An Automata-Theoretic Approach to Constraint LTL [⋆]

Stéphane Demri [a], Deepak D'Souza [b]

[a] *LSV, CNRS & ENS Cachan & INRIA Futurs (projet SECSI), France*

[b] *Dept. of Computer Science & Automation, Indian Institute of Science, Bangalore, India*

Abstract

We consider an extension of linear-time temporal logic (LTL) with constraints interpreted over a concrete domain. We use a new automata-theoretic technique to show PSPACE decidability of the logic for the constraint systems $(\mathbb{Z}, <, =)$ and $(\mathbb{N}, <, =)$. Along the way, we give an automata-theoretic proof of a result of [1] when the constraint system satisfies the completion property. Our decision procedures extend easily to handle extensions of the logic with past-time operators and constants, as well as an extension of the temporal language itself to monadic second order logic. Finally we show that the logic becomes undecidable when one considers constraint systems that allow a counting mechanism.

Key words: temporal logic, logics of space and time, model-checking

1 Introduction

Constraint temporal logics. In classical Linear-time Temporal Logic (LTL) [3], properties of a program state are captured using propositions. These propositions could stand for properties like "the value of variable x is non-negative", or that "the value of x is less than that of y", or even that "the value of x in the current state is less than the value of y in the next state". In a natural extension of the logic, one may consider allowing assertions directly on the value of variables, as in "$x \geq 0$", "$x < y$", or "$x < Oy$". The type of variables and the kind of constraints allowed, leads us to the study of *constraint temporal logics.*

[⋆] A preliminary version of this paper appeared as [2].

In this paper we consider constraint temporal logics which are parameterised by a constraint system $\mathcal{D}$ which comprises a concrete domain and an interpretation for relations. The logic is essentially obtained from LTL by replacing propositions by atomic constraints in $\mathcal{D}$. In classical LTL variables represent propositions and models for its formulae are sequences of propositional valuations for these variables. These models can be viewed as having a "spatial" axis (here the elements *true* and *false*), along which the variables move. In constraint logics, the spatial axis for the models will comprise elements from the domain of $\mathcal{D}$. For example, with the constraint system $(\mathbb{N}, <, =)$ one is allowed to use atomic constraints involving $<$ and $=$, and variables which range over natural numbers. The formula $\Box(x < y)$ in the logic parameterised by $(\mathbb{N}, <, =)$ is interpreted over a sequence of $\mathbb{N}$-valuations for the variables x and y, and asserts that at every point in the future, the value of the variable x is less than the value of y. This formula is of course satisfiable, a candidate satisfying model being $ss\cdots$, where the valuation s assigns 1 to x and 2 to y.

Constraint temporal logics have been introduced and studied by logicians in the field of knowledge representation [4–8]. Spatio-temporal logics, as they are better known there, involve a hybrid of temporal logic and constraint systems, with varying degrees of interaction. For instance, one may be permitted to refer to the value of a variable x in the *next* time instant, leading to constraints of the form $(x < Ox)$. More generously, one may be permitted to refer to a future value of a variable, as in $(x < \Diamond y)$ which asserts that the current value of x is less than *some* future value of y.

While research in the area has focused mainly on logics involving constraint systems that have as domains intervals [9,1] and regions [10–12], with a variety of decidability and complexity results, there seems to have been little attention paid to commonly used constraint systems of the form $(D, <, =)$, with D as the integers $\mathbb{Z}$ or natural numbers $\mathbb{N}$. Comon and Cortier [13] consider a constraint system with the natural numbers $\mathbb{N}$ as the underlying domain and constraints of the form $x < Oy+2$, and show that the corresponding constraint temporal logic is undecidable. They identify a decidable fragment of the logic by restricting the use of the "Until" operator. Balbiani and Condotta [1] prove a general decidability result for constraint systems that satisfy a "completion" property and terms of the form $O \cdots Ox$. The completion property says that given a finite consistent set of constraints X, a valuation for a subset of variables which satisfies the constraints in X involving only those variables, can be extended to a valuation which satisfies all of X. The constraint system $(\mathbb{R}, <, =)$ satisfies this property, and the decidability of the constraint logic based on $(\mathbb{R}, <, =)$ in PSPACE follows from this result.

Our contribution. In this paper we concentrate on a constraint temporal logic called CLTL($\mathcal{D}$), for Constraint LTL parameterised by a constraint system $\mathcal{D}$, in which we restrict ourselves to terms of the form $O \cdots Ox$.

Our main technical contribution is a decision procedure for the satisfiability problem for the logic over the constraint systems $\mathcal{Z} = (\mathbb{Z}, <, =)$ and $\mathcal{N} = (\mathbb{N}, <, =)$. These logics are interesting from a technical point of view. As a typical illustration, the formula $\Box(x > Ox)$ has no (infinite) $\mathbb{N}$-models, while it does have $\mathbb{R}$-models and $\mathbb{Z}$-models. These constraint systems do not satisfy the completion property, and hence we cannot use the technique of [1] here. Further, in contrast to classical LTL, we show in Sec. 6 that the set of models $L(\varphi)$ of a formula φ in these logics is, in general, *not* ω-regular (i.e. it cannot be accepted by an automaton on infinite words). Nevertheless, we show that their satisfiability problems are decidable, and in fact PSPACE-complete. Our approach is automata-theoretic in that we associate with a given formula φ in CLTL($\mathcal{Z}$), an automaton $\mathcal{A}_\varphi^{\mathcal{Z}}$ which is non-empty iff φ has a $\mathbb{Z}$-model (and similarly for the $\mathcal{N}$ case). Roughly speaking, the technique used is as follows: we find an ω-regular superset M of $L(\varphi)$, which has the property that all its ultimately periodic words (those of the form $\tau \cdot \delta^\omega$) are also in $L(\varphi)$. This guarantees that M is non-empty iff $L(\varphi)$ is. We can now check the emptiness of M – using standard automata theory techniques – to decide the satisfiability of φ.

The second contribution of this paper arises from the fact that we use a general automata-theoretic approach which is also modular in nature. The approach is automata-theoretic in the sense of [14] in that we construct an automaton for a given formula whose language is non-empty if and only if the formula is satisfiable. It is modular in that the automaton we construct is the intersection of two separate automata, one corresponding to the underlying logical language, and the other for the constraint system under consideration. There are many benefits of such an approach and we hope these will be evident in the paper. In particular, this approach facilitates a transparent argument for the result of [1] for constraint systems satisfying the completion property. For such constraint systems $\mathcal{D}$ we can further show that the set of models of formulas in this logic is ω-regular (more precisely we speak here of the set of induced "frame sequences" or symbolic models). The modular approach also makes it easy to generalize the decision procedure in a transparent manner to handle past operators in LTL as well as to augment the underlying logical language from LTL to Büchi's monadic second order logic [15].

The third part of our contribution is that we establish decidability and complexity results for some natural extensions of Constraint LTL. We consider an extension which allows constants to be used in constraints, and show that for "dense" constraint systems like $(\mathbb{R}, <, =)$ and for the discrete constraint systems $(\mathbb{Z}, <, =)$ and $(\mathbb{N}, <, =)$, we can reduce the satisfiability problem to the standard case by eliminating the constants. We also introduce a class of constraint systems having a "counting" mechanism and we establish that the satisfiability problem for CLTL($\mathcal{D}$) becomes undecidable when $\mathcal{D}$ has a "counting" mechanism. In particular the logic for the constraint system $(\mathbb{N}, =, =_{+1},)$ is undecidable. Finally, we make use of our modular approach to show the decidability of the satisfiability and model-checking problems for MSO extensions of CLTL($\mathcal{D}$).

Applications. Our main motivation in this work has been to examine decidability issues related to a natural extension of LTL. However, there are some potential applications of the work here. The constraint logics considered here can be viewed as a convenient specification language for the behaviour of non-terminating programs. Valuation sequences can be thought of as snapshots of a program's variables at specific points of time during its execution. The work in this paper can be used to tell whether the given specification is realizable, and if so, to synthesize a program that meets the specification. For example, for a CLTL($\mathbb{Z}, <, =$) specification φ, our technique enables us to tell whether φ is satisfiable, and if it is, gives us an ultimately periodic frame sequence $\tau \cdot \delta^{\omega}$, along with a procedure to label it with integers, to obtain a model for the given specification. These finite frame sequences τ and δ, along with the labeling procedure, can be thought of as the required program whose behaviour satisfies the given specification.

Our modular automata-theoretic technique has also been useful in subsequent related work as detailed below, where it has been used as a fruitful starting point.

Related work. The analysis of the reachability problem for counter systems is ubiquitous in the verification of infinite-state systems [16–19] and this partly motivates the versions of constraint LTL introduced in this paper. On the logical side, temporal logics with Presburger constraints have been developed in [20,21,13,22], some of which have quite expressive decidable fragments. Spatio-temporal logics, see e.g. [12,23–25] are also examples of versions of LTL with constraint systems for which the constraint system has indeed a spatial structure. Examples of such spatial systems satisfying the completion property and those not satisfying this property can be found in [1] and [26]. Incidentally, a property similar to completion is used in Constraint Satisfaction Problems [27]. Complexity and decidability results for such logics can be found in [23]. In the introduction of concrete domains in description logics is due to [6] and since then, such logic-based formalisms for knowledge representation have been intensively studied, see e.g. [28,29]. Hence, LTL over concrete systems can be technically viewed as a subclass of description logics over concrete domains in which the models are linear. However, the introduction of LTL over Presburger constraints is motivated by the need to model-check counter systems whereas concrete domains in description logics have been introduced to integrate concrete qualities into description logic concepts.

