (\mathsf{SPath}_{w_k}[\mathrm{T}](\rho))$. Given $\alpha \in \mathbb{N}^k$, we study the existence of a strategy σ_1 of $\mathcal{P}_1$ such that for all σ_2 of $\mathcal{P}_2$, $\mathsf{SPath}_{w_j}[\mathrm{T}](\langle \sigma_1, \sigma_2 \rangle) \leq \alpha_j$ for each $j \in [\![k]\!]$.

[4] Notice that in [17], authors assume that $w(e) = 1$ for all $e \in E$.

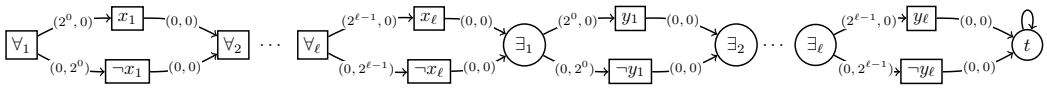


Figure 5 A multi-dimensional shortest-path game where an exponential memory size is required.

Consider the game played on the arena depicted in Figure 5 where $\mathcal{P}_1$ (resp. $\mathcal{P}_2$) controls circles (resp. squares) and $\mathrm{T} = \{t\}$. This illustrates a situation where exponential memory is needed: a strategy that solves the game for the threshold $(2^\ell - 1, 2^\ell - 1)$ requires a memory with size exponential in the number of vertices. Intuitively, $\mathcal{P}_1$ has to remember the choice of $\mathcal{P}_2$ in vertex $\forall_j$ and do the opposite in $\exists_j$ for all $j \in [\![\ell]\!]$. This multi-dimensional version of the TSE problem is PSPACE-complete [22]. The PSPACE-hardness, proved via a reduction from the *quantified subset-sum problem* [67, Lemma 4], holds even if $|\,\mathrm{T}\,| = 1$ and $k = 2$.

3.3.2 Step-counting-memory Strategies: Unrestricted Weights

We now turn to the case where $w\colon E \to \mathbb{Z}$. In this setting, complex phenomena already occur in one-player shortest-path games. We consider the one-player shortest-path game on the arena depicted in Figure 4b with target $\{t\}$. There is no optimal strategy to reach t from v in this game. Indeed, given any strategy σ_1 ensuring the cost α, there exists a *better* strategy σ_1' ensuring $\alpha - 1$ (remember that we aim at minimising the cost). For a given threshold α, there exists a strategy ensuring α, but this strategy needs memory. The memory is used to count the number of loops to take before moving to t.

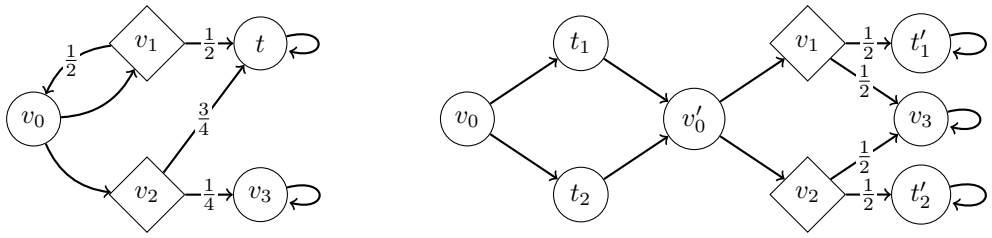
We call strategies that follow a memoryless strategy that accumulates negative weight for a fixed number of steps before switching to a memoryless shortest-path strategy a strategy with *step-counting memory*. If there exists a strategy ensuring a threshold α (encoded in binary), then there is a strategy with step-counting memory that ensures α with a memory of pseudo-polynomial size, i.e., of size exponential in the size of the encoding of α (recall that the size of the memory refers to the size of the Mealy machine).

When considering two-player zero-sum shortest-path games, similar, but slightly more subtle phenomena occur. Let us consider the shortest-path game on the arena of Figure 4c where $\mathcal{P}_1$ (resp. $\mathcal{P}_2$) controls the circle (resp. square) vertices with target $\{t\}$. Starting from v_0, $\mathcal{P}_1$ has an optimal strategy σ_1 ensuring -10. At first glance, one could think that the memoryless strategy consisting in reaching v_1 is optimal, but in this case, t will never be reached as $\mathcal{P}_2$ will always go back to v_0. An optimal strategy σ_1 consists in counting 10 visits to v_1 before reaching t. Facing σ_1, the optimal (memoryless) strategy for $\mathcal{P}_2$ is to directly reach t, whose resulting outcome is a simple path. Two-player zero-sum shortest-path games can be solved in pseudo-polynomial time. In general, $\mathcal{P}_1$ does not necessarily have an optimal strategy, but if he has one, there is one with step-counting memory of pseudo-polynomial size. This contrasts with $\mathcal{P}_2$ who always has an optimal memoryless strategy [20, 21].

On the one hand, the multi-dimensional variant of the TSE problem discussed in Section 3.3.1 is undecidable if we allow unrestricted weights [61]. On the other hand, regarding NEs in n-player shortest-path games, the example from Figure 4b directly implies that NEs are not guaranteed to exist in this setting.

4 Stochastic Arenas

In this section, we consider arenas with randomised transitions. We focus on the case where one player progresses in a stochastic environment, i.e., one-player stochastic games, also known as Markov decision processes. In Section 4.1, we introduce our formal model as an



(a) An MDP where $\mathcal{P}_1$ has an incentive to move to v_2 only if the number of steps to visit t is limited.

(b) An MDP where both randomisation and memory are necessary to visit $\{t_1, t'_1\}$ and $\{t_2, t'_2\}$ with probability at least $\frac{3}{4}$.

Figure 6 Example MDPs. Circles and diamonds resp. represent $\mathcal{P}_1$ and stochastic vertices.

extension of the model of Section 2 and describe the problems we consider. Section 4.2 is the counterpart of Section 3 for the problems described in this section; we overview the complexity of strategies necessary to address these problems and comment on algorithms.

