Understanding Clock Constraints Coalgebraically

Grygoriy Zholtkevych[0000−0002−7515−2143]* and Maksym Labzhaniia[0000−0002−2666−3959]

Department of Theoretical and Applied Computer Science
School of Mathematics and Computer Science
V.N. Karazin Kharkiv National University
4, Svobody Sqr., Kharkiv, 61022, Ukraine
g.zholtkevych@karazin.ua; m.labzhaniia@gmail.com

Abstract. The paper is devoted to the problem specifying causality relationships in distributed (including cyber-physical) systems. This problem is studied based on the coalgebraic approach used authors for studying safety constraints for distributed systems. The special class of coalgebras, counter-based detectors, is introduced and studied in the paper. It is shown that this approach allows using the technique of Diophantine equations for specifying clock constraints of the system being studied. The advantage of the approach is the possibility for defining the complexity of detectors that provides to control respond time of the detectors in the system.

Keywords: a coalgebra, a detector coalgebra, a counter-based detector, Diophantine equation, Clock Constraint Specification Language

1 Introduction

This paper presents the authors' research continuing the research presented in [1] and applying the proposed approach and results in a more specific situation.

The general context of our study is determined by the observation that modern technical systems are, as a rule, compound and smart. Moreover, such systems can be hybrid in the sense that ones consisted of physical, cybernetic (software), and, maybe, social components. This character of the systems should be taken into account under designing ones.

The software components of such systems are reactive, that is, they are designed for providing the required behaviour of the system, and not for obtaining any computational result.

Components of such systems are distributed in space but should act for guaranteeing the required behaviour of the system in the whole. So, supporting the necessary causality relationships during operating of system components is one of the principal problems of designing a system of such a kind.

* The author thanks professors R. de Simone, F. Mallet, and L. Liquori for detailed discussions of the problems related to this paper during his internship at Inria Sophia Antipolis - Médi and Campus France for funding this internship.

Also, we need to remark the incorrect behaviour of a system of the considering kind can have serious and some times catastrophic consequences for the system surroundings. Hence, systems of this class can be characterised as safety-critical systems. Thus, all designing solutions for them should be rigorous verified.

This research is some attempt to build a foundation for rigorous and effective specification and analysis of causality relationships in systems of the considering class.

This paper applies the general framework proposed in [1] for constructing rigorous models of logical time intended for using under developing domain-specific languages for specifying causality constraints.

The paper is structured as follows.

Sec. 2 reminds basic concepts and notation.

Sec. 3 contains the necessary definitions and results for understanding the general coalgebraic framework.

Sec. 4 contains a survey of used below results from [1].

Sec. 5 explains how results mentioned above can be used for specification and analysis causality constraints.

Sec. 6 is the key in the article. Here, it is introduced the concept of a counter–based detector; it is explained that this class of detectors is not complete; it is constructed a Diophantine representation for detectors of this class.

Finally, Sec. 7 presents the construction of a universal simulator based on the Diophantine representation for counter-based detectors.

2 Basic Concepts and Notation

This section is a copy of the corresponding section of our article [1] and here it is presented only for providing notation consistency. Thus, we assume that X is a finite set with at least two elements. Elements of this set are usually interpreted as system notifications.

A mapping (generally speaking partial) $\boldsymbol{s} : \mathbb{N} \to X$ is called an X-***sequence*** if for any $k \in \mathbb{N}$, $\boldsymbol{s}\, k'$ is defined whenever $\boldsymbol{s}\, k$ is defined and $0 \leq k' < k$.

An X-sequence $\boldsymbol{s}$ is called an X-***word*** if there exists $k \in \mathbb{N}$ such that $\boldsymbol{s}\, k$ is undefined; in contrast, an X-sequence $\boldsymbol{s}$ is called a X-***stream*** if $\boldsymbol{s}\, k$ is defined for all $k \in \mathbb{N}$.

We use the notation X^* for referring to the set of all X-words, and $X^{\mathbb{N}}$ for referring to the set of all X-streams. The set X^* contains the sequence defined nowhere, which is below denoted by $\boldsymbol{\epsilon}$.

We use also the notation X^∞ for referring to the set $X^* \bigcup X^{\mathbb{N}}$, and X^+ for referring to the set of all X-words without the word defined nowhere.

For an X-word $\boldsymbol{u}$, we denote by $|\boldsymbol{u}|$ the minimal natural number such that $\boldsymbol{u}\, |\boldsymbol{u}|$ is undefined.
The value $|\boldsymbol{u}|$ where $\boldsymbol{u} \in X^*$ is called ***length*** of $\boldsymbol{u}$.

As usually, we identify $n \in X$ with the X-word $\boldsymbol{u} \in X^*$ such that $\boldsymbol{u}\, 0 = n$ and $\boldsymbol{u}\, k$ is undefined if $k > 0$.

For $\boldsymbol{u} \in X^*$ and $\boldsymbol{s} \in X^\infty$, we denote by $\boldsymbol{us}$ the next X-sequence

$$\boldsymbol{us} = \lambda\ k \in \mathbb{N} \,.\begin{cases} \boldsymbol{u}\,k & \text{if } k < |\boldsymbol{u}| \\ \boldsymbol{s}(k - |\boldsymbol{u}|) & \text{otherwise} \end{cases}$$

Below we need in the following set

$$n^{-1} \cdot A = \{\boldsymbol{u} \in X^* \mid n\boldsymbol{u} \in A\} \qquad \text{where } A \subset X^* \text{ and } n \in X.$$

For $\boldsymbol{s} \in X^\infty$ and $m \in \mathbb{N}$, we denote by $\boldsymbol{s}_{m..}$ the next X-sequence

$$\boldsymbol{s}_{m..} = \lambda\ k \in \mathbb{N} \,.\begin{cases} \boldsymbol{s}(k+m) & \text{if this value is defined} \\ \text{is undefined} & \text{otherwise} \end{cases}$$

Also for $\boldsymbol{s} \in X^\infty$ and $m, l \in \mathbb{N}$, we denote by $\boldsymbol{s}_{m..l}$ the next X-word

$$\boldsymbol{s}_{m..l} = \lambda\ k \in \mathbb{N} \,.\begin{cases} \boldsymbol{s}(k+m) & \text{if this value is defined and } k < l - m \\ \text{is undefined} & \text{otherwise} \end{cases}$$

The principal concepts for our studying is given by the following definitions

Definition 1. *A subset $P \subset X^*$ is called* ***prefix-free*** *if $\boldsymbol{u} \in P$ ensures $\boldsymbol{u}_{0..m} \notin P$ whenever $0 \leq m < |\boldsymbol{u}|$.*

Remark 1. If a prefix-free subset of X^* contains $\boldsymbol{\epsilon}$ then this subset is $\{\boldsymbol{\epsilon}\}$. Indeed, if a prefix-free subset of X^* contains both $\boldsymbol{\epsilon}$ and another X-word $\boldsymbol{u}$ then $\boldsymbol{u}_{0..0} = \boldsymbol{\epsilon}$ cannot belong to this subset. This contradiction grounds the remark.