The general approach presented in [2] and further developed in this long version, has been the starting point to further analyse temporal logics with constraint systems. Some of these are detailed below.

- Periodicity constraints are considered in [30] with applications for the automaton-based technique to define calendars.
- An extension of CLTL($\mathbb{N}, <, =$) with constants and periodicity constraints is shown in PSPACE in [31] generalizing the PSPACE upper bound of CLTL($\mathbb{N}, <, =$) established in this paper.
- The branching-time extension of [31] is explored in [32] that generalizes the decidability result.
- Complexity and decidability issues for fragments of LTL with Presburger constraints are analyzed in [33].
- A survey on Constraint LTL over Presburger constraints can be found in [34].

In [30,35–37], versions of Constraint LTL with the freeze operator that allow to express constraints of the form $(x < \Diamond y)$ are studied, leading to various complexity and decidability results. For instance, such an operator can be also encoded in logical formalisms for data words and XML documents [38] or for computations of systems with unboundedly many locations as resources [39], to quote a few examples.

Finally, the different variants of LTL over constraint systems presented in this paper can be viewed as fragments of first-order LTL where the domain of interpretation of variables is fixed as well as the interpretation of predicate symbols. Moreover, variables in CLTL($\mathcal{D}$) correspond to unary predicate symbols interpreted as singletons (the values of variables). However, CLTL($\mathcal{D}$) has no quantification over elements of the domain and in that sense it corresponds to a very weak first-order extension of plain LTL. A variant of first-order LTL has been introduced in [40] to verify data-driven web applications. The interplay between temporal operators and first-order quantifiers is also restricted since no quantification can occur in the scope of temporal operators, which guarantees better computational properties.

Plan of the paper. In the next couple of sections we define our logic and state its important relation to classical LTL. Sec. 4 addresses constraint systems which satisfy the completion property, and Secs. 5-7 address the satisfiability problem for the constraint systems $\mathcal{Z}$ and $\mathcal{N}$. In Sec. 9 & 10 we consider extensions to the logic. Though the emphasis in most of the paper is on the satisfiability problem for the logics, in Sec. 8 we also look at the model-checking problem for the logic CLTL($\mathcal{D}$).

2 Definitions

The set of natural numbers, integers, rationals, and reals, will be denoted by $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, and $\mathbb{R}$ respectively. An infinite word (or sequence) over an alphabet Σ is a function $\alpha : \mathbb{N} \to \Sigma$, written as $\alpha(0)\alpha(1)\cdots$. For an infinite word α, we will use $\alpha[i,j]$ to denote the finite word $\alpha(i)\alpha(i+1)\cdots\alpha(j)$, and $\alpha[i,\infty]$ to denote the infinite suffix $\alpha(i)\alpha(i+1)\cdots$. The set of all infinite words over Σ is denoted by Σ^ω. For a finite word τ and a finite or infinite word γ, we use $\tau \cdot \gamma$ to denote the concatenation of τ and γ, and τ^ω to denote the infinite word $\tau \cdot \tau \cdots$.

A *(concrete) constraint system* $\mathcal{D}$ is of the form $(D, R_1 \ldots, R_n, \mathcal{I})$, where D is a non-empty set referred to as the *domain*, and each R_i is a predicate symbol of arity a_i, with $\mathcal{I}(R_i) \subseteq D^{a_i}$ being its interpretation. We will suppress the mention of $\mathcal{I}$ whenever it is clear from the context. Let us fix such a constraint system $\mathcal{D}$ for the rest of this section.

An *(atomic) $\mathcal{D}$-constraint* over a finite set of variables U is of the form $R(x_1, \ldots, x_a)$, where R is a predicate symbol of arity a in $\mathcal{D}$ and each $x_i \in U$. A D-valuation over U is a map $s : U \to D$. We will use the notation $val_D(U)$ to denote the set of D-valuations over U. Let $c = R(x_1, \ldots, x_a)$ be a $\mathcal{D}$-constraint over U, and let s be a D-valuation over U. Then we say s satisfies c, written $s \models_{\mathcal{D}} c$, iff $(s(x_1), \ldots, s(x_a)) \in \mathcal{I}(R)$.

In the temporal logic we consider, we will allow terms of the form $O \cdots Ox$ to appear in an atomic constraint. The idea is that a term OOx, for example, will refer to the variable x, 2 steps ahead. Formally, an *(atomic) $\mathcal{D}$-term constraint* over a set of variables U, is of the form

$$R(O^{n_1}x_1, \ldots, O^{n_a}x_a)$$

where $x_1, \ldots, x_a \in U$, and $n_1, \ldots, n_a \in \mathbb{N}$. Here O^i stands for the juxtaposition of the next-state operator O i times, with O^0x representing just x. A $\mathcal{D}$-term constraint is interpreted over a sequence of D-valuations. A D-valuation sequence over a set of variables U is an infinite sequence σ of D-valuations over U. We say a D-valuation sequence σ over U satisfies a $\mathcal{D}$-term constraint $c = R(O^{n_1}x_1, \ldots, O^{n_a}x_a)$ over U, written $\sigma \models_{\mathcal{D}} c$, iff

$$(\sigma(n_1)(x_1), \ldots, \sigma(n_a)(x_a)) \in \mathcal{I}(R).$$

Let c be a $\mathcal{D}$-term constraint over a set of variables U. By the *O-length* of

c we will mean the value $i+1$ where i is the largest j for which O^j occurs in c. Thus the O-length of the term constraint $x < Oy$ is 2. The truth of a term constraint c in a valuation sequence σ is clearly determined completely by $\sigma[0, k-1]$, where k is the O-length of c.

A term constraint c over U of O-length k can also be viewed in a natural way as a constraint over the variables $U \times \{0, \ldots, k-1\}$, by identifying $O^i x$ with (x, i). Whenever it is convenient we will take the liberty of adopting this view.

We now introduce CLTL($\mathcal{D}$), the Constraint Linear-Time Temporal Logic parameterised by the constraint system $\mathcal{D}$. Let *Var* be a countably infinite set of variables, which we fix for the rest of this paper. The syntax of CLTL($\mathcal{D}$) is given by:

$$\varphi ::= c \mid \neg\varphi \mid (\varphi \vee \varphi) \mid O\varphi \mid (\varphi U \varphi)$$

where c is a $\mathcal{D}$ term constraint over the set of variables *Var*. The symbol 'O' is overloaded herein (used for $\mathcal{D}$-terms and as a temporal operator) but this will not cause any confusion.

Models for CLTL($\mathcal{D}$) formulas are D-valuation sequences over the variables *Var*. Let φ be a CLTL($\mathcal{D}$) formula, and σ be D-valuation sequence over *Var*. The satisfaction relation $\sigma \models \varphi$ is defined inductively below.

$\sigma \models c$ iff $\sigma \models_{\mathcal{D}} c$

$\sigma \models \neg\varphi$ iff $\sigma \not\models \varphi$

$\sigma \models \varphi \vee \psi$ iff $\sigma \models \varphi$ or $\sigma \models \psi$

$\sigma \models O\varphi$ iff $\sigma[1, \infty] \models \varphi$

$\sigma \models \varphi U \psi$ iff $\exists k \in \mathbb{N}$ such that $\sigma[k, \infty] \models \psi$ and $\forall i : 0 \leq i < k,\ \sigma[i, \infty] \models \varphi$.

We use the derived operators $\Diamond$ and $\Box$ which stand for "sometime" and "always", with the semantics $\Diamond\varphi \equiv (\top U \varphi)$, and $\Box\varphi \equiv \neg(\top U \neg\varphi)$.

Let $L^{\mathcal{D}}(\varphi)$ denote the set of models of a CLTL($\mathcal{D}$) formula φ. Thus $L^{\mathcal{D}}(\varphi) = \{\sigma \in (val_D(Var))^\omega \mid \sigma \models \varphi\}$.

Let $voc(\varphi)$ denote the finite set of variables that appear in φ. The truth of φ in a valuation sequence is thus determined completely by the values of the variables in $voc(\varphi)$, and so we can describe models of φ by restricting them to the variables in $voc(\varphi)$.