4.1 Markov Decision Processes

In this section, we extend notions of Section 2 to the one-player probabilistic setting.

Probability Distributions. Let A be a finite or countable set. We let $\mathcal{D}(A)$ denote the set of (discrete) probability distributions over A, i.e., functions $p\colon A \to [0,1]$ with $\sum_{a\in A} p(a) = 1$. For any $p \in \mathcal{D}(A)$, we let $\mathsf{supp}(p) = \{a \in A \mid p(a) > 0\}$ denote the support of p.

Markov Decision Processes. Let (V, E) be a directed graph as in Section 2. A *Markov decision process (MDP)* or one-player stochastic arena (on (V, E)) is a tuple $\mathcal{M} = (V_1, V_\Delta, E, \Delta)$ where (V_1, V_Δ) is a partition of V into vertices of $\mathcal{P}_1$ and stochastic vertices and $\Delta\colon V_\Delta \to \mathcal{D}(V)$ is a stochastic transition function such that $\mathsf{supp}(\Delta(v)) = \{v' \in V \mid (v, v') \in E\}$ for all $v \in V_\Delta$ (i.e., $\Delta(v)$ assigns a positive probability to all successors of v and only to them). We assume that MDPs are deadlock-free. MDPs subsume one-player game arenas.

▶ Remark 5. MDPs can alternatively be defined by a set of states, a set of actions, and a probabilistic transition function that assigns distributions over successor states to pairs of states and actions. The definition of MDPs presented here is equivalent [27], in addition to being compatible with the definitions of the previous section.

Definitions of plays and histories are identical to those provided in Section 2 for non-stochastic arenas. We will use the same notation. A *multi-weighted MDP* is a pair $(\mathcal{M}, w)$ where $w\colon E \to \mathbb{N}^k$ is a multi-dimensional (non-negative) weight function. An *event* is a set of measurable plays (for the Borel σ-algebra on the set of plays).

We illustrate two MDPs in Figure 6. Plays in MDPs proceed differently than in deterministic games: due to the presence of stochastic vertices, there may be several outcomes from an initial vertex for a given strategy of $\mathcal{P}_1$. At each round of a play, if the current vertex v is controlled by $\mathcal{P}_1$, then he chooses the next vertex, however if v is stochastic, then the next vertex is chosen randomly according to the distribution $\Delta(v)$. For instance, in the MDP of Figure 6a, the next vertex from v_1 is chosen uniformly at random between v_0 or t.

Randomised Strategies. In MDPs, we consider more general strategies than in Section 3. Randomised strategies yield a distribution over the possible choices of $\mathcal{P}_1$ instead of providing an action deterministically. These strategies allow a trade-off between different (and possibly

incompatible) goals. For instance, in the MDP depicted in Figure 6b, if $\mathcal{P}_1$ wants to ensure that the states t_1 and t_2 are each visited with probability $\frac{1}{2}$, he cannot do so without randomisation. In this example, both states t_1 and t_2 cannot be visited along the same path.

Formally, a (randomised) strategy (of $\mathcal{P}_1$) is a function $\sigma\colon \mathrm{Hist}_1 \to \mathcal{D}(V)$ such that, for all histories $h = h_0 \dots h_\ell \in \mathrm{Hist}_1$, $\mathsf{supp}(\sigma(h)) \subseteq \{v \in V \mid (h_\ell, v) \in E\}$. A strategy σ is *pure* if it does not use randomisation, i.e., if for all histories $h \in \mathrm{Hist}_1$, $|\mathsf{supp}(\sigma(h))| = 1$. Memoryless (randomised) strategies are defined in the same way as previously.

We say that a randomised strategy is *finite-memory* if it can be encoded by a (stochastic) Mealy machine $\mathfrak{M} = (\mathrm{M}, m_0, \mathsf{up}, \mathsf{nxt})$ where M, m_0 and up are defined in the same way as previously and $\mathsf{nxt}\colon \mathrm{M} \times V_1 \to \mathcal{D}(V)$ is changed to output distributions over successors vertices rather than a single successor. The strategy induced by a Mealy machine is defined similarly to the case of deterministic arenas, except it outputs distributions instead of vertices.

We remark that finite-memory randomised strategies can be defined in alternative ways, e.g., by also including randomisation in the initialisation or in the updates of the Mealy machine. Such alternatives are not equivalent to the one given above [56]. We do not consider richer models: the one above is sufficient to solve the problems considered in this section.

Outcomes. Let σ be a strategy of $\mathcal{P}_1$ and v_0 be an initial vertex. By fixing a strategy and initial vertex, we obtain a purely stochastic process, i.e., a Markov chain over the set of histories of the MDP. Intuitively, for each history h, one can assign a probability to the plays that extend h, defined as the product of the probability each edge along the history being taken according to the strategy of $\mathcal{P}_1$ and the stochastic transition function. This distribution extends in a unique manner to all events. We denote it by $\mathbb{P}^\sigma_{v_0}$. We refer the reader to, e.g., [5], for a formal definition of this distribution.

We stress that the nature of outcomes differs from the non-stochastic case. On the one hand, the outcome of a strategy profile from a vertex is a singular play in the non-stochastic setting. On the other hand, with MDPs, we obtain a Markov chain after fixing the strategy of $\mathcal{P}_1$ and the initial vertex. Furthermore, there can be infinitely many plays resulting from the Markov chain, even when the strategy of $\mathcal{P}_1$ is pure. This can be observed on the MDP of Figure 6a by considering the initial vertex v_0 and the pure memoryless strategy of $\mathcal{P}_1$ that moves from v_0 to v_1. Due to the stochastic vertex v_1, there are infinitely many plays played according to this strategy.