Definition 2 (see [2]). *A* ***safety constraint*** *is a subset $S \subset X^{\mathbb{N}}$ such that $\boldsymbol{s} \in S$ if for any $m \in \mathbb{N}$, $\boldsymbol{s}_{0..m} = \boldsymbol{s}'_{0..m}$ for some $\boldsymbol{s}' \in S$.*

3 Coalgebras Preliminaries

In this section, we remind the basic definitions and facts related to the concept of a coalgebra in an arbitrary category. In addition, we give some specific concepts in the case when $\mathbb{C} = \textbf{Set}$.

Thus, we assume the category $\mathbb{C}$ and the endofunctor $\boldsymbol{F}$ of $\mathbb{C}$ are given and held fixed in this section[1].

Definition 3. *A morphism a of $\mathbb{C}$ is called an* ***$\boldsymbol{F}$-coalgebra*** *if the equation $\operatorname{cod} a = \boldsymbol{F}(\operatorname{dom} a)$ is fulfilled. In the case, $\operatorname{dom} a$ is called the* ***carrier*** *of a and denoted below by $\underline{a}$.*

[1] General information on category theory can be found in [3,4], proofs of some facts formulated in the subsection can be found in [1, Sec. 3]

Definition 4. *Let a and b be $\boldsymbol{F}$-coalgebras then a morphism $f\colon \underline{a} \to \underline{b}$ is called an **$\boldsymbol{F}$-morphism** from a into b (symbolically, $f : a \to b$) if the diagram*

$$\begin{tikzcd}
\underline{a} \arrow[r, "f"] \arrow[d, "a"'] & \underline{b} \arrow[d, "b"] \\
\boldsymbol{F}\,\underline{a} \arrow[r, "\boldsymbol{F}\,f"] & \boldsymbol{F}\,\underline{b}
\end{tikzcd}$$

commutes or, equivalently, the equation $(\boldsymbol{F}\,f)\,a = b\,f$ holds.

Proposition 1. *The class of $\boldsymbol{F}$-coalgebras equipped with $\boldsymbol{F}$-morphisms is a category denoted usually by $\mathsf{Coalg}_{\boldsymbol{F}}(\mathbb{C})$ or $\mathsf{Coalg}_{\boldsymbol{F}}$ if the category $\mathbb{C}$ is clear from the context.*

Definition 5.
(1) *A terminal object of $\mathsf{Coalg}_{\boldsymbol{F}}$ if it exists is called a **final** $\boldsymbol{F}$-coalgebra, which is denoted by $\nu\boldsymbol{F}$.*
(2) *For any $\boldsymbol{F}$-coalgebra a, the unique $\boldsymbol{F}$-morphism from a into $\nu\boldsymbol{F}$ is called an **anamorphism** and denoted by $[\![a]\!]$.*

4 Safety Behavioural Constraints

This section contains some servey of the results obtained in [1].

In [1], three endofunctors T, $\mathsf{S}_{\mathbf{N}}$, and $\mathsf{D}_{\mathbf{N}}$ of category Set are considered. Here $\mathbf{N}$ refers to some finite set of system notifications. The definitions of these functors are given in Table 1.

Table 1. Functors T, $\mathsf{S}_{\mathbf{N}}$, and $\mathsf{D}_{\mathbf{N}}$ definitions

$\boldsymbol{F}$	$\boldsymbol{F}\,X$ where $X \in \mathsf{Set}$	$\boldsymbol{F}\,f$ where $X, Y \in \mathsf{Set}$ and $f : X \to Y$
T	$\mathsf{T}\,X = 1 + X$	$\mathsf{T}\,f = \lambda\, x \in \mathsf{T}\,X \,.\, \begin{cases} \Downarrow & \text{if } x = \Downarrow \\ fx & \text{otherwise} \end{cases}$
$\mathsf{S}_{\mathbf{N}}$	$\mathsf{S}_{\mathbf{N}}\,X = \mathbf{N} \times X$	$\mathsf{S}_{\mathbf{N}}\,f = \mathrm{id}_{\mathbf{N}} \times f$
$\mathsf{D}_{\mathbf{N}}$	$\mathsf{D}_{\mathbf{N}}\,X = (1 + X)^{\mathbf{N}}$	$\mathsf{D}_{\mathbf{N}}\,f = \lambda\, \phi \in \mathsf{D}_{\mathbf{N}}\,X \,.\, \lambda\, \boldsymbol{n} \in \mathbf{N} \,.\, \begin{cases} \Downarrow & \text{if } \phi\,\boldsymbol{n} = \Downarrow \\ f(\phi\,\boldsymbol{n}) & \text{otherwise} \end{cases}$

In Table 1, "+" means disjoint union, $1 = \{\Downarrow\}$ for some $\Downarrow$, and "$\times$" means Cartesian product.