As an example, consider the constraint system $\mathcal{N} = (\mathbb{N}, <, =)$ with the usual interpretation of the symbols '$<$' and '$=$'. Let φ be the formula $\Box(x < Oy)$. Let us represent a valuation s restricted to $voc(\varphi) = \{x, y\}$ as $(s(x), s(y))$. Then φ is satisfied by the $\mathbb{N}$-valuation sequence $\sigma = (1,2)(2,4)(3,6)\cdots$, but not by $\sigma' = (1,1)(2,2)(2,2)\cdots$. It is visually convenient to describe a valuation sequence like σ using the notation below:

$$\begin{array}{l} y\ 2\ 4\ 6\ 8\ \cdots \\ x\ 1\ 2\ 3\ 4\ \cdots \end{array}$$

As another example, the valuation sequence below satisfies the formula $\Diamond(y < O^2 z) \wedge \neg\Box(y < O^2 z)$:

$$\begin{array}{l} z\ 8\ 0\ 3\ 2\ \cdots \\ y\ 5\ 1\ 0\ 1\ \cdots \\ x\ 1\ 0\ 4\ 4\ \cdots \end{array}$$

We recall briefly the definition of classical LTL. We are given a countable set of propositions P, and formulas are formed in much the same way as CLTL($\mathcal{D}$), except that instead of constraints c we have propositional assertions p from the set P. Models of LTL formulas are sequences of propositional valuations to variables in P. A propositional valuation over P can be represented as a subset of P, specifying the subset of propositions set to true. A propositional valuation sequence is thus a sequence over 2^P. The satisfaction relation $\alpha \models_{\mathrm{LTL}} \varphi$, for a propositional valuation sequence α and LTL formula φ, is given as follows. $\alpha \models_{\mathrm{LTL}} p$ iff $p \in \alpha(0)$, with the semantics of the remaining operators being the same as that of CLTL($\mathcal{D}$).

We can view the logic CLTL($\mathcal{D}$) as a generalization of classical LTL. LTL is equivalent to the constraint logic CLTL($\{0,1\}, \mathit{true}$) with $\mathcal{I}(\mathit{true}) = \{1\}$. A translation from LTL into CLTL($\{0,1\}, \mathit{true}$) consists of replacing the propositional variable p_i by $\mathit{true}(x_i)$. For instance,

$$\{p_2, p_3\} \cdot \{p_3\} \cdot \{p_1, p_3\} \ldots \models_{\mathrm{LTL}} \Diamond(p_1 \wedge p_3)$$

iff

$$\begin{array}{ll} x_3\ 1\ 1\ 1\ \cdots & \\ x_2\ 1\ 0\ 0\ \cdots & \models \Diamond(\mathit{true}(x_1) \wedge \mathit{true}(x_3)). \\ x_1\ 0\ 0\ 1\ \cdots & \end{array}$$

3 Semantics via classical LTL

The semantics of CLTL($\mathcal{D}$) formulas can also be given in terms of classical LTL. This view of the logic will play an important role in the developments in this paper.

Before we proceed, we will assume for convenience of notation that the formulas in CLTL($\mathcal{D}$) are over a fixed, *finite* set of variables V. This assumption involves no loss of generality as far as our decision procedures are concerned, since a given formula can speak of only a finite number of variables in Var. Henceforth, unless explicitly stated, we will assume valuations, valuation sequences, and term constraints, to be over the fixed finite set of variables V.

Next we introduce some required terminology. A *frame* over a set of variables U, w.r.t. a constraint system $\mathcal{D}$, is essentially a maximally consistent set of $\mathcal{D}$-constraints over U. More precisely, let $\mathit{frame}_{\mathcal{D}}(s, U)$ denote the (possibly empty) set of $\mathcal{D}$-constraints over U satisfied by a D-valuation s over U. Then a set of $\mathcal{D}$-constraints X is a *frame* over the variables U, w.r.t. $\mathcal{D}$, if there exists a D-valuation s over U such that $X = \mathit{frame}_{\mathcal{D}}(s, U)$.

In anticipation of a later need we define what we call the *frame checking problem* for a constraint system $\mathcal{D}$. This is the problem of deciding whether a given set of $\mathcal{D}$-constraints X, and a finite set of variables U, whether X is a frame over U, w.r.t. $\mathcal{D}$.

We now want to refer to the frame induced by a k-length segment of a valuation sequence over V, which we will call a k-frame. Let σ be a D-valuation sequence and let $k \in \mathbb{N}$. We use $\mathit{atc}(\mathcal{D}, k)$ to denote the set of all $\mathcal{D}$-term constraints over V, of O-length at most k. Let $k\text{-}\mathit{frame}_{\mathcal{D}}(\sigma)$ denote the set of all term constraints over V of O-length at most k, that are satisfied by σ. Thus $k\text{-}\mathit{frame}_{\mathcal{D}}(\sigma) = \{c \in \mathit{atc}(\mathcal{D}, k) \mid \sigma \models_{\mathcal{D}} c\}$. A *$k$-frame* w.r.t. $\mathcal{D}$ is a subset r of $\mathit{atc}(\mathcal{D}, k)$ such that $r = k\text{-}\mathit{frame}_{\mathcal{D}}(\sigma)$, for some D-valuation sequence σ.

As an example, consider the constraint system $\mathcal{N} = (\mathbb{N}, <, =)$, and the $\mathbb{N}$-valuation sequence σ of Sec. 2. We take V to be $\{x, y\}$. Then the following set is a 2-frame w.r.t. $\mathcal{N}$, and coincides with $2\text{-}\mathit{frame}_{\mathcal{N}}(\sigma)$:

$$\begin{aligned} \{\ & x < y, x < Ox, x < Oy, y < Oy, Ox < Oy, \\ & y = Ox, Ox = y, x = x, y = y, Ox = Ox, Oy = Oy\ \}. \end{aligned}$$

Extending the view of term constraints as constraints, a k-frame can be viewed as a frame over the set of variables $V \times \{0, \ldots, k-1\}$ in the expected manner.

We say a pair of k-frames (r, r') (w.r.t. $\mathcal{D}$) is *locally consistent*, if for all R (of say arity a), and for all $1 \leq n_1, \ldots, n_a \leq k-1$:

$$R(O^{n_1}x_1, \ldots, O^{n_a}x_a) \in r \text{ iff } R(O^{n_1-1}x_1, \ldots, O^{n_a-1}x_a) \in r'.$$

Accordingly, a k-frame sequence ρ will be said to be locally consistent if for all $i \in \mathbb{N}$, the pairs $(\rho(i), \rho(i+1))$ are locally consistent.

A given D-valuation sequence σ induces, for a given k, a canonical locally consistent k-frame sequence ρ, w.r.t. $\mathcal{D}$, denoted $k\text{-}fs_{\mathcal{D}}(\sigma)$, and given by $\rho(i) = k\text{-}frame_{\mathcal{D}}(\sigma[i,\infty])$. The sequence σ is indeed a concrete model of CLTL($\mathcal{D}$) whereas $k\text{-}fs_{\mathcal{D}}(\sigma)$ can be viewed as one of its symbolic representations by abstracting the values by $\mathcal{D}$-terms.

We extend k-fs to act on sets of valuation sequences in the natural way; thus $k\text{-}fs_{\mathcal{D}}(L) = \{k\text{-}fs_{\mathcal{D}}(\sigma) \mid \sigma \in L\}$. Conversely, a k-frame sequence ρ will be said to *admit* a D-valuation sequence, if there exists a D-valuation sequence σ such that $k\text{-}fs_{\mathcal{D}}(\sigma) = \rho$. In that case, the concrete model σ is said to realize the symbolic model ρ.

As an illustration, consider once again the constraint system $\mathcal{N}$, with $V = \{x\}$. The 2-frame sequence r^ω, where $r = \{x < Ox, x = x, Ox = Ox\}$, admits the $\mathbb{N}$-valuation sequence

$$x \; 0 \; 1 \; 2 \; 3 \cdots .$$

However the 2-frame sequence r^ω where $r = \{x > Ox, x = x, Ox = Ox\}$ does *not* admit any $\mathbb{N}$-valuation sequence, since we cannot have an infinite strictly decreasing sequence of natural numbers.

We say a CLTL($\mathcal{D}$) formula φ has O-length k if the largest O-length of term constraints in φ is k. Thus the O-length of the formulae $\Box(O(x < Oy))$ and $\Box(x < Oy)$ is 2.

Consider now a CLTL($\mathcal{D}$) formula φ, and let k be its O-length. We can view φ as an LTL formula over the set of "propositions" $atc(\mathcal{D}, k)$, by viewing each term constraint c that appears in φ syntactically as a proposition p_c, or just c for notational convenience. Thus, for a sequence $\rho \in (2^{atc(\mathcal{D},k)})^\omega$, we have the (symbolic) satisfaction relation $\rho \models_{\mathrm{LTL}} \varphi$ defined by the usual semantics of LTL. We use $L_{\mathrm{LTL}}(\varphi)$ to denote the set

$$\{\rho \in (2^{atc(\mathcal{D},k)})^\omega \mid \rho \models_{\mathrm{LTL}} \varphi\}.$$

The following proposition is now a direct consequence of the semantics of CLTL($\mathcal{D}$) and LTL, and summarizes the link between the two logics:

Lemma 3.1 *Let φ be a* CLTL($\mathcal{D}$) *formula of O-length k. Let σ be a D-valuation sequence, and let $\rho = k\text{-}fs_{\mathcal{D}}(\sigma)$. Then $\sigma \models \varphi$ iff $\rho \models_{\mathrm{LTL}} \varphi$.* $\Box$

For a formula φ in CLTL($\mathcal{D}$) of O-length k, let $L^{\mathcal{D}}_{fs}(\varphi)$ denote the set of k-frame sequences induced by the valuation sequence models of φ. That is

$$L^{\mathcal{D}}_{fs}(\varphi) = \{k\text{-}fs_{\mathcal{D}}(\sigma) \mid \sigma \models \varphi\}.$$

In the light of Lemma 3.1, $L^{\mathcal{D}}_{fs}(\varphi)$ is the set of all k-frame sequences (w.r.t.