Relevant Events and Problems. In the following, we consider two types of events derived from gain and cost functions. Let T denote a target set and w a (one-dimensional) weight function. The first type of event relates to the purely Boolean reachability objective. Such an event is a set $\{\rho \in \mathrm{Plays} \mid \mathsf{Reach}[\mathrm{T}](\rho) = 1\}$. For conciseness, we will abusively denote this event by $\mathsf{Reach}[\mathrm{T}]$. The second type of event pertains to shortest-path problems, and these events are given by sets $\{\rho \in \mathrm{Plays} \mid \mathsf{SPath}_w[\mathrm{T}](\rho) \leq \alpha\}$ for some threshold $\alpha \in \mathbb{N}$.

We study both one-dimensional and multi-dimensional problems. In the one-dimensional case, we are interested in the maximal probability of an event and computing strategies that realise this probability. A strategy is *optimal* if it attains this probability.

In the multi-dimensional case, we want to satisfy lower-bound constraints on the probability of several events. More precisely, let $v_0 \in V$ be an initial vertex, $\Omega_1, \dots, \Omega_k$ be events (of the same type) and $\beta_1, \dots, \beta_k \in [0, 1]$ be thresholds. We want to decide the existence of strategy σ such that for all $j \in [\![k]\!]$, $\mathbb{P}^\sigma_{v_0}(\Omega_j) \geq \beta_j$, and synthesise such a strategy if it exists.

We remark that there also exist metrics other than probability to measure the quality of strategies that we do not discuss here. For instance, given a cost function, one could aim to find a strategy minimising the expected cost (e.g., see [8] for the shortest path cost function).

4.2 Reachability Problems in Markov Decision Processes

In this section, we overview the problems introduced at the end of the previous section. In Section 4.2.1, we present one-dimensional problems for which pure strategies suffice. We follow with their multi-dimensional counterparts in Section 4.2.2, for which randomised strategies are required. In both sections, we first discuss the Boolean reachability objective, then the shortest-path variant. For the former, memoryless strategies suffice in the one-dimensional case, and the same memory structure as in two-player generalised reachability games suffices in the multi-dimensional case. In the shortest-path case, the memory keeps track of the accumulated weight (for each dimension) in addition to the information used in the unweighted case. We fix an MDP $\mathcal{M} = (V_1, V_\Delta, E, \Delta)$ for this section.

4.2.1 Pure Strategies: One-dimensional Problems

Memoryless Strategies: Reachability. The maximum reachability probability can always be achieved using a *pure memoryless* strategy. Such a strategy can be obtained once the optimal probabilities have been computed for all vertices. First, we restrict edges that leave $\mathcal{P}_1$ vertices to those that enter a successor vertex with highest maximum probability. A memoryless strategy is optimal if and only if it uses only the restricted edges and, from all vertices with a positive maximum reachability probability, there is a path according to the strategy to a target state. A natural way to compute such a strategy is to select one such that, from all states with positive maximum reachability probability, there is a path to a target that is as short as possible (e.g., by a backwards induction approach).

Due to the nature of outcomes in this setting, in contrast with the non-stochastic setting, it is not possible to enforce that targets are reached by simple paths, even if a target is reached almost-surely. This can be observed on the MDP of Figure 6a. The (only) optimal memoryless strategy of $\mathcal{P}_1$ is the one that moves from v_0 to v_1, and it ensures t is reached with probability 1 from v_0 (and v_1). As explained previously, there are infinitely many plays played according to this strategy, and the time to reach a target in these plays is not bounded from above.

Computing the maximum probability of reaching a target set T in $\mathcal{M}$ can be done in polynomial time via linear programming [5]. Furthermore, deciding whether there is a strategy ensuring that a target is reached almost-surely is P-hard (for reasons similar to two-player zero-sum Boolean reachability games). The existence of strategies that ensure that T is reached almost-surely depends only on the graph underlying $\mathcal{M}$ and the partition of vertices, and is independent of the probabilities assigned by Δ. We remark that this special case is no easier than the general one from a complexity standpoint.

Counting-memory Strategies: Weight-constrained Reachability. Let T be a target set, $\alpha \in \mathbb{N}$, $\beta \in [0, 1]$ be thresholds and w be a one-dimensional weight function. In general, strategies that maximise the probability of the event $\Omega = \{\rho \in \text{Plays} \mid \mathsf{SPath}_w[\mathrm{T}](\rho) \leq \alpha\}$ require memory.

To illustrate this, let us consider the MDP of Figure 6a, assume that all edges have a weight of 1 (i.e., we count steps), and that $\mathcal{P}_1$ wants to maximise the probability of reaching t within four steps from v_0. We only describe the strategy in v_0 when two or four steps remain, as it is the only decision state and two steps are required to return to it. If two steps remain, the only optimal move is to go from v_0 to v_2, as it ensures the target is reached with probability $\frac{3}{4}$ instead of $\frac{1}{2}$. If four steps remain, moving to v_1 is the only optimal move, as it ensures a greater probability of success than $\frac{3}{4}$ due to the probability of success from v_0

when two steps remain being $\frac{3}{4}$. We observe that optimal strategies in this setting may select edges that would otherwise not be chosen when maximising the probability of $\mathsf{Reach}[\{t\}]$. In other words, $\mathcal{P}_1$ has an incentive to select riskier moves due to the weight constraints.

Deciding whether the maximum probability of Ω is at least β is PSPACE-hard, even for acyclic MDPs, for which it is PSPACE-complete [45]. In full generality, the problem can be solved in time polynomial in the size of the MDP and the threshold α, i.e., in pseudo-polynomial time. To compute the maximum probability and derive a maximising strategy, we can augment vertices of the MDP with the accumulated weight from the initial vertex and make vertices with an accumulated weight exceeding α absorbing (i.e., all edges leaving these vertices return to it). This ensures the expanded MDP is finite. On this expanded MDP, the optimal probabilities and an optimal strategy can be computed using the techniques described above, by considering as targets the expanded original target vertices with an accumulated weight of at most α.