In the mentioned above paper, any $\mathsf{S}_{\mathbf{N}}$-system $\sigma : \underline{\sigma} \to \mathsf{S}_{\mathbf{N}}\,\underline{\sigma}$ is interpreted as a system with output $\mathbf{N}$ (see [1, Subsec. 4.2]), any $\mathsf{D}_{\mathbf{N}}$-system $\mathfrak{a} : \underline{\mathfrak{a}} \to \mathsf{D}_{\mathbf{N}}\,\underline{\mathfrak{a}}$ is interpreted as a detector of behavioural violations (see [1, Subsec. 4.3]), and interrelation between systems with output and detectors is established using the bifunctor combining a system and a detector into the system with termination

$$\mathsf{Join} : \mathbf{Sys}(\mathsf{S}_{\mathbf{N}}) \times \mathbf{Sys}(\mathsf{D}_{\mathbf{N}}) \to \mathbf{Sys}(\mathsf{T})$$

(see [1, Subsec. 5.1]). This bifunctor is defined as follows, let σ and $\mathfrak{a}$ be a system with output $\mathbf{N}$ and an $\mathbf{N}$-detector respectively, one can define the system with termination $\mathsf{Join}(\sigma, \mathfrak{a}) : \underline{\sigma} \times \underline{\mathfrak{a}} \to \mathsf{T}(\underline{\sigma} \times \underline{\mathfrak{a}})$ by the next rules

$$\mathsf{Join}(\sigma, \mathfrak{a}) = \lambda\ (x, y) \in \underline{\sigma} \times \underline{\mathfrak{a}}\ .\ \begin{cases} \Downarrow & \text{if } (\mathfrak{a}y)(\sigma_{\text{out}}\, x) = \Downarrow \\ \big\langle\, \sigma_{\text{tr}}\, x, (\mathfrak{a}y)(\sigma_{\text{out}}\, x) \big\rangle & \text{otherwise} \end{cases} \tag{1a}$$

Further for systems σ and τ with output $\mathbf{N}$ and $\mathbf{N}$-detectors $\mathfrak{a}$ and $\mathfrak{b}$, an $\mathsf{S}_{\mathbf{N}}$-morphism $f : \sigma \to \tau$, and a detector morphism $g : \mathfrak{a} \to \mathfrak{b}$, we define the mapping $\mathsf{Join}(f, g) : \underline{\sigma} \times \underline{\mathfrak{a}} \to \underline{\tau} \times \underline{\mathfrak{b}}$ by the formula

$$\mathsf{Join}(f, g) = f \times g. \tag{1b}$$

Above, we use the representation $\sigma\, x = \langle \sigma_{\text{out}}\, x, \sigma_{\text{tr}}\, x \rangle$.

Further in [1], it is proposed to describe the set of streams accepted by a detector $\mathfrak{a}$ as follows

– first of all for $\boldsymbol{s} \in \mathbf{N}^{\mathbb{N}}$, let us define the following system $[\boldsymbol{s}]$ with output namely

$$\underline{[\boldsymbol{s}]} = \big\{ \boldsymbol{s}_{k..} \mid k \in \mathbb{N} \big\} \quad \text{and} \quad [\boldsymbol{s}] = \lambda\ \boldsymbol{t} \in \underline{[\boldsymbol{s}]}\ .\ \big\langle\, \boldsymbol{t}\, 0, \boldsymbol{t}_{1..} \big\rangle;$$

– now for any $\mathbf{N}$-detector $\mathfrak{a}$ and $x \in \underline{\mathfrak{a}}$, let us define the following set

$$[\![\mathfrak{a}]\!]_x = \{ \boldsymbol{s} \in \mathbf{N}^{\mathbb{N}} \mid [\![\mathsf{Join}([\boldsymbol{s}], \mathfrak{a})]\!] \langle \boldsymbol{s}, x \rangle = \infty \}.$$

One of the main results obtained in [1, Lemmas 5 and 6] is established that any safety constraint has the described above form.

Below we use this approach for studying causality constraints in distributed systems.

5 Causality Constraints as a Kind of Safety Behavioural Ones

Anywhere in this section, C is an arbitrary but fixed finite set of clocks and $\mathbf{N}\mathsf{C}$ is the corresponding set of clock notifications that is the set of all non-empty subset of C. Informally, each message contains a list of clocks had ticked at the same time as the message was sent. Thus, notifications are no longer atomic entities or, in other words, first-class citizens. Now, clock ticks are atomic entities or, in other words, first-class citizens, whose combinations are notifications. In this section, we show how the results obtained in [1] should be adjusted for this more specific situation.

5.1 Clock and Schedules Coalgebraically

The logic clock model considers clock tick notification streams as the only information available about temporal order between events. These streams are called schedules in the model context. Thus, the coalgebraic meaning of schedules is related to the endofunctor $\mathsf{S}_{\mathbf{NC}} : \mathbf{Set} \to \mathbf{Set}$ defined like to the endofunctor $\mathsf{S}_{\mathbf{N}}$ from [1, Subsec. 4.2] namely

$\mathsf{S}_{\mathbf{NC}}\, X = \mathbf{NC} \times X$ for any object X of category $\mathbf{Set}$;

$\mathsf{S}_{\mathbf{NC}}\, f = \mathrm{id}_{\mathbf{NC}} \times f$ for any objects X and Y of category $\mathbf{Set}$ and a morphism $f : X \to Y$

then a system equipped with the clock set C is an $\mathsf{S}_{\mathbf{NC}}$-system.

In other words, a system equipped with a clock set C is a mapping $\sigma : X \to \mathsf{S}_{\mathbf{NC}}\, X$ that both specifies the state transition $\sigma_{\mathrm{tr}} : X \to X$ and the corresponding ensemble of clock ticks $\sigma_{\mathrm{out}} : X \to \mathbf{NC}$ of the system such that $\sigma = \langle \sigma_{\mathrm{out}}, \sigma_{\mathrm{tr}} \rangle$ is a universal arrow from X into $\mathbf{NC} \times X$.

A final $\mathsf{S}_{\mathbf{NC}}$-system $\boldsymbol{\nu}\mathsf{S}_{\mathbf{NC}}$ is defined on the set $\mathbf{NC}^{\mathbb{N}}$ of all schedules and has $\lambda\ \boldsymbol{s} \in \mathbf{NC}^{\mathbb{N}}\ .\ \boldsymbol{s}_{1..}$ as the transition function and $\lambda\ \boldsymbol{s} \in \mathbf{NC}^{\mathbb{N}}\ .\ \boldsymbol{s}\,0$ as the output function (see, [1, Subsec. 4.2]).

Due to the general results mentioned in [1, Sec. 3 and Subsec. 4.2], our capacity to formulate causality constraints for any system equipped with a clock set C are limited by statements in schedules terms only.