$\mathcal{D}$) that satisfy φ as an LTL formula, *and* admit a D-valuation sequence. Equivalently, $L^{\mathcal{D}}_{fs}(\varphi)$ contains the symbolic models having a concrete model satisfying φ.

4 A general decidability result

Let $\mathcal{D} = (D, R_1, \ldots, R_n)$ be a constraint system. The satisfiability problem for CLTL($\mathcal{D}$) is: given a CLTL($\mathcal{D}$) formula φ, does there exist a D-valuation sequence which satisfies φ? In other words, is $L^{\mathcal{D}}(\varphi) \neq \emptyset$? (equivalently, is $L^{\mathcal{D}}_{fs}(\varphi) \neq \emptyset$?).

From Lemma 3.1 we have the following corollary.

Corollary 4.1 *A* CLTL($\mathcal{D}$) *formula φ of O-length k is satisfiable iff there exists a locally consistent k-frame sequence which (symbolically) satisfies φ as a classical* LTL *formula,* and *which admits a D-valuation sequence, i.e. a concrete model.*

This suggests a way to answer the satisfiability problem for the logic. Recall the automata-theoretic approach to solving the satisfiability problem for classical LTL [41]. One builds an automaton $\mathcal{A}^{\mathrm{LTL}}_{\varphi}$ which accepts precisely the models of the given formula ϕ. This automaton can then be checked for nonemptiness to decide the satisfiability of the given formula. In our case we can build the formula automaton $\mathcal{A}^{\mathrm{LTL}}_{\varphi}$ corresponding to the LTL version of the given formula φ. If we could now construct automata that

(1) filter out k-frame sequences that are not locally-consistent and
(2) filter out k-frame sequences that do not admit D-valuations;

then we can intersect them and check the resulting automaton for nonemptiness to decide the satisfiability of φ. This approach has structural similarities with [42] where nonemptiness of timed languages is checked by considering untimed languages.

As we will see in Sec. 6, there are constraint systems for which (2) above is not possible, and one has to adopt a slightly different approach for them. However, there are constraint systems for which it suffices to ensure (1), and (2) is automatically taken care of. Constraint systems that satisfy a "completion" property are one class of such constraint systems. In the rest of this section we elaborate on these ideas.

A constraint system $\mathcal{D}$ is said to satisfy the *completion* property if, essentially, given a consistent set of constraints X over a set of variables U, any partial valuation which respects the constraints in X involving only the assigned variables, can be extended to a valuation which respects all the constraints in X. More precisely, for a set of constraints X and a subset of variables U, let us use the notation $X \restriction U$ to denote the set of constraints in X which use only variables in U. Then, the constraint system $\mathcal{D}$ satisfies *completion* if given

- a frame X over a finite set of variables U w.r.t. $\mathcal{D}$,
- a subset U' of U, and
- a valuation s' over U', such that $\mathit{frame}_{\mathcal{D}}(s', U') = X \restriction U'$.

there exists a valuation s over U which extends s' and satisfies $\mathit{frame}_{\mathcal{D}}(s, U) = X$.

The constraint system $(\mathbb{R}, <, =)$ is one example of a system that satisfies the completion property. We will prove this fact in the next section where we show that the completion property is closely related to "denseness" of the domain.

An example of a constraint system which does not satisfy the completion property is $(\mathbb{Z}, <, =)$, since for the set of constraints $X = \{x < y, x < z, z < y, x = x, y = y, z = z\}$ over the set of variables $U = \{x, y, z\}$, the partial valuation $s : x \mapsto 0, y \mapsto 1$ satisfies the constraints in X involving x and y, but cannot be extended to a valuation which satisfies the constraints $x < z$ and $z < y$ in X.

We now observe that:

Lemma 4.2 *Let $\mathcal{D}$ be a constraint system which satisfies the completion property. Then every locally consistent k-frame sequence w.r.t. $\mathcal{D}$ admits a D-valuation sequence.*

Proof Let $\rho = r_0 r_1 \ldots$ be a locally consistent k-frame sequence w.r.t. $\mathcal{D} = (D, R_1, \ldots, R_n)$. We define a D-valuation sequence $\sigma = s_0 s_1 \cdots$ with the property that $\rho = k\text{-}\mathit{fs}_{\mathcal{D}}(\sigma)$.

We take the liberty of viewing each r_i as a frame over the variables $U_i = V \times \{i, \ldots, i+k-1\}$. Since each r_i is a k-frame, for each i there exists a D-valuation t_i over U_i such that $r_i = \mathit{frame}_{\mathcal{D}}(t_i, U_i)$. For each $i \in \{0, \ldots, k-1\}$ and $x \in V$, we define $s_i(x) = t_0(x, i)$. Now suppose that s_i is defined for $0 \leq i \leq j$ with $j \geq k-1$. Let us define s_{j+1}. Consider the set of constraints r_{j+1}. We know that r_{j+1} is a frame over U_{j+1}. Further consider the subset $U' = V \times \{j+1, \ldots, j+k-1\}$ of U_{j+1}. The restriction t' of the valuation t_j to U', is such that $r_{j+1} \restriction U' = \mathit{frame}_{\mathcal{D}}(t', U')$. Thus, by the completion property we have a D-valuation t'' which extends t' to the variables $V \times \{j+1, \ldots, j+k\}$, and satisfies $r_{j+1} = \mathit{frame}_{\mathcal{D}}(t'', U_{j+1})$. We can now define $s_{j+1}(x) = t''(x, j+1)$ for each $x \in V$.

The sequence $\sigma = s_0 s_1 \cdots$ can be completely defined and σ has the property that $k\text{-}fs_{\mathcal{D}}(\sigma) = \rho$. Hence ρ admits the D-valuation sequence σ.

$\square$

Here is an interesting property of the language of k-frames defined by a CLTL($\mathcal{D}$) formula, which we had alluded to in the beginning of this section. Recall that a Büchi automaton over an alphabet Σ is simply a classical automaton $\mathcal{A} = (Q, q_0, \longrightarrow, F)$ over Σ, but with the set of final states F used as an acceptance condition on infinite words $\alpha \in \Sigma^\omega$. A run $\rho : \mathbb{N} \rightarrow Q$ on α is accepting if $\rho(i) \in F$ for infinitely many $i \in \mathbb{N}$. $L(\mathcal{A})$ is the set of infinite words accepted by $\mathcal{A}$, and $L \subseteq A^\omega$ is termed ω-regular if $L = L(\mathcal{A})$ for some Büchi automaton $\mathcal{A}$ over Σ. The class of w-regular languages are known to be effectively closed under the boolean operations of union, intersection, and complement. Further, the nonemptiness problem for these automata is easily decidable, being essentially a graph reachability problem. We refer the reader to [43] for further details.

Lemma 4.3 *Let $\mathcal{D}$ be a constraint system satisfying the completion property. Let φ be a* CLTL($\mathcal{D}$) *formula. Then $L^{\mathcal{D}}_{fs}(\varphi)$ is ω-regular.*

Proof Let φ be of O-length k. We define a Büchi automaton $\mathcal{A}^{\mathcal{D}}_{\varphi}$ over the alphabet $2^{atc(\mathcal{D},k)}$, such that $L(\mathcal{A}^{\mathcal{D}}_{\varphi}) = L^{\mathcal{D}}_{fs}(\varphi)$. Define $\mathcal{A}^{\mathcal{D}}_{\varphi} = \mathcal{A}^{\mathrm{LTL}}_{\varphi} \cap \mathcal{A}^{\mathcal{D},k}_{lc}$ (i.e. the automaton which accepts the intersection of the languages of the two automata), where $\mathcal{A}^{\mathrm{LTL}}_{\varphi}$ is the Vardi-Wolper automaton [41] for the LTL formula φ, and $\mathcal{A}^{\mathcal{D},k}_{lc}$ is a Büchi automaton over $2^{atc(\mathcal{D},k)}$, defined below, which accepts locally consistent k-frame sequences.

Define $\mathcal{A}^{\mathcal{D},k}_{lc} = (Q, q_0, \longrightarrow, F)$ where Q is the set of k-frames w.r.t. $\mathcal{D}$, along with a separate start state q_0; $\longrightarrow$ is given by $q_0 \xrightarrow{r} r$, and $r \xrightarrow{r'} r'$ iff (r, r') is locally consistent (here r and r' range over k-frames w.r.t. $\mathcal{D}$); and $F = Q$. $\square$

Thus, if $\mathcal{A}^{\mathcal{D}}_{\varphi}$ can be constructed effectively, Lemma 4.3 gives us a way to decide the satisfiability problem for a constraint system $\mathcal{D}$ which satisfies the completion property: construct $\mathcal{A}^{\mathcal{D}}_{\varphi}$ and check it for nonemptiness. We know that $\mathcal{A}^{\mathrm{LTL}}_{\varphi}$ can indeed be constructed [41]. However, whether automaton $\mathcal{A}^{\mathcal{D},k}_{lc}$ can be constructed effectively depends on the decidability of the frame-checking problem for $\mathcal{D}$, since the transition relation of the automaton must be computable and for this one needs to recognize whether a set of constraints constitutes a frame.