Due to memoryless strategies being sufficient to maximise reachability probabilities, it follows that finite-memory strategies of pseudo-polynomial size that keep track of the accumulated weight suffice to maximise the probability of Ω, i.e., a *counting memory* is sufficient. A memory of pseudo-polynomial size is necessary in certain cases. Whether a move is optimal in a vertex in this setting is an intrinsic property of the difference between the bound α and the weight accumulated when entering the vertex. Unlike strategies with step-counting memory presented in Section 3.3.2, the strategies here do not only switch once from one memoryless strategy to another based on the number of steps taken, but instead make decisions based on the accumulated weight. In particular, in a given state, there may be more than two different memoryless strategies assigned to the memory states of a Mealy machine encoding an optimal strategy (e.g., this would occur if there were an additional state in the MDP of Figure 6a that reaches t with probability $\frac{7}{8}$ in exactly three steps).

4.2.2 Randomised Strategies: Multi-dimensional Problems

Branching-memory Strategies: Multi-objective Reachability. We now consider multiple targets $\mathrm{T}_1, \dots \mathrm{T}_k$ and probability thresholds $\beta_1, \dots, \beta_k \in [0,1]$, and the problem of existence and synthesis of a strategy σ such that for all $j \in [\![k]\!]$, $\mathbb{P}^\sigma_{v_0}(\mathsf{Reach}[\mathrm{T}_j]) \geq \beta_j$. Instead of maximising the probability of having $\mathsf{GReach}[\mathrm{T}_1, \dots, \mathrm{T}_k] = 1$, we treat each target separately. This allows one to study the trade-off between the different reachability objectives. The special case where, for all $j \in [\![k]\!]$, $\beta_j = 1$, is nonetheless equivalent to deciding the existence of a strategy ensuring that $\mathsf{GReach}[\mathrm{T}_1, \dots, \mathrm{T}_k] = 1$ almost-surely.

Unlike all previously introduced problems, randomised strategies are necessary to satisfy these multi-dimensional constraints, as was explained when introducing randomised strategies. Furthermore, memory may be necessary. Consider the MDP of Figure 6b and the target sets $\mathrm{T}_1 = \{t_1, t_1'\}$ and $\mathrm{T}_2 = \{t_2, t_2'\}$. To visit both target sets with probability $\frac{3}{4}$ from v_0, pure strategies do not suffice; at best, one target can be visited almost-surely and the other with a probability of $\frac{1}{2}$. Furthermore, randomised memoryless strategies can also be argued to be insufficient. A suitable strategy is one that selects a successor of v_0 uniformly at random, and then moves (deterministically) from v_0' to v_{3-j} if t_j was visited (i.e., the memory keeps track of previously visited target sets).

Furthermore, a memory of size exponential in the number of targets is necessary and sufficient. Consider the graph $\mathbf{QBF}_1$ of Figure 2 for the formula in Equation (1), and let existential (resp. universal) vertices be controlled by $\mathcal{P}_1$ (resp. be stochastic). For the targets we considered for generalised reachability, exponential memory is necessary to ensure all targets are visited almost-surely. In general, the same type of branching memory as in two-player zero-sum generalised reachability games is sufficient.

In a nutshell, the role of the randomisation in strategies is to balance the different objectives, whereas the role of the memory in strategies is to recall the visited targets. In the special case where all targets are absorbing, no memory is required [33], therefore randomised memoryless strategies suffice.

The problem can be shown to be PSPACE-hard using the previously sketched construction for general QBF formulas [62]. The case where $\beta_1 = \ldots = \beta_k = 1$ is PSPACE-complete [62]. The case where targets are absorbing can be solved in polynomial time by linear programming. The general problem can be solved in time polynomial in the size of the MDP and exponential in the number of targets, by reduction to the case where targets are absorbing [33].

Branching-counting-memory Strategies: Multi-objective Weight-constrained Reachability. In addition to the targets and probability thresholds fixed above, we consider a multi-dimensional weight function $w\colon E \to \mathbb{N}^k$ and thresholds $\alpha_1, \ldots, \alpha_k \in \mathbb{N}$. A strategy σ such that $\mathbb{P}^\sigma_{v_0}(\mathsf{SPath}_{w_j}[\mathrm{T}_j] \leq \alpha_i) \geq \beta_j$ holds for all $j \in [\![k]\!]$, if it exists, can always be found with a combination of the traits of the counting-memory and branching-memory strategies from the two previous problems. In other words, for such constraints, randomised finite-memory strategies that keep track of the accumulated weight and the sets of visited targets suffice. Therefore, memory exponential in the number of targets and the encoding of the bounds α_j is sufficient. The need for an exponential-sized memory in general follows from the need for an exponential-size memory for unweighted multi-objective reachability.

These strategies can be synthesised by merging the algorithmic ideas for the two previous problems [62]. First, one augments the MDP vertices with vectors of accumulated weights for dimensions j for which the bound α_j has not been exceeded yet. Next, by defining targets for each dimension in the augmented MDP as in the one-dimensional shortest-path problem, checking the existence of a strategy and synthesising it can be done via algorithms for multi-dimensional reachability. It follows that this problem can be solved in time polynomial in the size of the MDP and exponential in the number of targets and encoding of the bounds α_j. It also inherits PSPACE-hardness from the previous problem. This problem is undecidable if we allow weights in $\mathbb{Z}$ [62].

5 Conclusion

The goal of this paper was to look at reachability-related problems through the lens of memory. Now that our journey has reached its end, we briefly comment on what lies beyond the intended scope of this paper.

We have only considered *perfect information turn-based* games. Relaxing these hypotheses induces a need for more complex strategies, even in zero-sum Boolean reachability games. For instance, in concurrent games, neither player may have a winning strategy. Furthermore, randomisation may be necessary to maximise one's probability of winning. There may not be a strategy that ensures the maximal probability of $\mathcal{P}_1$, although this probability can be approached by memoryless randomised strategies [29, 47]. Moreover, without perfect information, $\mathcal{P}_1$ may need a memory of exponential size, in addition to randomisation, to maximise the reachability probability [7]. Deciding whether $\mathcal{P}_1$ has an almost-surely winning strategy in a concurrent reachability game of imperfect information is 2-EXPTIME-complete [7], a stark contrast with the turn-based perfect information setting.