5.2 Detectors of Causality Constraint Violations

Detectors of causality constraint violations are described with using the endofunctor $\mathsf{D}_{\mathbf{NC}} : \mathbf{Set} \to \mathbf{Set}$ defined like to [1, Sec. 4.3] namely

$\mathsf{D}_{\mathbf{NC}}\, X = (1 + X)^{\mathbf{NC}}$ for any object X of category $\mathbf{Set}$;

$\mathsf{D}_{\mathbf{NC}}\, f = \lambda\ \phi \in (1 + X)^{\mathbf{NC}}\ .\ \lambda\ \boldsymbol{n} \in \mathbf{NC}\ .\ \begin{cases} \Downarrow & \text{if } \phi\,\boldsymbol{n} = \Downarrow \\ f(\phi\,\boldsymbol{n}) & \text{otherwise} \end{cases}$

for any objects X and Y of category $\mathbf{Set}$ and a morphism $f : X \to Y$.

Definitions of bifunctor Join, sets of the form $[\![\mathfrak{a}]\!]_x$ where $\mathfrak{a}$ is an $\mathbf{NC}$-detector, and $x \in \underline{\mathfrak{a}}$ are remained without changing.

6 Counter-based Detectors and Their Diophantine Representation

In this section, we consider a special class of $\mathbf{NC}$-detectors[2]. Some detectors of this class are used for defining semantics of Clock Constraint Specification Language (CCSL) [5].

In this section, we use the function $\mathrm{occ}_c : \mathbf{NC}^* \to \mathbb{N}$ associated with $c \in \mathsf{C}$ as follows

$$\mathrm{occ}_c = \lambda\ \boldsymbol{u} \in \mathbf{NC}^*\ .\ \begin{cases} 0 & \text{if } \boldsymbol{u} = \boldsymbol{\epsilon} \\ \mathrm{occ}_c\, \boldsymbol{u}' & \text{if } \boldsymbol{u} = \boldsymbol{u}'\boldsymbol{n} \text{ for } \boldsymbol{u}' \in \mathbf{NC}^*,\ \boldsymbol{n} \in \mathbf{NC}, \text{ and } c \notin \boldsymbol{n} \\ \mathrm{occ}_c\, \boldsymbol{u}' + 1 & \text{if } \boldsymbol{u} = \boldsymbol{u}'\boldsymbol{n} \text{ for } \boldsymbol{u}' \in \mathbf{NC}^*,\ \boldsymbol{n} \in \mathbf{NC}, \text{ and } c \in \boldsymbol{n} \end{cases}$$

[2] In this section, C is some clock set

This function counts how many times the corresponding clock ticked if the schedule begin coincides with the function argument.

6.1 Counter-based Detectors

Let $\mathsf{V} \subset \mathbf{NC} \times \mathbb{N}^{\mathsf{C}}$ then an $\mathbf{NC}$-detector $\mathfrak{c}_{\mathsf{V}}$ is called a counter-based $\mathbf{NC}$-detector if it is defined as follows

$$\underline{\mathfrak{c}_{\mathsf{V}}} = \mathbb{N}^{\mathsf{C}}$$
$$\mathfrak{c}_{\mathsf{V}} = \lambda\, x \in \mathbb{N}^{\mathsf{C}} \,.\, \lambda\, \boldsymbol{n} \in \mathsf{NC} \,.\, \begin{cases} \Downarrow & \text{if } \langle \boldsymbol{n}, x\rangle \in \mathsf{V} \\ x + \boldsymbol{n} & \text{otherwise} \end{cases}$$

where $x + \boldsymbol{n} = \lambda\, c \in \mathsf{C} \,.\, \begin{cases} (x\, c) + 1 & \text{if } c \in \boldsymbol{n} \\ x\, c & \text{otherwise} \end{cases}$.

Our nearest aim is to establish whether the family of causality constraints recognised by counter-based detectors is a family with a universal detector.

The first step for achieving the claimed aim is given by the next theorem.

Theorem 1. *Let P be a prefix-free subset of NC^+ then there exists $\mathsf{A} \subset \mathbf{NC} \times \mathbb{N}^{\mathsf{C}}$ such that $[\![\mathfrak{c}_{\mathsf{A}}]\!]x = P$ for some $x \in \underline{\mathfrak{c}_{\mathsf{A}}}$ if and only if P meets the following condition*

for any $\boldsymbol{v} \in \mathsf{NC}^+$,
$\boldsymbol{u}'\boldsymbol{v} \in P$ is equivalent to $\boldsymbol{u}''\boldsymbol{v} \in P$ whenever $\boldsymbol{u}', \boldsymbol{u}'' \in \mathsf{NC}^+$ are such that
$\boldsymbol{u}'_{0..m} \notin P$ and $\boldsymbol{u}''_{0..k} \notin P$ for all $0 < m \le |\boldsymbol{u}'|$, $0 < k \le |\boldsymbol{u}''|$ and $\operatorname{occ}_c \boldsymbol{u}' = \operatorname{occ}_c \boldsymbol{u}''$ for each $c \in \mathsf{C}$. (2)

For proving the theorem, we need some auxiliary results.

Lemma 1. *For any $\mathsf{V} \subset \mathbf{NC} \times \mathbb{N}^{\mathsf{C}}$ and $x \in \underline{\mathfrak{c}_{\mathsf{V}}}$, $[\![\mathfrak{c}_{\mathsf{V}}]\!]x$ meets condition (2).*

Proof. Let us assume $P = [\![\mathfrak{c}_{\mathsf{V}}]\!]x$ for some $\mathsf{V} \subset \mathbf{NC} \times \mathbb{N}^{\mathsf{C}}$ and $x \in \underline{\mathfrak{c}_{\mathsf{V}}}$ and take $\boldsymbol{u}', \boldsymbol{u}'' \in \mathsf{NC}^+$ such that $\boldsymbol{u}'_{0..m} \notin P$ and $\boldsymbol{u}'_{0..k} \notin P$ for all $0 < m \le |\boldsymbol{u}'|$, $0 < k \le |\boldsymbol{u}''|$ and $\operatorname{occ}_c \boldsymbol{u}' = \operatorname{occ}_c \boldsymbol{u}''$ for each $c \in \mathsf{C}$.
Then the first group of conditions ensures $\mathfrak{c}^+_{\mathsf{V}}(x, \boldsymbol{u}'), \mathfrak{c}^+_{\mathsf{V}}(x, \boldsymbol{u}'') \in \underline{\mathfrak{c}_{\mathsf{V}}}$ and the condition $\operatorname{occ}_c \boldsymbol{u}' = \operatorname{occ}_c \boldsymbol{u}''$ for each $c \in \mathsf{C}$ ensures $\mathfrak{c}^+_{\mathsf{V}}(x, \boldsymbol{u}') = \mathfrak{c}^+_{\mathsf{V}}(x, \boldsymbol{u}'')$.
Now, for an arbitrary $\boldsymbol{v} \in \mathsf{NC}^+$ and $0 < k \le |\boldsymbol{v}|$, we have (see [1, Lemma 1])