In particular, if $\mathcal{D}$ is such that frame checking can be done in PSPACE (in the size of the frame), then the satisfiability problem for CLTL($\mathcal{D}$) can be solved in PSPACE in the size of the given formula. The automata $\mathcal{A}^{\mathrm{LTL}}_{\varphi}$ and $\mathcal{A}^{\mathcal{D},k}_{lc}$ have sizes exponential in the length of φ. However, they are both implicitly defined graphs whose adjacency relation can be checked in PSPACE in the length of φ. For $\mathcal{A}_{lc}^{\mathcal{D},k}$ we need the assumption that frame checking for $\mathcal{D}$ can be done in PSPACE. Now checking emptiness of the language accepted by a Büchi automaton essentially requires checking whether there is a final state reachable from the initial state, which is again reachable from itself. It is well-known that reachability on an n-node graph can be done non-deterministically in NLOGSPACE. Thus, emptiness checking for $\mathcal{A}_{\varphi}^{\mathcal{D}}$ can be done non-deterministically in PSPACE in the length of φ. Since deterministic and non-deterministic PSPACE coincide, we can conclude that it can be done deterministically in polynomial space. This is essentially the result of [1], presented in an automata-theoretic manner.

Theorem 4.4 ([1]) *The satisfiability problem for* CLTL($\mathcal{D}$) *when $\mathcal{D}$ satisfies completion and allows frame-checking in* PSPACE, *is in* PSPACE. $\square$

Finally, if $\mathcal{D}$ is *non-trivial* in the sense that at least one of the relations R is not $\emptyset$ or D^a (where a is the arity of R), the satisfiability problem of classical LTL (which is known to be PSPACE-hard [44]) can be reduced to the satisfiability problem for CLTL($\mathcal{D}$). We can do this by translating an LTL formula ψ into a CLTL($\mathcal{D}$) formula φ as follows. With each proposition p we associate a set of distinct variables $\{x_1^p, \ldots, x_a^p\}$. Further, these sets are disjoint for each pair of propositions. We now replace each proposition p in ψ by $R(x_1^p, \ldots, x_a^p)$. The formula ψ is now LTL satisfiable iff φ is CLTL($\mathcal{D}$) satisfiable.

To summarize:

Theorem 4.5 *The satisfiability problem for* CLTL($\mathcal{D}$) *when $\mathcal{D}$ is non-trivial, satisfies completion, and allows frame-checking in* PSPACE, *is* PSPACE*-complete.* $\square$

Examples of constraint systems satisfying the completion property can be found in Corollary 5.4 (including $(\mathbb{R}, <, =)$, $(\mathbb{Q}, <, =)$ and $(D, =)$ for any infinite set D).

5 Domains of the form $(D, <, =)$

In this section we consider domains of the form $(D, <, =)$, where D is an infinite set, '$<$' is any strict total ordering on D, and '$=$' is the equality relation. We introduce a natural way of representing frames over such domains as labelled directed graphs. This representation will play a useful role in subsequent arguments in this paper. We then give a characterisation of such domains that satisfy the completion property.

For a constraint system of the form $(D, <, =)$ it is convenient to visualise a frame as a labelled, directed graph. For the rest of this section let us fix a constraint system $\mathcal{D} = (D, <, =)$ with D an infinite set, and '$<$' a strict total ordering on D. We represent a frame X over a finite set of variables U by a $\{<, =\}$-labelled, directed graph G_X over the vertices U, where we place a '$\sim$'-labelled edge (for $\sim \in \{<, =\}$) from x to y precisely when $x \sim y \in X$.

Such a graph clearly satisfies the conditions that:

(1) There is an edge between every pair of vertices;
(2) If there is a '$=$'-labelled edge from x to y then there is also one from y to x;
(3) There are no *strict* cycles – i.e. directed cycles containing a '$<$'-labelled edge.

We call such a graph (i.e. a $\{<, =\}$-labelled directed graph over a set of vertices U which satisfies the conditions 1–3 above) a *frame graph* over U.

Conversely, given a frame graph H over a finite set of vertices U, the set of constraints $\mathit{gtof}(H) = \{x \sim y \mid x \xrightarrow{\sim} y \text{ in } H\}$ constitutes a frame over U. To see this, observe that we can label the vertices of H by natural numbers via a labelling $l : U \to \mathbb{N}$, which respects the edges in H – i.e. if $u \xrightarrow{\sim} v$ then $l(u) \sim l(v)$. The map l is said to be an edge-respecting labelling. This follows since we can quotient H by equating vertices joined by edges labelled by '$=$' to get a DAG (in fact a total order) on U, which can then be linearized. Since U is finite, this gives us a natural edge-respecting labelling l of U by numbers in $\mathbb{N}$. From a finite non-decreasing sequence of natural numbers we can always obtain a corresponding (order preserving) sequence in D, which gives us an edge-respecting D-labelling l' of H. Viewing l' as a D-valuation over U, it is immediate that $\mathit{frame}(l', U) = \mathit{gtof}(H)$. Thus $\mathit{gtof}(H)$ is a frame.

Frame graphs are thus a faithful representation of frames in the following sense:

Lemma 5.1 *Let X be a frame over U w.r.t. $\mathcal{D}$. Then*

(1) G_X is a frame graph over U.
(2) If H is a frame graph over U then $\mathit{gtof}(H)$ is a frame over U.
(3) $\mathit{gtof}(G_X) = X$.
(4) A D-valuation s over U is such that $\mathit{frame}(s, U) = X$ iff s is an edge-respecting labelling of G_X. $\square$

Observe that the frame-checking problem for $\mathcal{D}$ amounts to checking whether the graph corresponding to the given frame satisfies the conditions 1–3 in the definition of a frame graph above. Since cycle detection in a directed graph can be done in NLOGSPACE, we conclude that the frame checking problem for $\mathcal{D}$ is in NLOGSPACE.

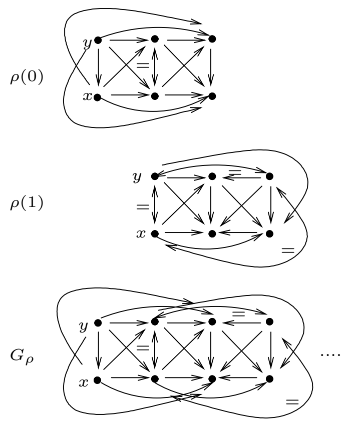


Fig. 1. A 3-frame sequence over the variables $\{x, y\}$

Extending the representation of frames as frame graphs, a locally consistent k-frame sequence ρ can be represented as a single $\{<, =\}$-labelled, directed graph G_ρ. The graph G_ρ is essentially the superimposition (in an overlapping manner) of the graphs corresponding to the successive k-frames. Fig. 1 shows the first two 3-frames of a locally consistent k-frame sequence represented as a single graph. In the figure we have left out self-loops on the vertices, as well as '$<$' labels on edges, for brevity. More precisely, G_ρ has $V \times \mathbb{N}$ as its set of vertices, and an edge $(x, i) \xrightarrow{\sim} (y, j)$ ($\sim \in \{<, =\}$) iff either $i \leq j$ and $(x \sim O^{j-i}y) \in \rho(i)$, or, $i > j$ and $(O^{i-j}x \sim y) \in \rho(j)$.

In a natural way, a D-valuation sequence σ can be viewed as a D-labelling l of the vertices of G_ρ by defining $l(x, i) = \sigma(i)(x)$. Conversely, a D-labelling l of G_ρ gives us a D-valuation sequence σ given by $\sigma(i)(x) = l(x, i)$. It thus follows that:

Lemma 5.2 *A locally consistent k-frame sequence ρ admits a D-valuation sequence σ iff σ is an edge-respecting D-labelling of G_ρ.*

We now show that the completion property for domains of the form $(D, <, =)$ is strongly related to denseness of the domain. As usual we say D is *dense* (with respect to the ordering $<$) if for each d, d' in D with $d < d'$, there exists d'' in D, such that $d < d'' < d'$. We say D is *open* iff for each $d \in D$, there exist $d', d'' \in D$ with $d' < d < d''$.

Lemma 5.3 *Let $\mathcal{D}$ be a constraint system of the form $(D, <, =)$ where D is infinite and $<$ is a total order. Then, $\mathcal{D}$ satisfies completion iff $\mathcal{D}$ is dense and open.*

Proof We will make use of the representation of frames as graphs here.

Let $\mathcal{D}$ satisfy completion. Then D must be dense and open. To see this consider the frame $X = \{x < y, y < z, x < z\}$, over the variables $U = \{x, y, z\}$ (we have suppressed the mention of the identity constraints $x = x$ etc, for brevity).

Let $d, d' \in D$ with $d < d'$. Then the labelling l' over $U' = \{x, z\}$ given by (d, d') is an edge-respecting labelling of G_X restricted to U'. By the completion property, we should be able to extend l' to an edge-respecting D-labelling l over U. We thus have an element $l(z) \in D$ with $d < l(z) < d'$. Similarly for openness, given a $d \in D$ we can consider the trivial edge-respecting labelling of the single vertex y, which assigns d to y. This labelling must be extendable to a labelling of U, which gives us elements $l(x)$ and $l(z)$ of D satisfying $l(x) < d < l(z)$.