Throughout this paper, we have only dealt with *finite* arenas. Reachability problems on infinite arenas with a finite description have also garnered interest. Notable such problems include the halting problems for Turing [68] or Minsky [57] machines, which are undecidable.

The problem of solving zero-sum reachability games on the configuration graph of a pushdown automaton is EXPTIME-complete [71]. The reachability problem in Petri nets and vector addition systems was recently shown to be Ackermann-complete [54, 28]. Regarding questions related to strategy complexity in infinite games, one can mention [10] where the memory requirements of winning strategies are studied in zero-sum games where the goal is to construct (i.e., reach) a word from a regular language.

When considering strategies, we only introduced *randomisation* in situations were it was required. In particular, we did not discuss the potential trade-off between the size of a deterministic Mealy machine and the use of randomisation. In some cases, as discussed in [58, 24, 46, 25], considering randomised strategies in situations where pure strategies suffice can yield smaller memory bounds for optimal (or close-to-optimal) strategies. Specifically studying the existence of pure strategies to solve certain problems can sometimes turn out more complex than searching for randomised strategies [30].

We have highlighted some similarities in the memory structure needed to solve some problems that are often treated as different. We believe that the memory structure of strategies should be investigated deeper, in line with the effort to understand when complex strategies are required that is described in Section 1. There exist works that study alternatives to Mealy machine to represent strategies, e.g., models based on Turing machine [41], decision trees for memoryless strategies (e.g., [2, 3]) or compact counter-based models in a setting with limited resources [9].

In Section 3.2, we noted that although Büchi objectives appear more complex than reachability objectives, solving games with objectives derived from the former is often simpler than dealing with games based on the latter. A key ingredient enabling efficient algorithms to solve games involving Büchi objectives is the ability to heavily rely on SCCs. From this, one could question whether there is a relation between the SCC decomposition of a reachability game and the memory structure needed to solve it.

References

1 Eric Allender. Reachability problems: An update. In S. Barry Cooper, Benedikt Löwe, and Andrea Sorbi, editors, *Computation and Logic in the Real World, Third Conference on Computability in Europe, CiE 2007, Siena, Italy, June 18-23, 2007, Proceedings*, volume 4497 of *Lecture Notes in Computer Science*, pages 25–27. Springer, 2007. doi:10.1007/978-3-540-73001-9_3.

2 Pranav Ashok, Mathias Jackermeier, Jan Kretínský, Christoph Weinhuber, Maximilian Weininger, and Mayank Yadav. dtControl 2.0: Explainable strategy representation via decision tree learning steered by experts. In Jan Friso Groote and Kim Guldstrand Larsen, editors, *Proceedings (Part II) of the 27th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2021, Held as Part of ETAPS 2021, Luxemburg City, Luxemburg, March 27–April 1, 2021*, volume 12652 of *Lecture Notes in Computer Science*, pages 326–345. Springer, 2021. doi:10.1007/978-3-030-72013-1_17.

3 Pranav Ashok, Jan Kretínský, Kim Guldstrand Larsen, Adrien Le Coënt, Jakob Haahr Taankvist, and Maximilian Weininger. SOS: safe, optimal and small strategies for hybrid markov decision processes. In David Parker and Verena Wolf, editors, *Proceedings of the 16th International Conference on Quantitative Evaluation of Systems, QEST 2019, Glasgow, UK, September 10–12, 2019*, volume 11785 of *Lecture Notes in Computer Science*, pages 147–164. Springer, 2019. doi:10.1007/978-3-030-30281-8_9.

4 Bengt Aspvall, Michael F. Plass, and Robert Endre Tarjan. A linear-time algorithm for testing the truth of certain quantified Boolean formulas. *Information Processing Letters*, 8(3):121–123, 1979. doi:10.1016/0020-0190(79)90002-4.

5 Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008.

6 Richard Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16:87–90, 1958.

7 Nathalie Bertrand, Blaise Genest, and Hugo Gimbert. Qualitative determinacy and decidability of stochastic games with signals. *Journal of the ACM*, 64(5):33:1–33:48, 2017. doi:10.1145/3107926.

8 Dimitri P. Bertsekas and John N. Tsitsiklis. An analysis of stochastic shortest path problems. *Mathematics of Operations Research*, 16(3):580–595, 1991. doi:10.1287/moor.16.3.580.

9 Frantisek Blahoudek, Tomás Brázdil, Petr Novotný, Melkior Ornik, Pranay Thangeda, and Ufuk Topcu. Qualitative controller synthesis for consumption Markov decision processes. In Shuvendu K. Lahiri and Chao Wang, editors, *Proceedings (Part II) of the 32nd International Conference on Computer Aided Verification, CAV 2020, Los Angeles, CA, USA, July 21–24, 2020*, volume 12225 of *Lecture Notes in Computer Science*, pages 421–447. Springer, 2020. doi:10.1007/978-3-030-53291-8_22.

10 Patricia Bouyer, Nathanaël Fijalkow, Mickael Randour, and Pierre Vandenhove. How to play optimally for regular objectives? In Kousha Etessami, Uriel Feige, and Gabriele Puppis, editors, *Proceedings of the 50th International Colloquium on Automata, Languages, and Programming, ICALP 2023, July 10–14, 2023, Paderborn, Germany*, volume 261 of *LIPIcs*, pages 118:1–118:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ICALP.2023.118.

11 Patricia Bouyer, Stéphane Le Roux, Youssouf Oualhadj, Mickael Randour, and Pierre Vandenhove. Games where you can play optimally with arena-independent finite memory. *Logical Methods in Computer Science*, 18(1), 2022. doi:10.46298/lmcs-18(1:11)2022.

12 Patricia Bouyer, Youssouf Oualhadj, Mickael Randour, and Pierre Vandenhove. Arena-independent finite-memory determinacy in stochastic games. In Serge Haddad and Daniele Varacca, editors, *Proceedings of the 32nd International Conference on Concurrency Theory, CONCUR 2021, Virtual Conference, August 24–27, 2021*, volume 203 of *LIPIcs*, pages 26:1–26:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.CONCUR.2021.26.