$$\begin{aligned} \mathfrak{c}^+_{\mathsf{V}}(x, \boldsymbol{u}'\boldsymbol{v}_{0..k}) &= \mathfrak{c}^+_{\mathsf{V}}(\mathfrak{c}^+_{\mathsf{V}}(x, \boldsymbol{u}'), \boldsymbol{v}_{0..k}) \\ &= \mathfrak{c}^+_{\mathsf{V}}(\mathfrak{c}^+_{\mathsf{V}}(x, \boldsymbol{u}''), \boldsymbol{v}_{0..k}) = \mathfrak{c}^+_{\mathsf{V}}(x, \boldsymbol{u}''\boldsymbol{v}_{0..k}) \end{aligned}$$

Thus, one can conclude $\boldsymbol{u}'\boldsymbol{v} \in P$ is equivalent to $\boldsymbol{u}''\boldsymbol{v} \in P$.

Proof (Proof of Theorem 1). Taking into account Lemma 1, we need only to prove that for any prefix-free $P \subset \mathbf{NC}$ satisfying condition (2), there exists

$\mathsf{V} \subset \mathbf{N}\mathsf{C} \times \mathbb{N}^{\mathsf{C}}$ such that $P = [\![\mathfrak{c}_{\mathsf{V}}]\!]x$ for some $x \in \underline{\mathfrak{c}_{\mathsf{V}}}$.
To do this let us take

$$\begin{aligned}\mathsf{V} = \{\langle \boldsymbol{n}, x\rangle \in \mathbf{N}\mathsf{C} \times \mathbb{N}^{\mathsf{C}} \mid \\ \boldsymbol{v}\boldsymbol{n} \in P \text{ for some } \boldsymbol{v} \in \mathbf{N}\mathsf{C}^* \text{ such that } x = \lambda\, c \in \mathsf{C}\,.\, \mathrm{occ}_c\, \boldsymbol{v}\}\end{aligned}$$

and check that $[\![\mathfrak{c}_{\mathsf{V}}]\!]z = P$ where $z = \lambda\, c \in \mathsf{C}\,.\,0$ i.e. that for any $\boldsymbol{u} \in \mathbf{N}\mathsf{C}^+$, $\boldsymbol{u} \in [\![\mathfrak{c}_{\mathsf{V}}]\!]z$ iff $\boldsymbol{u} \in P$.
Let us assume that $\boldsymbol{u} \in [\![\mathfrak{c}_{\mathsf{V}}]\!]z$.
If $\boldsymbol{u} = \boldsymbol{n}$ for some $\boldsymbol{n} \in \mathbf{N}\mathsf{C}$ then $\boldsymbol{n} \in [\![\mathfrak{c}_{\mathsf{V}}]\!]z$ means $(\mathfrak{c}_{\mathsf{V}} z)\,\boldsymbol{n} = \Downarrow$ i.e. $\boldsymbol{n} \in P$ because of $\lambda\, c \in \mathsf{C}\,.\,\mathrm{occ}_c\, \boldsymbol{v} = z$ iff $\boldsymbol{v} = \boldsymbol{\epsilon}$.
If $\boldsymbol{u} \in [\![\mathfrak{c}_{\mathsf{V}}]\!]z$ and $m = |\boldsymbol{u}| > 1$ then for $x_0 = z$, we have $x_{k+1} = (\mathfrak{c}_{\mathsf{V}} x_k)(\boldsymbol{u}\,k) \in \underline{\mathfrak{c}_{\mathsf{V}}}$ whenever $k = 0, \ldots, m-2$ and $(\mathfrak{c}_{\mathsf{V}} x_{m-1})(\boldsymbol{u}(m-1)) = \Downarrow$. In other words, $\langle \boldsymbol{u}\,k, x_k\rangle \notin \mathsf{V}$ where $k = 0, \ldots, m-2$ and $\langle \boldsymbol{u}(m-1), x_{m-1}\rangle \in \mathsf{V}$.
Thus, $u_{0..k} \notin P$ for any $0 < k \leq m-1$ but there exists $v \in \mathbf{N}\mathsf{C}^*$ such that $vu(m-1) \in P$ and $\lambda\, c \in \mathsf{C}\,.\,\mathrm{occ}_c\, \boldsymbol{v} = \lambda\, c \in \mathsf{C}\,.\,\mathrm{occ}_c\, \boldsymbol{u}_{0..m-1}$. But the condition $\boldsymbol{v}\boldsymbol{u}(m-1) \in P$ implies $\boldsymbol{v}_{0..l} \notin P$ for any $0 < l \leq |\boldsymbol{v}|$ due to P is a prefix free set. Thus, condition (2) ensures $\boldsymbol{u} = \boldsymbol{u}_{0..m-1}(\boldsymbol{u}(m-1)) \in P$.
Now assume $\boldsymbol{u} \in P$ and $m = |\boldsymbol{u}|$.
Then we have $\boldsymbol{u}_{0..k} \notin P$ for any k such that $0 < k < m$ and $\langle \boldsymbol{u}(m-1), \lambda\, c \in \mathsf{C}\,.\,\mathrm{occ}_c\, u_{0..m-1}\rangle \in \mathsf{V}$ due to the definition of V. One can conclude that $\langle \boldsymbol{u}k, \lambda\, c \in \mathsf{C}\,.\,\mathrm{occ}_c\, u_{0..k}\rangle \notin \mathsf{V}$ for any $0 \leq k < m-1$. Indeed, otherwise there exists $0 \leq k < m-1$ and $\boldsymbol{v} \in \mathbf{N}\mathsf{C}^*$ such that $\boldsymbol{v}(\boldsymbol{u}k) \in P$ and $\lambda\, c \in \mathsf{C}.\mathrm{occ}_c\, \boldsymbol{v} = \lambda\, c \in \mathsf{C}.\mathrm{occ}_c\, \boldsymbol{u}_{0..k}$. But condition (2) ensures $\boldsymbol{u}_{0..k}(\boldsymbol{u}k) = \boldsymbol{u}_{0..k+1} \in P$ for some $0 \leq k < m-1$. Thus $\mathfrak{c}^+_{\mathsf{V}}(z, \boldsymbol{u}_{0..k}) \in \underline{\mathfrak{c}_{\mathsf{V}}}$ for $0 < k < m$ and $\mathfrak{c}^+_{\mathsf{V}}(z, \boldsymbol{u}) = \Downarrow$. Therefore we have $\boldsymbol{u} \in [\![\mathfrak{c}_{\mathsf{V}}]\!]z$.