Conversely, let D be open and dense. We show that it satisfies the completion property. Let X be a frame over a finite set of variables U. Let U' be a non-empty subset of U, and l' an edge-respecting labelling for the subgraph of G_X induced by U'. Let x be a vertex in $U \setminus U'$. We will label x with $d \in D$, such that the resulting labeling $l = s' \cup \{(x, d)\}$ of $U' \cup \{x\}$ is an edge-respecting labelling of the subgraph induced by $U \cup \{x\}$. This will suffice to prove the completion property, since this procedure can be repeatedly applied to obtain a labelling of U which extends l'.

Let $X_{=x}$ be the set of vertices in U' to which there is an '$=$'-labelled edge from x in G_X; $X_{<x}$ the set of vertices in U' from which there is a '$<$'-labelled edge to x; and $X_{x<}$ the set of vertices in U' to which there is a '$<$'-labelled edge from x. Let $p = \max\{l'(y) \mid y \in X_{<x}\}$ and $q = \min\{l'(z) \mid z \in X_{x<}\}$ (we will use the values of p and q only when the sets $X_{<x}$ and $X_{x<}$, are respectively non-empty.) Note that p and q are values in D, and by denseness and openness of D there exist elements p', q', r in D such that $p < p'$, $q' < q$ and $p < r < q$.

We label x as follows:

(1) if $X_{=x}$ is non-empty, label x by $l'(y)$ for any $y \in X_{=x}$,
(2) if $X_{<x}$ is non-empty and both $X_{=x}$ and $X_{x<}$ are empty, label x by p',
(3) if $X_{x<}$ is non-empty and both $X_{=x}$ and $X_{<x}$ are empty, label x by q'.
(4) if $X_{<x}$ and $X_{x<}$ are non-empty and $X_{=x}$ is empty, label x by r.

To argue that d is an acceptable label for x, suppose to the contrary, that it was not. Then there must exist a vertex y in U' and an edge between x and y which is inconsistent with the labels of x and y. Two cases arise here: We have $x \xrightarrow{<} y$ but $l(x) \geq l(y)$ (and the symmetric case of $y \xrightarrow{<} x$), or $x \xrightarrow{=} y$ and $l(x) \neq l(y)$.

For the first case, it must have been the case that x was labelled by clause 1. Thus there must be another vertex z in U' with $x \xrightarrow{=} z$ in G_X, and $l(x) = l(z)$. But since l' was an edge-respecting labelling, we must have $y \xrightarrow{<} z$ or $y \xrightarrow{=} z$ in G_X. This gives us a strict cycle $x \xrightarrow{<} y \xrightarrow{<} z \xrightarrow{=} x$ or $x \xrightarrow{<} y \xrightarrow{=} z \xrightarrow{=} x$ in G_X, which is a contradiction.

For the second case to have arisen, it must again be the case that x was labelled by clause 1. Thus there is a vertex z such that $x \xrightarrow{=} z$ and $l(z) = l(x)$. Once again this gives us a strict cycle in G_X which is a contradiction. $\square$

It now follows that:

Corollary 5.4 *The following constraint systems are non-trivial and satisfy the completion property [1]:*

(1) $(\mathbb{R}, <, =)$, $(\mathbb{Q}, <, =)$.
(2) $(D, <, =)$ *where* D *is any dense open subset of* $\mathbb{R}$.
(3) $(D^n, \prec, =)$ *for* $n \geq 1$, *and* $D \in \{\mathbb{Q}, \mathbb{R}\}$, *where* $\prec$ *is for instance a lexicographical ordering based on* $(D, <, =)$.
(4) $(D, =)$ *for any infinite set* D. $\square$

It now follows from Theorem 4.5 that:

Corollary 5.5 *The satisfiability problem for* CLTL($\mathcal{D}$) *for each of the constraint systems* $\mathcal{D}$ *in Corollary 5.4 (in particular* $(\mathbb{R}, <, =)$*) is* PSPACE*-complete.* $\square$

Constraint systems satisfying the completion property can be found in [9,45]. Other constraint domains used in spatio-temporal logics as spatial domains lead to a PSPACE upper bound, see e.g. [23, Chapter 16] and [25].

Finally, we point out an important property of G_ρ which is that it does not contain any strict cycles:

Corollary 5.6 *Let* $\mathcal{D} = (D, <, =)$ *be such that* $<$ *is a strict total order on* D. *Then, for a locally consistent* k*-frame sequence* ρ *w.r.t.* $\mathcal{D}$, *the graph* G_ρ *has* no *strict cycles.*

Proof It is not difficult to see that a k-frame with respect to $\mathcal{D}$ is also one w.r.t. $(\mathbb{R}, <, =)$. Thus ρ is also a locally consistent k-frame sequence w.r.t. $(\mathbb{R}, <, =)$. Now by Corollary 5.4 and Lemma 4.2, ρ admits an $\mathbb{R}$-valuation sequence. Equivalently, there is an edge-respecting $\mathbb{R}$-labelling of G_ρ. Thus a strict cycle in G_ρ would imply a label in G_ρ is less than itself, which is a contradiction. $\square$

6 Satisfiability for CLTL$(\mathbb{Z}, <, =)$

In this section we consider the constraint system $\mathcal{Z} = (\mathbb{Z}, <, =)$. It follows from Lemma 5.3 that $\mathcal{Z}$ does not satisfy the completion property. As a result we cannot appeal to Theorem 4.4 to solve the satisfiability problem for the logic CLTL($\mathcal{Z}$). In fact, we will show that unlike the case for $\mathbb{R}$, the language $L^{\mathcal{Z}}_{fs}(\varphi)$ is *not* ω-regular in general (Corollary 6.5). Nonetheless, we can still solve the satisfiability problem for CLTL($\mathcal{Z}$) automata-theoretically. The idea is to define a Büchi automaton $\mathcal{A}^{\mathcal{Z}}_{\varphi}$ which accepts a *superset* of $L^{\mathcal{Z}}_{fs}(\varphi)$ with the property that all *ultimately periodic* words in it are also in $L^{\mathcal{Z}}_{fs}(\varphi)$. It will then follow that $L(\mathcal{A}^{\mathcal{Z}}_{\varphi})$ is non-empty iff φ is CLTL($\mathcal{Z}$)-satisfiable.

We begin with a characterisation of locally consistent k-frame sequences which admit a $\mathbb{Z}$-valuation sequence, along the lines of [20, Lemma 5.5]. Let ρ be a locally consistent k-frame sequence (which in this section we understand to be w.r.t. $\mathcal{Z}$). For a directed path p in G_ρ, let $slen(p)$ denote the *strict* length of p – i.e. the number of '$<$'-labelled edges in p if this number is finite, and ω otherwise. For any two vertices u, v in G_ρ, define $slen(u,v)$ to be the supremum of $slen(p)$ over directed paths p from u to v, if it exists, and ω otherwise. Whenever there is no directed path from u to v, $slen(u,v)$ takes the value 0 by convention.

Lemma 6.1 *Let ρ be a locally consistent k-frame sequence. Then ρ admits a $\mathbb{Z}$-valuation sequence iff for all $u, v \in G_\rho$, $slen(u,v) < \omega$.*

Proof If ρ admits a $\mathbb{Z}$-valuation sequence σ, let l be the corresponding edge-respecting labelling of G_ρ. Then clearly $slen(u,v) \leq |l(v) - l(u)|$ for all vertices u, v in G_ρ. Thus, there cannot exist vertices u and v with $slen(u,v) = \omega$.

Conversely, suppose G_ρ satisfies the given condition. We assume that k is at least 2 (for $k = 1$, it is clear that ρ always admits a $\mathbb{Z}$-valuation). One can verify that the procedure given below produces an edge-respecting $\mathbb{Z}$-labelling l of G_ρ. This in turn implies that ρ admits a $\mathbb{Z}$-valuation sequence. We assume an ordering $\prec$ on variables, and use it to define an ordering of vertices in G_ρ given by $(x,i) \prec (y,j)$ iff $i < j$, or $i = j$ and $x \prec y$.

(1) Label the vertices in order. Begin by labelling the first, say $(x,0)$, by 0.
(2) In general, if X is the portion of the graph already labelled, and u is the next vertex to be labelled:
  (a) if there is a directed path from u to a vertex in X, set $l(u) = \min\{l(v) - slen(u,v) \mid v \in X,\ \exists \text{ a path from } u \text{ to } v\}$, else,
  (b) set $l(u) = \max\{l(v) + slen(v,u) \mid v \in X,\ \exists \text{ a path from } v \text{ to } u\}$.

Note that with $k \geq 2$ every vertex other than $(x,0)$ has a vertex preceding it in the ordering above, to which it is connected by an edge (from the relation $(\xrightarrow{\leq} \cup \xrightarrow{=} \cup \xrightarrow{\geq})^*$). Hence case (2a) or (2b) is always applicable, and the procedure never gets stuck.

To argue that the procedure does give us a valid $\mathbb{Z}$-labelling for G_ρ, let us assume the contrary. Then there must be a first time where the procedure labels a vertex, say u, with a value which contradicts the strict length of a path from or to this vertex u. Moreover, since there are no strict cycles in G_ρ, this path must be from u to an already labelled vertex, or from an already labelled vertex to u. Let the vertex at the other end of the offending path be v, and let the vertices labelled up to this point be X. Note that $v \in X$. There are two cases to examine:

(1) The offending path p is from u to v, and $\mathit{slen}(p) > l(v) - l(u)$. But in this case, step (2a) of the procedure is applicable, and hence $l(u) \leq l(v) - \mathit{slen}(p)$. Thus this case is not possible.