13 Patricia Bouyer, Mickael Randour, and Pierre Vandenhove. The true colors of memory: A tour of chromatic-memory strategies in zero-sum games on graphs (invited talk). In Anuj Dawar and Venkatesan Guruswami, editors, *Proceedings of the 42nd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2022, IIT Madras, Chennai, India, December 18–20, 2022*, volume 250 of *LIPIcs*, pages 3:1–3:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.FSTTCS.2022.3.

14 Romain Brenguier. *Equilibres de Nash dans les jeux concurrents : application aux jeux temporisés. (Nash equilibria in concurrent games : application to timed games)*. PhD thesis, École normale supérieure de Cachan, France, 2012. URL: https://tel.archives-ouvertes.fr/tel-00827027.

15 Léonard Brice, Jean-François Raskin, and Marie van den Bogaard. On the complexity of SPEs in parity games. In Florin Manea and Alex Simpson, editors, *Proceedings of the 30th EACSL Annual Conference on Computer Science Logic, CSL 2022, Göttingen, Germany, February 14–19, 2022*, volume 216 of *LIPIcs*, pages 10:1–10:17. Schloss Dagstuhl –Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.CSL.2022.10.

16 Thomas Brihaye, Véronique Bruyère, Aline Goeminne, and Jean-François Raskin. Constrained existence problem for weak subgame perfect equilibria with ω-regular Boolean objectives. *Information and Computation*, 278:104594, 2021. doi:10.1016/j.ic.2020.104594.

17 Thomas Brihaye, Véronique Bruyère, Aline Goeminne, Jean-François Raskin, and Marie van den Bogaard. The complexity of subgame perfect equilibria in quantitative reachability games. *Logical Methods in Computer Science*, 16(4), 2020. URL: https://lmcs.episciences.org/6883.

18 Thomas Brihaye, Véronique Bruyère, Aline Goeminne, and Nathan Thomasset. On relevant equilibria in reachability games. *Journal of Computer and System Sciences*, 119:211–230, 2021. doi:10.1016/j.jcss.2021.02.009.

19 Thomas Brihaye, Julie De Pril, and Sven Schewe. Multiplayer cost games with simple Nash equilibria. In Sergei N. Artëmov and Anil Nerode, editors, *Proceedings of the International Symposium on Logical Foundations of Computer Science, LFCS 2013, San Diego, CA, USA, January 6–8, 2013*, volume 7734 of *Lecture Notes in Computer Science*, pages 59–73. Springer, 2013. doi:10.1007/978-3-642-35722-0_5.

20 Thomas Brihaye, Gilles Geeraerts, Axel Haddad, and Benjamin Monmege. To reach or not to reach? Efficient algorithms for total-payoff games. In Luca Aceto and David de Frutos-Escrig, editors, *Proceedings of the 26th International Conference on Concurrency Theory, CONCUR 2015, Madrid, Spain, September 1–4, 2015*, volume 42 of *LIPIcs*, pages 297–310. Schloss Dagstuhl –Leibniz-Zentrum für Informatik, 2015. doi:10.4230/LIPIcs.CONCUR.2015.297.

21 Thomas Brihaye, Gilles Geeraerts, Axel Haddad, and Benjamin Monmege. Pseudopolynomial iterative algorithm to solve total-payoff games and min-cost reachability games. *Acta Informatica*, 54(1):85–125, 2017. doi:10.1007/s00236-016-0276-z.

22 Thomas Brihaye and Aline Goeminne. Multi-weighted reachability games. *CoRR*, abs/2308.09625, 2023. doi:10.48550/arXiv.2308.09625.

23 Véronique Bruyère, Quentin Hautem, and Jean-François Raskin. On the complexity of heterogeneous multidimensional games. In Josée Desharnais and Radha Jagadeesan, editors, *Proceedings of the 27th International Conference on Concurrency Theory, CONCUR 2016, Québec City, Canada, August 23–26, 2016*, volume 59 of *LIPIcs*, pages 11:1–11:15. Schloss Dagstuhl –Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.CONCUR.2016.11.

24 Krishnendu Chatterjee, Luca de Alfaro, and Thomas A. Henzinger. Trading memory for randomness. In *Proceedings of the 1st International Conference on Quantitative Evaluation of Systems, QEST 2004, Enschede, The Netherlands, 27–30 September 2004*, pages 206–217. IEEE Computer Society, 2004. doi:10.1109/QEST.2004.1348035.

25 Krishnendu Chatterjee, Mickael Randour, and Jean-François Raskin. Strategy synthesis for multi-dimensional quantitative objectives. *Acta Informatica*, 51(3-4):129–163, 2014. doi:10.1007/s00236-013-0182-6.

26 Stephen A. Cook. The complexity of theorem-proving procedures. In Michael A. Harrison, Ranan B. Banerji, and Jeffrey D. Ullman, editors, *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, STOC 1971, Shaker Heights, Ohio, USA, May 3–5, 1971*, pages 151–158. ACM, 1971. doi:10.1145/800157.805047.

27 Costas Courcoubetis and Mihalis Yannakakis. Markov decision processes and regular events. *IEEE Transactions on Automatic Control*, 43(10):1399–1418, 1998. doi:10.1109/9.720497.

28 Wojciech Czerwiński and Lukasz Orlikowski. Reachability in vector addition systems is Ackermann-complete. In *Proceedings of the 62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7–10, 2022*, pages 1229–1240. IEEE, 2021. doi:10.1109/FOCS52979.2021.00120.

29 Luca de Alfaro, Thomas A. Henzinger, and Orna Kupferman. Concurrent reachability games. *Theoretical Computer Science*, 386(3):188–217, 2007. doi:10.1016/j.tcs.2007.07.008.