Corollary 1. *If $P = [\![\mathfrak{c}_{\mathsf{V}}]\!]x$ for some $\mathsf{V} \subset \mathbf{N}\mathsf{C} \times \mathbb{N}^{\mathsf{C}}$ and $x \in \underline{\mathfrak{c}_{\mathsf{V}}}$ then $P = [\![\mathfrak{c}_{\mathsf{V}'}]\!]z$ for some $\mathsf{V}' \subset \mathbf{N}\mathsf{C} \times \mathbb{N}^{\mathsf{C}}$ and $z = \lambda\, c \in \mathsf{C}\,.\,0$.*

Corollary 2. *The family of clock constraints being detected by counter-based detectors is a constraint family with a universal detector.*

Proof. To prove this fact, we use [1, Theorem 5]. Assume $P = [\![\mathfrak{c}_{\mathsf{V}}]\!]z$ for some $\mathsf{V} \subset \mathbf{N}\mathsf{C} \times \mathbb{N}^{\mathsf{C}}$.
Let $\boldsymbol{n} \notin P$, $\boldsymbol{u}'_{0..m}, \boldsymbol{u}''_{0..k} \notin \boldsymbol{n}^{-1} \cdot P$ for all $0 < m \leq |\boldsymbol{u}'|$ and $0 < k \leq |\boldsymbol{u}''|$, and $\mathrm{occ}_c\, \boldsymbol{u}' = \mathrm{occ}_c\, \boldsymbol{u}''$ for any $c \in \mathsf{C}$ then $\boldsymbol{n}\boldsymbol{u}'_{0..m} \notin P$ and $\boldsymbol{n}\boldsymbol{u}''_{0..k} \notin P$. Theorem 1 ensures $(\boldsymbol{n}\boldsymbol{u}')\boldsymbol{v} \in P$ and $(\boldsymbol{n}\boldsymbol{u}'')\boldsymbol{v} \in P$ are equivalent for any $\boldsymbol{v} \in \mathbf{N}\mathsf{C}^+$ i.e. $\boldsymbol{u}'\boldsymbol{v} \in \boldsymbol{n}^{-1} \cdot P$ and $\boldsymbol{u}''\boldsymbol{v} \in \boldsymbol{n}^{-1} \cdot P$ are equivalent.

Theorem 1 allows us to show the impossibility of confining ourselves to counters based detectors, as shown in the next example and the remark following after it.

Example 1. Let us take $\mathsf{C} = \{c_1^+, c_1^-, \ldots, c_m^+, c_m^-\}$ for some $m > 1$ and define the next $\mathbf{N}\mathsf{C}$-detector $\mathfrak{q}$

$$\underline{\mathfrak{q}} = \{c_1, \ldots, c_m\}^*$$

$$\mathfrak{q} = \lambda\ \boldsymbol{u} \in \{c_1, \ldots, c_m\}^*\ .\ \lambda\ \boldsymbol{n} \in \mathbf{N}\mathsf{C}\ . \begin{cases} \boldsymbol{u}\{c_k\} \text{ if } \boldsymbol{n} = \{c_k^+\} \\ \qquad\qquad \text{for some } k = 1, \ldots, m \\ \boldsymbol{u}' \quad \text{if } \boldsymbol{u} = \{c_k\}\boldsymbol{u}' \text{ and } \boldsymbol{n} = \{c_k^-\} \\ \qquad\qquad \text{for some } k = 1, \ldots, m \\ \Downarrow \quad \text{otherwise} \end{cases}$$

Also, let us take $P = [\![\mathfrak{q}]\!]\boldsymbol{\epsilon}$, and $\{c_i^+\}\{c_j^+\}, \{c_j^+\}\{c_i^+\} \in \mathbf{N}\mathsf{C}^+$ for any $1 \le i \ne j \le m$. Then it is evident $\mathfrak{q}^+(\boldsymbol{\epsilon}, \{c_i^+\}\{c_j^+\}) \ne \Downarrow$, $\mathfrak{q}^+(\boldsymbol{\epsilon}, \{c_j^+\}\{c_i^+\}) \ne \Downarrow$, and $\mathrm{occ}_{c_k^\pm}\{c_i^+\}\{c_j^+\} = \mathrm{occ}_{c_k^\pm}\{c_j^+\}\{c_i^+\}$ for all $k = 1, \ldots, m$. In other words, $\{c_i^+\} \notin P$, $\{c_i^+\}\{c_j^+\} \notin P$, $\{c_j^+\} \notin P$, $\{c_j^+\}\{c_i^+\} \notin P$, and $\mathrm{occ}_{c_k^\pm}\{c_i^+\}\{c_j^+\} = \mathrm{occ}_{c_k^\pm}\{c_j^+\}\{c_i^+\}$ for all $k = 1, \ldots, m$.
The assumption $P = [\![\mathfrak{c}_\mathsf{V}]\!]x$ for some $\mathsf{V} \subset \mathbf{N}\mathsf{C} \times \mathbb{N}^\mathsf{C}$ and $x \in \underline{\mathfrak{c}_\mathsf{V}}$ leads to the equivalence of the statements $\{c_i^+\}\{c_j^+\}\{c_i^-\} \in P$ and $\{c_j^+\}\{c_i^+\}\{c_i^-\} \in P$ due to Lemma 1.
But the definition of $\mathfrak{q}$ ensures $\{c_i^+\}\{c_j^+\}\{c_i^-\} \notin P$ and $\{c_j^+\}\{c_i^+\}\{c_i^-\} \in P$. This contradiction proofs that for any $\mathsf{V} \subset \mathbf{N}\mathsf{C} \times \mathbb{N}^\mathsf{C}$ and $x \in \underline{\mathfrak{c}_\mathsf{V}}$, $P = [\![\mathfrak{c}_\mathsf{V}]\!]x$ is false.