(2) The offending path p is from v to u with $\mathit{slen}(p) > l(u) - l(v)$. Here again two possibilities arise.

    In the first possibility vertex u was labelled by an application of step (2a) of the procedure. So there must have been a vertex w in X with a path q from u to w, and $l(u) = l(w) - \mathit{slen}(q)$. But since v and w were labelled without any discrepancy, it must be the case that $l(w) - l(v) \geq \mathit{slen}(p) + \mathit{slen}(q)$. Thus $l(u) = l(w) - \mathit{slen}(q) \geq l(v) + \mathit{slen}(p)$. This contradicts our assumption that $\mathit{slen}(p) > l(u) - l(v)$, and hence this possibility is ruled out too.

    The second possibility is that u was labelled by an application of step (2b) of the procedure. But then it must be the case that $l(u) \geq l(v) + \mathit{slen}(p)$. Thus this case is ruled out too.

□

The natural question is now: can we characterize, by means of automata, the locally consistent k-frame sequences that admit a $\mathbb{Z}$-valuation? Consider the condition $(C_{\mathbb{Z}})$ below on a locally consistent k-frame sequence ρ. In what follows, by an infinite *forward* (respectively *backward*) path in G_ρ we will mean a sequence $d : \mathbb{N} \to V \times \mathbb{N}$ satisfying:

(1) for all $i \in \mathbb{N}$, there is an edge from $d(i)$ to $d(i+1)$ (respectively, an edge from $d(i+1)$ to $d(i)$),
(2) for all $i \in \mathbb{N}$, if $d(i)$ is in level j, then $d(i+1)$ is in a level greater than or equal to $j+1$. By the "level" of a vertex (x, j) we mean j.

Such a path d will be called *strict* if there exist infinitely many i for which there is a '$<$'-labelled edge from $d(i)$ to $d(i+1)$ (respectively, from $d(i+1)$ to $d(i)$).

Two vertices are in the same k-frame iff the difference of levels between them is strictly less than k. Here is the condition $(C_{\mathbb{Z}})$.

$(C_{\mathbb{Z}})$: There *do not* exist vertices u and v in the same k-frame in G_ρ satisfying:

(1) there is an infinite forward path d from u,
(2) there is an infinite backward path e from v,
(3) either d or e is strict, and
(4) for each $i, j \in \mathbb{N}$, whenever $d(i)$ and $e(j)$ belong to the same k-frame

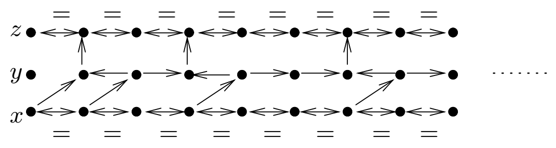


Fig. 2. $\rho^{\mathbb{Z}}_{bad}$ satisfies $(C_{\mathbb{Z}})$ but does not admit a $\mathbb{Z}$-valuation sequence.

there is an edge labelled '$<$' from $d(i)$ to $e(j)$.

It follows from Lemma 6.1 that condition $(C_{\mathbb{Z}})$ is necessary for ρ to admit a $\mathbb{Z}$-valuation sequence: if ρ does not satisfy $(C_{\mathbb{Z}})$ – i.e. there are vertices u and v satisfying the conditions (1)-(4) above – then u and v are such that $slen(u, v) = \omega$, and hence by Lemma 6.1 ρ does not admit a $\mathbb{Z}$-valuation sequence.

But it is *not* sufficient as witnessed by the 2-frame sequence $\rho^{\mathbb{Z}}_{bad}$ in Fig. 2. In the figure we have shown only the relevant edges. Essentially from the vertex (x, i), for i of the form $\frac{j(j+1)}{2}$, we have a strict path of strict length $j + 1$ by going from (x, i) to $(y, i+1)$, and then on to $(y, i+j+1)$. The frame sequence $\rho^{\mathbb{Z}}_{bad}$ can be defined formally as follows. We have $V = \{x, y, z\}$ and:

- $\{x < y < z,\ x < Oy,\ x < Oz,\ z > Ox, z > Oy,\ Ox < y < Oz,\ x = Ox,\ z = Oz\} \subseteq \rho^{\mathbb{Z}}_{bad}(i)$ for every $i \geq 0$;
- $y > Oy \in \rho^{\mathbb{Z}}_{bad}(\frac{i \times (i+1)}{2})$ for every $i \geq 0$;
- $y < Oy \in \rho^{\mathbb{Z}}_{bad}(i)$ for every $i \geq 0$ such that $i \neq \frac{i' \times (i'+1)}{2}$ for some i'.

$\rho^{\mathbb{Z}}_{bad}$ clearly satisfies condition $(C_{\mathbb{Z}})$: there are no strict forward or backward paths from any vertex in G_ρ. However, note that the vertices $u = (x, 0)$ and $v = (z, 0)$ have unbounded strict length – i.e. $slen(u, v) = \omega$, and hence ρ cannot admit a $\mathbb{Z}$-valuation sequence.

However, when ρ is an *ultimately periodic* word condition $(C_{\mathbb{Z}})$ is indeed sufficient. We say an infinite word α is *ultimately periodic* if it is of the form $\tau \cdot \delta^\omega$ for some finite words τ and δ.

Lemma 6.2 *Let ρ be an ultimately periodic, locally consistent, k-frame sequence. Then ρ admits a $\mathbb{Z}$-valuation sequence iff ρ satisfies $(C_{\mathbb{Z}})$.*

Before we proceed with the proof, here are a few observations which can be readily verified. In what follows, $\rho = \tau \cdot \delta^\omega$ is an ultimately periodic, locally consistent k-frame sequence.

(1) We term a level in G_ρ a δ-boundary if it is of the form $|\tau| + i \cdot |\delta|$ for some $i \in \mathbb{N}$. An l-suffix of G_ρ, denoted $G_\rho[l, \infty]$, is the subgraph of G_ρ on the vertices $V \times \{l, l+1, \ldots\}$. Let l and m be δ-boundaries in G_ρ, with $l < m$. Then there is an isomorphism between the suffixes $G_\rho[l, \infty]$ and $G_\rho[m, \infty]$, given by $(x, n) \to (x, n + (m - l))$, where $n \geq l$. Accordingly,

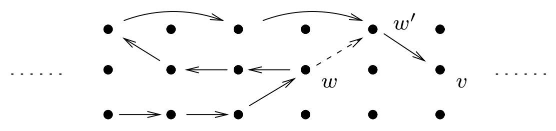


Fig. 3. Short-circuiting a backward loop

we say vertices (x, n) and $(x, n+i)$ are isomorphic if $(x, n+i)$ is the image of (x, n) under one of the isomorphisms above (i.e. n is on or to the right of a δ-boundary, and i is a multiple of $|\delta|$).

(2) A forward path between two isomorphic vertices implies an infinite forward path in G_ρ. Say there is a forward path p from u to u' and u, u' are isomorphic. Then u' is isomorphic to a vertex u'' ahead of it, under the same isomorphism. Thus there is an isomorphic copy of the path p, which goes from u' to u''. This argument can be repeatedly used to concatenate copies of p to obtain an infinite forward path from u.

In a similar manner, a backward path between two isomorphic vertices implies an infinite backward path in G_ρ.

(3) Here is a property of G_ρ which holds even in the absence of periodicity. Let p be a path from u to v in G_ρ, with v at a level ahead of u. Then we can obtain a *forward* path p' from u to v in G_ρ. This is done by "short-circuiting" the portions of the path p that loop backward. With reference to Fig 3, if p does not take a forward edge at a vertex w, then let w' be the first vertex it visits to the right of w (if it exists). Since this edge to w' is from a vertex either to the left or at the same level as w, it must be the case that w and w' are in a common k-frame, and hence there must be an edge between w and w'. This edge must be either an '$=$' or '$<$'-labelled edge from w to w', since otherwise G_ρ would contain a strict cycle. Furthermore, if the backward loop contained a strict edge, the edge from w to w' is strict.

A similar argument holds when p is a path from u to v with u ahead of v. In this case we have a *backward* path p' from v to u.

We return now to the proof of Lemma 6.2.

Proof If ρ admits a $\mathbb{Z}$-valuation sequence, then by Lemma 6.1 ρ must satisfy the condition $(C_\mathbb{Z})$.

Conversely, let $\rho = \tau \cdot \delta^\omega$ be a locally consistent, ultimately periodic k-frame sequence. Suppose ρ does not admit a $\mathbb{Z}$-valuation sequence. Then we will show that ρ fails to meet condition $(C_\mathbb{Z})$.

By Lemma 6.1 there must exist two vertices u and v in G_ρ with $slen(u, v) = \omega$. We will use this property to produce a picture in G_ρ which violates $(C_\mathbb{Z})$. Let l be a level to the right of the first δ-boundary and the vertices u and v. Let m be the level $l + k \cdot |\delta| \cdot |V|$.