30 Florent Delgrange, Joost-Pieter Katoen, Tim Quatmann, and Mickael Randour. Simple strategies in multi-objective MDPs. In Armin Biere and David Parker, editors, *Proceedings (Part I) of the 26th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2020, Held as Part of ETAPS 2020, Dublin, Ireland, April 25–30, 2020*, volume 12078 of *Lecture Notes in Computer Science*, pages 346–364. Springer, 2020. doi:10.1007/978-3-030-45190-5_19.

31 Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959. doi:10.1007/BF01386390.

32 Stefan Dziembowski, Marcin Jurdziński, and Igor Walukiewicz. How much memory is needed to win infinite games? In *Proceedings of the 12th Annual IEEE Symposium on Logic in Computer Science, LICS 1997, Warsaw, Poland, June 29 – July 2, 1997*, pages 99–110. IEEE Computer Society, 1997. doi:10.1109/LICS.1997.614939.

33 Kousha Etessami, Marta Z. Kwiatkowska, Moshe Y. Vardi, and Mihalis Yannakakis. Multi-objective model checking of Markov decision processes. *Logical Methods in Computer Science*, 4(4), 2008. doi:10.2168/LMCS-4(4:8)2008.

34 Nathanaël Fijalkow, Nathalie Bertrand, Patricia Bouyer-Decitre, Romain Brenguier, Arnaud Carayol, John Fearnley, Hugo Gimbert, Florian Horn, Rasmus Ibsen-Jensen, Nicolas Markey, Benjamin Monmege, Petr Novotný, Mickael Randour, Ocan Sankur, Sylvain Schmitz, Olivier Serre, and Mateusz Skomra. Games on graphs. *CoRR*, abs/2305.10546, 2023. doi:10.48550/arXiv.2305.10546.

35 Nathanaël Fijalkow and Florian Horn. The surprizing complexity of reachability games. *CoRR*, abs/1010.2420, 2010. doi:10.48550/arXiv.1010.2420.

36 Nathanaël Fijalkow and Florian Horn. Les jeux d'accessibilité généralisée. *Technique et Science Informatiques*, 32(9-10):931–949, 2013. doi:10.3166/tsi.32.931-949.

37 Robert W. Floyd. Algorithm 97: Shortest path. *Communications of the ACM*, 5(6):345, 1962. doi:10.1145/367766.368168.

38 Lester R. Ford. Network flow theory. Paper P-923, The RAND Corporation, Santa Monica, California, August 14, 1956. URL: http://www.rand.org/pubs/papers/P923.html.

39 James Friedman. A non-cooperative equilibrium for supergames. *Review of Economic Studies*, 38(1):1–12, 1971. URL: https://EconPapers.repec.org/RePEc:oup:restud:v:38:y:1971:i:1:p:1-12.

40 Drew Fudenberg and David Levine. Subgame-perfect equilibria of finite-and infinite-horizon games. *Journal of Economic Theory*, 31(2):251–268, 1983.

41 Marcus Gelderie. *Strategy machines: representation and complexity of strategies in infinite games*. PhD thesis, RWTH Aachen University, 2014. URL: http://darwin.bth.rwth-aachen.de/opus3/volltexte/2014/5025.

42 Hugo Gimbert and Wiesław Zielonka. Games where you can play optimally without any memory. In Martín Abadi and Luca de Alfaro, editors, *Proceedings of the 16th International Conference on Concurrency Theory, CONCUR 2005, San Francisco, CA, USA, August 23–26, 2005*, volume 3653 of *Lecture Notes in Computer Science*, pages 428–442. Springer, 2005. doi:10.1007/11539452_33.

43 Hugo Gimbert and Wiesław Zielonka. Pure and Stationary Optimal Strategies in Perfect-Information Stochastic Games with Global Preferences. Unpublished, 2009. URL: https://hal.archives-ouvertes.fr/hal-00438359.

44 Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research [outcome of a Dagstuhl seminar, February 2001]*, volume 2500 of *Lecture Notes in Computer Science*. Springer, 2002. doi:10.1007/3-540-36387-4.

45 Christoph Haase and Stefan Kiefer. The odds of staying on budget. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *Proceedings (Part II) of the 42nd International Colloquium on Automata, Languages, and Programming, ICALP 2015, Kyoto, Japan, July 6–10, 2015*, volume 9135 of *Lecture Notes in Computer Science*, pages 234–246. Springer, 2015. doi:10.1007/978-3-662-47666-6_19.

46 Florian Horn. Random fruits on the Zielonka tree. In Susanne Albers and Jean-Yves Marion, editors, *Proceedings of the 26th International Symposium on Theoretical Aspects of Computer Science, STACS 2009, Freiburg, Germany, February 26–28, 2009*, volume 3 of *LIPIcs*, pages 541–552. Schloss Dagstuhl –Leibniz-Zentrum für Informatik, Germany, 2009. doi:10.4230/LIPIcs.STACS.2009.1848.

47 Rasmus Ibsen-Jensen. Concurrent games. In Nathanaël Fijalkow, editor, *Games on Graphs*. arXiv, 2023.

48 Neil Immerman. Number of quantifiers is better than number of tape cells. *Journal of Computer and System Sciences*, 22(3):384–406, 1981. doi:10.1016/0022-0000(81)90039-8.

49 Neil D. Jones. Space-bounded reducibility among combinatorial problems. *Journal of Computer and System Sciences*, 11(1):68–85, 1975. doi:10.1016/S0022-0000(75)80050-X.

50 Leonid Khachiyan, Endre Boros, Konrad Borys, Khaled M. Elbassioni, Vladimir Gurvich, Gábor Rudolf, and Jihui Zhao. On short paths interdiction problems: Total and node-wise limited interdiction. *Theory of Computing Systems*, 43(2):204–233, 2008. doi:10.1007/s00224-007-9025-6.

51 Stephen Cole Kleene. Representation of events in nerve nets and finite automata. *C. E. Shannon and J. McCarthy (ed.). Automata Studies. Princeton University Press*, pages 3–42, 1956.