Remark 2. This abstract example has practical value. Indeed, let we have $m(m > 1)$ servers each of them provides access to the queue being stored by it. Then c_k^+ and c_k^- can be interpreted as append and pop operations respectively for the k^{th} queue.
Example 1 guarantees the behaviour like to a queue behaviour for the system of the servers cannot be ensured by a counter-based detector.

Remark 3. Reasoning similar to the reasoning given in Example 1 leads to the conclusion that the behaviour like to a stack for the system of the servers cannot be ensured by a counter-based detector.

6.2 Diophantine Representation of a Counter-Based Detector

The definition of a counter-based detector ensures that such a detector $\mathfrak{c}_\mathsf{V}$ is uniquely defined by the corresponding set $\mathsf{V} \subset \mathbf{N}\mathsf{C} \times \mathbb{N}^\mathsf{C}$, which below is referred as the ***defining set*** of a counter-based detector. Hence, to specify $\mathfrak{c}_\mathsf{V}$ we need to specify V. Taking into account that any tool for specifying a set can specify a recursively enumerable set only, we concentrate on counter-based detectors with the recursively enumerable determining set. Such a counter-based detector is below called a ***recursively defined counter-based detector***. Everywhere below we consider counter-based detectors with the recursively enumerable defining set only.

Note that a set $\mathsf{V} \subset \mathbf{N}\mathsf{C} \times \mathbb{N}^\mathsf{C}$ is recursively enumerable if and only if the sets $\mathsf{V}_{\boldsymbol{n}} = \{x \in \mathbb{N}^\mathsf{C} \mid \langle \boldsymbol{n}, x\rangle \in \mathsf{V}\}$ are recursively enumerable for all $\boldsymbol{n} \in \mathbf{N}\mathsf{C}$.

This claim follows directly from the finiteness of $\mathbf{NC}$. In other words, to specify a recursively enumerable set $\mathsf{V} \subset \mathbf{NC} \times \mathbb{N}^{\mathsf{C}}$ is equivalent to specify the set $\{\mathsf{V}_{\boldsymbol{n}} \mid \boldsymbol{n} \in \mathbf{NC}\}$ of recursively enumerable subsets of $\mathbb{N}^{\mathsf{C}}$.
Due to Matiyasevich-Davis-Robinson-Putnam theorem (see, [6]), the last is equivalent to specifying for any $\boldsymbol{n} \in \mathbf{NC}$ a polynomial $P_{\boldsymbol{n}}$ such that

$$\mathsf{V}_{\boldsymbol{n}} = \{x \in \mathbb{N}^{\mathsf{C}} \mid P_{\boldsymbol{n}}(x, y_1, \ldots, y_s) = 0 \text{ for some } y_1, \ldots, y_s \in \mathbb{N}\}.$$

Thus, any family of the species $\{P_{\boldsymbol{n}} \in \mathbb{Z}[x_1, \ldots, x_m, y_1, \ldots, y_s] \mid \boldsymbol{n} \in \mathbf{NC}\}$ where m is the cardinal number of C and s is some positive natural number specifies the set $\mathsf{V} \subset \mathbf{NC} \times \mathbb{N}^{\mathsf{C}}$ as follows
(1) let us select some enumeration $\{c_1, \ldots, c_m\}$ of clocks from C;
(2) let us associate natural variable x_i with clock c_i $(1 \leq i \leq m)$;
(3) for each $\boldsymbol{n} \in \mathbf{NC}$, let us specify polynomials $P_{\boldsymbol{n}} \in \mathbb{Z}[x_1, \ldots, x_m, y_1, \ldots, y_s]$;
(4) let us assume

$$\langle \boldsymbol{n}, x\rangle \in \mathsf{V} \quad \text{iff} \quad \exists\, y_1 \in \mathbb{N}; \ldots; y_s \in \mathbb{N},\ P_{\boldsymbol{n}}(x, y_1, \ldots, y_s) = 0.$$

Below we refer to this manner of specifying the defining set of a counter-based detector as to a ***Diophantine Representation*** of the detector.

Example 2 (of a Diophantine Representation). Let us assume $\mathsf{C} = \{c_1, c_2\}$ and consider the clock constraints specification defined as in Table 2. This specification determines that clock c_1 is strictly faster than clock c_2 and can be considered as a specification of the Lamport's relation "happen before" (see [7]).

Table 2. An example of a Diophantine Representation

c_1	c_2	$P(x_1, x_2, y_1)$	Diophantine set
0	1	$x_1 - x_2 + y_1$	$\{(x_1, x_2) \in \mathbb{N}^2 \mid x_1 \leq x_2\}$
1	0	$1 + x_1 - x_2 + y_1$	$\{(x_1, x_2) \in \mathbb{N}^2 \mid x_1 < x_2\}$
1	1	$x_1 - x_2 + y_1$	$\{(x_1, x_2) \in \mathbb{N}^2 \mid x_1 \leq x_2\}$

Note that the rows corresponding to the situation c_1 does not tick but c_2 ticks and also to the situation c_1 and c_2 tick at the same time are coincide in columns 3 and 4. Hence, we can represent Table 2 as follows

$$\begin{array}{ll} c_2 & \Rightarrow x_1 - x_2 + y_1 = 0 \\ c_1 \wedge \neg c_2 & \Rightarrow 1 + x_1 - x_2 + y_1 = 0 \end{array}$$

7 Universal Diophantine Simulator of Recursive Counter-Based Detectors

In this section, we use Diophantine Representation for describing a universal algorithm simulating any counter-based detector if its defining set is recursive.

The main advantage of this result is that it gives a metric for estimating the complexity of the detector and the detection problem. Namely, the number of additional variables and the power of the polynomial representing the corresponding Diophantine set can be used for defining such a metric. The detailed discussion of this problem one can find in [8].

Let us assume the the universal Diophantine simulator is set up for simulating the counter-based detector with the defining set specified by polynomials $\{P_{\boldsymbol{n}}(x_1,\ldots,x_m,y_1,\ldots,y_s) \mid \boldsymbol{n} \in \mathsf{NC}\}$ under assumption that the clock set is enumerated as follows $\mathsf{C} = \{c_1,\ldots,c_m\}$. Also, we choose some algorithm α enumerating the set $\mathbb{N}^s$ without repetitions. We mean below that $\alpha\, i$ is the i-th member of this enumeration.

We begin describing the universal Diophantine simulator of recursive counter-based detectors with specifying its composite structure (see Fig. 1). The simulator consists of one passive component (`NotificationLog`) and two active ones (`NotificationListener` and `NotificationHandler`).

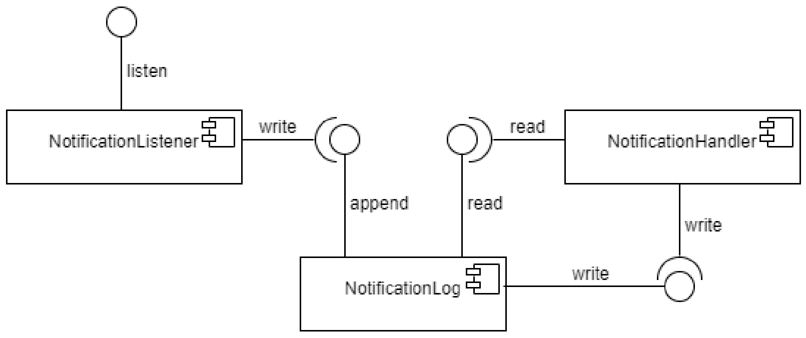

Fig. 1. Composite structure of the universal Diophantine simulator of recursive counter-based detectors

Component `NotificationLog` is a data store that allows to write data items in append and direct access modes and to read ones in direct access mode.

Component `NotificationListener` provides listening the notifications channel and writing the corresponding data items using the interface of component `NotificationLog` in accordance with the next algorithm.

```
NotificationListener:
(1) $x_k \leftarrow 0$ for $k = 1,\ldots,m$
(2) **wait on** receiving a notification
(3)     **let** the received notification be $\boldsymbol{n}$ **then** append to NotificationLog the item $(x_1,\ldots,x_m,\boldsymbol{n},0)$
(4)     $x_k \leftarrow x_k + 1$ for $k = 1,\ldots,m$ such that $c_k \in \boldsymbol{n}$
(5) **go to** (2)
```

Component `NotificationHandler` provides handling data items stored with `NotificationLog` for recognising behaviour violation in accordance with the following algorithm.

```
    NotificationHandler:
 (1) $i \leftarrow 0$
 (2) $j \leftarrow 0$
 (3) **try** to read $j$-th element from NotificationLog
 (4)   **if** the attempt is successful **then**
 (5)     **let** $(x_1, \ldots, x_m, \boldsymbol{n}, t)$ be the reading result
 (6)     $(y_1, \ldots, y_s) \leftarrow \alpha\, t$
 (7)     **if** $P_{\boldsymbol{n}}(x_1, \ldots, x_m, y_1, \ldots y_s) = 0$ **then**
           signal about the recognised violation and **terminate** the handling
 (8)     rewrite $(x_1, \ldots, x_m, \boldsymbol{n}, t+1)$ at $j$-th entry of NotificationLog
 (9)     $j \leftarrow j + 1$
(10)     **if** $j > i$ **then** $i \leftarrow i + 1$ and $j \leftarrow 0$
(11)     **go to** (3)
(12)   **else go to** (2)
```

Of course, the proposed universal Diophantine detector is not efficient. For providing efficiency a detector, we need to require the decidability of Diophantine problems related to the detector. For example, we can consider detectors with linear in $y_1, \ldots, y_s$ the related to them Diophantine problems. In this case, the complexity of a detector is determined by the complexity of generalized Euclid's algorithm finding the greatest common divisor of coefficients for the corresponding polynomials. Perhaps, other classes of decidable Diophantine problems give also detectors with the controllable complexity.

8 Conclusion

Summing up the mentioned above we can conclude that idea to use the coalgebraic approach for studying logical time in distributed systems gives some interesting results.

First of all, we see that the coalgebraic technique developed in [1] can be used in this more special case. Of course, in this case, we have a possibility enriching the model by using the specific structure of a notification set. It gives us to introduce a mechanism of counting clock ticks and to use values of the corresponding counters for formulating behavioural constraints.

Unfortunately, such an arithmetization does not provide specifying all reasonable causality constraints as it is demonstrated by Example 1 and Remark 3. But it provides a possibility to use a concept of Diophantine complexity [8] for classifying causality constraints of such a kind in accordance with the efficiency of detecting algorithms.

We need to note that assessing the efficiency of detecting algorithms requires answers the following questions

– How to build new detectors from simpler ones?
– Do such constructions possess the universality property?

– Can assess the complexity of the constructed detectors based on the complexity of their components?

These questions define the directions of our future research.

References

1. Zholtkevych, G., Labzhaniia, M.: Understanding Safety Constraints Coalgebraically. In: Computational Linguistics and Intelligent Systems, CEUR Workshop Proceedings, vol. 2604, pp. 1–19. CEUR-WS (2020)
2. Alpern, B., Schneider, F.: Recognizing safety and liveness. Distributed Computing 2, 117–126 (1987)
3. Mac Lane, S.: Categories for the Working Mathematician. Springer, 2nd edn. (2010)
4. Awodey, S.: Category Theory. Oxford University Press, 2nd edn. (2010)
5. André, C.: Syntax and Semantics of the Clock Constraint Specification Language (CCSL). Tech. Rep. 6925 version 2, Inria (June 2009), https://hal.inria.fr/inria-00384077/document
6. Matiyasevich, Y.: Hilbert's 10th Problem. The MIT Press (1993)
7. Lamport, L.: Time, Clocks, and the Ordering of Events in a Distributed System. CACM 21(7), 558–565 (1978)
8. Matiyasevich, Y.V.: Diophantine complexity. Journal of Soviet Mathematics 55, 1603–1610 (1991)