52 François Laroussinie, Nicolas Markey, and Ghassan Oreiby. Model-checking timed ATL for durational concurrent game structures. In Eugene Asarin and Patricia Bouyer, editors, *Proceedings of the 4th International Conference on Formal Modeling and Analysis of Timed Systems, FORMATS 2006, Paris, France, September 25–27, 2006*, volume 4202 of *Lecture Notes in Computer Science*, pages 245–259. Springer, 2006. doi:10.1007/11867340_18.

53 Stéphane Le Roux, Arno Pauly, and Mickael Randour. Extending finite-memory determinacy by Boolean combination of winning conditions. In Sumit Ganguly and Paritosh K. Pandya, editors, *Proceedings of the 38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2018, Ahmedabad, India, December 11–13, 2018*, volume 122 of *LIPIcs*, pages 38:1–38:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.FSTTCS.2018.38.

54 Jérôme Leroux. The reachability problem for Petri nets is not primitive recursive. In *Proceedings of the 62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7–10, 2022*, pages 1241–1252. IEEE, 2021. doi:10.1109/FOCS52979.2021.00121.

55 James C. A. Main. Arena-independent memory bounds for nash equilibria in reachability games. *CoRR*, abs/2310.02142, 2023. doi:10.48550/arXiv.2310.02142.

56 James C. A. Main and Mickael Randour. Different strokes in randomised strategies: Revisiting Kuhn's theorem under finite-memory assumptions. In Bartek Klin, Slawomir Lasota, and Anca Muscholl, editors, *Proceedings of the 33rd International Conference on Concurrency Theory, CONCUR 2022, Warsaw, Poland, September 12–16, 2022*, volume 243 of *LIPIcs*, pages 22:1–22:18. Schloss Dagstuhl –Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.CONCUR.2022.22.

57 Marvin L. Minsky. *Computation: Finite and Infinite Machines*. Prentice-Hall, Inc., USA, 1967.

58 Benjamin Monmege, Julie Parreaux, and Pierre-Alain Reynier. Reaching your goal optimally by playing at random with no memory. In Igor Konnov and Laura Kovács, editors, *Proceedings of the 31st International Conference on Concurrency Theory, CONCUR 2020, Vienna, Austria, September 1–4, 2020*, volume 171 of *LIPIcs*, pages 26:1–26:21. Schloss Dagstuhl –Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.CONCUR.2020.26.

59 Martin J. Osborne and Ariel Rubinstein. *A course in game theory*. The MIT Press, 1994.

60 Christos H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994.

61 Mickael Randour. Games with multiple objectives. In Nathanaël Fijalkow, editor, *Games on Graphs*. arXiv, 2023.

62 Mickael Randour, Jean-François Raskin, and Ocan Sankur. Percentile queries in multi-dimensional Markov decision processes. *Formal methods in system design*, 50(2-3):207–248, 2017. doi:10.1007/s10703-016-0262-7.

63 Stéphane Le Roux and Arno Pauly. Extending finite-memory determinacy to multi-player games. *Information and Computation*, 261:676–694, 2018. doi:10.1016/j.ic.2018.02.024.

64 Bernard Roy. Transitivité et connexité. *Comptes Rendus de l'Académie des Sciences Paris*, 249:216–218, 1959.

65 Robert Endre Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972. doi:10.1137/0201010.

66 Wolfgang Thomas. On the synthesis of strategies in infinite games. In Ernst W. Mayr and Claude Puech, editors, *Proceedings of the 12th Annual Symposium on Theoretical Aspects of Computer Science, STACS 1995, Munich, Germany, March 2–4, 1995*, volume 900 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 1995. doi:10.1007/3-540-59042-0_57.

67 Stephen D. Travers. The complexity of membership problems for circuits over sets of integers. *Theoretical Computer Science*, 369(1-3):211–229, 2006. doi:10.1016/j.tcs.2006.08.017.

68 Alan M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, s2-42(1):230–265, 1937. doi:10.1112/plms/s2-42.1.230.

69 Michael Ummels. Rational behaviour and strategy construction in infinite multiplayer games. In S. Arun-Kumar and Naveen Garg, editors, *Proceedings of the 26th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FST TCS 2006, Kolkata, India, December 13–15, 2006*, volume 4337 of *Lecture Notes in Computer Science*, pages 212–223. Springer, 2006. doi:10.1007/11944836_21.

70 Michael Ummels. The complexity of Nash equilibria in infinite multiplayer games. In Roberto M. Amadio, editor, *Proceedings of the 11th International Conference on Foundations of Software Science and Computational Structures, FoSSaCS 2008, Held as Part of ETAPS 2008, Budapest, Hungary, March 29 – April 6, 2008*, volume 4962 of *Lecture Notes in Computer Science*, pages 20–34. Springer, 2008. doi:10.1007/978-3-540-78499-9_3.

71 Igor Walukiewicz. Pushdown processes: Games and model-checking. *Information and Computation*, 164(2):234–263, 2001. doi:10.1006/inco.2000.2894.

72 Stephen Warshall. A theorem on Boolean matrices. *Journal of the ACM*, 9(1):11–12, 1962. doi:10.1145/321105.321107.

73 Avi Wigderson. The complexity of graph connectivity. In Ivan M. Havel and Václav Koubek, editors, *Proceedings of the 17th International Symposium on Mathematical Foundations of Computer Science, MFCS 1992, Prague, zechoslovakia, August 24–28, 1992*, volume 629 of *Lecture Notes in Computer Science*, pages 112–132. Springer, 1992. doi:10.1007/3-540-55808-X_10.

74 James Worrell. The Skolem landscape (invited talk). In Kousha Etessami, Uriel Feige, and Gabriele Puppis, editors, *Proceedings of the 50th International Colloquium on Automata, Languages, and Programming, ICALP 2023, July 10–14, 2023, Paderborn, Germany*, volume 261 of *LIPIcs*, pages 5:1–5:2. Schloss Dagstuhl –Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ICALP.2023.5.