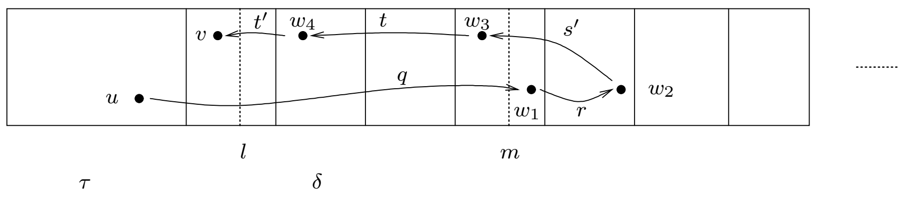


Fig. 4. The path p' for case (a)

Now consider a u-v path p with

$$slen(p) > (m+1) \cdot |V| + |\delta| \cdot |V|.$$

Such a path exists because $slen(u, v) = \omega$. Without any loss of generality we can assume p has no non-strict cycles (and hence no cycles at all), since a non-strict cycle can be eliminated without affecting the strict length of the path. Thus p never re-visits a vertex in G_ρ. It now follows that there must be at least $|\delta| \cdot |V|$ strict edges in p which occur to the *right* of level m.

Now consider the portion p from the time it first crosses to the right of m, till the time it crosses back to the left of m for the last time. Since there are only $|\delta| \cdot |V|$ non-isomorphic vertices to the right of m, it must be the case that this portion of the path visits two isomorphic vertices w_1 and w_2, both to the right of m, such that w_1 is visited first, and w_2 is visited next via a *strict* path. Using the observation on short-circuiting a path, we can now produce a path p' in G_ρ that begins at u, goes to w_1 via a forward path q, then visits w_2 via a strict path r, then goes back to v via a backward path s.

Two cases arise at this point: (a) w_1 is to the left of w_2, and (b) w_1 is to the right of w_2.

We deal with case (a) first. In this case r must be a strict forward path from w_1 to w_2. Notice also that the backward path s spans across the levels l to m. Since an edge in s can span at most k levels, by the choice of m, s must visit two isomorphic vertices w_3 and w_4 which lie between the levels l and m. Without loss of generality we assume w_3 is to the right of w_4. Let this backward path from w_4 to w_3 be t. Let us write the path s as $s'tt'$. This is shown in Fig. 4.

Since w_1 and w_2 are isomorphic, we have a strict forward path d from u (given by q followed by r, followed by isomorphic copies of r ad infinitum). Similarly, since w_4 and w_3 are isomorphic, we have an infinite backward (not necessarily strict) path e from v.

Further, we can assume that u and v lie in the same k-frame, since if this were not the case, with say v ahead of u, we can choose u to be any vertex along the path q that lies in the same k-frame as v.

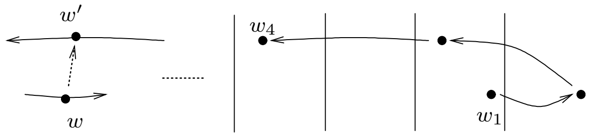


Fig. 5. Strict edge from w to w'

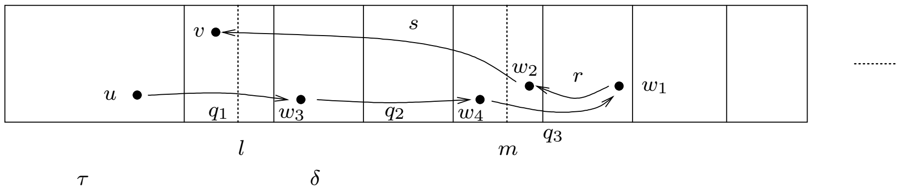


Fig. 6. The path p' for case (b)

It remains now to argue that for any two vertices w on d and w' on e, which are in the same k-frame, there is a strict edge from w to w'. Now note that the number of levels spanned by the paths r and t are a multiple of $|\delta|$; say $k_1 \cdot |\delta|$ and $k_2 \cdot |\delta|$ respectively (by isomorphism of (w_1, w_2) and (w_3, w_4)). Let $w_1 = (x, n_1)$ and $w_4 = (y, n_4)$. Then the vertices $(x, n_1 + ik_1k_2|\delta|)$ and $(y, n_4 + ik_1k_2|\delta|)$, for any i, are images of w_1 and w_4 under the same isomorphism. Hence there is a strict path isomorphic to $rs'tt'$ between these vertices as well. Further, these vertices also lie on the paths d and e respectively. Now given w and w' on d and e respectively, consider the next occurrence of the two isomorphic copies of w_1 and w_4 described above, to the right of both w and w' (see Fig. 5). Then there is a strict path between w and w' obtained by going right to the copy of w_1, then taking the strict copy of $rs'tt'$ to the copy of w_4, and then back to w'. Hence there must be a strict edge from w to w' (or else we would have a strict cycle in G_ρ, see Corollary 5.6).

This completes the proof that in this case there is a picture in G_ρ which contradicts $(C_\mathcal{Z})$.

Coming now to case (b), where we assume w_1 is to the *right* of w_2. Then the path r is a strict backward path from w_2 to w_1. Then the path p' from u to v appears as shown in figure Fig. 6, with q_2 being a forward path between two isomorphic vertices. Once again this gives us an infinite forward path from u given by q_1 followed by q_2 ad infinitum, and a *strict* backward path from v given by s followed by r ad infinitum. Using an argument similar to case (a) above, we once again get a picture in G_ρ which contradicts $(C_\mathcal{Z})$.

With this we conclude the proof of Lemma 6.2. $\square$

Now let φ be the given CLTL($\mathcal{Z}$) formula. Let k be the O-length of φ. Let us define $\mathcal{A}^{\mathcal{Z}}_{\varphi} = \mathcal{A}^{\mathrm{LTL}}_{\varphi} \cap \mathcal{A}^{\mathcal{Z},k}_{lc} \cap \mathcal{A}^{\mathcal{Z}}_{k}$, where $\mathcal{A}^{\mathrm{LTL}}_{\varphi}$ and $\mathcal{A}^{\mathcal{Z},k}_{lc}$ are as in Sec. 4, and $\mathcal{A}^{\mathcal{Z}}_{k}$ – described below – accepts k-frame sequences which satisfy $(C_\mathcal{Z})$.

Let $\mathcal{B}$ be a Büchi automaton over the alphabet of k-frames, which simply checks the negation of condition $(C_{\mathcal{Z}})$. Thus $\mathcal{B}$ non-deterministically guesses the vertices u and v, and then verifies the conditions (1)–(4). We non-deterministically choose in the beginning, whether to signal (via the Büchi condition) each time the d or e path sees an edge labelled '$<$'. Here is a more precise definition of the automaton $\mathcal{B}$:

Define $\mathcal{B} = (Q, S_0, \longrightarrow, F)$, over the alphabet of k-frames, where

- $S_0 = \{q_0\}$;
- $Q = \{q_0\} \cup (V \times \{0, \ldots, (k-1)\} \times V \times \{0, \ldots, (k-1)\} \times \{d, e\} \times \{0, 1\})$;
- $\longrightarrow$ is given by:
  - · $q_0 \xrightarrow{v} q_0$
  - · (guess (x, i) as u, and (y, j) as v, as well as which of d or e is strict) $q_0 \xrightarrow{v} (x, i, y, j, d, 0)$ and $q_0 \xrightarrow{v} (x, i, y, j, e, 0)$ for all x, i, y, j.
  - · (wait till x or y is at the edge of the frame) $(x, i, y, j, f, b) \xrightarrow{v} (x, i-1, y, j-1, f, b)$ for $f \in \{d, e\}$ and $b \in \{0, 1\}$, provided $i, j \geq 2$.
  - · (guess a forward edge from $(x, 1)$) $(x, 1, y, j, d, b) \xrightarrow{v} (z, i, y, j-1, d, b')$ provided
    $j > 1$,
    $(x, 0) \xrightarrow{<} (z, i) \in v$ and $b' = 1$, or $(x, 0) \xrightarrow{=} (z, i) \in v$ and $b' = 0$,
    $(z, i) \xrightarrow{<} (y, j-1) \in v$.
  - · similarly for $(y, 1)$, and for $(x, 1)$ and $(y, 1)$ simultaneously.
  - · similar clause for e.
- F comprises all states of the form $(x, i, y, j, f, 1)$.

The automaton $\mathcal{A}_k^{\mathcal{Z}}$ is now just the complement of $\mathcal{B}$.

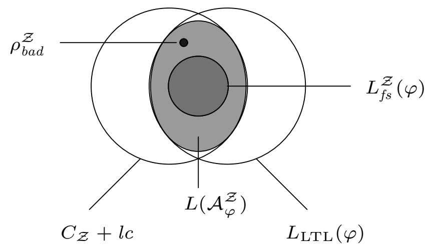
Lemma 6.3 *A* CLTL($\mathcal{Z}$) *formula* φ *is satisfiable iff* $L(\mathcal{A}_\varphi^{\mathcal{Z}})$ *is non-empty.*

Proof Suppose φ is satisfiable. Let σ be a $\mathbb{Z}$-valuation sequence such that $\sigma \models \varphi$. Let $\rho = k\text{-}fs_{\mathcal{Z}}(\sigma)$, where k is the O-length of φ. By Lemma 3.1, $\rho \in L(\mathcal{A}_\varphi^{\mathrm{LTL}})$. We know that ρ is a locally consistent k-frame sequence and hence $\rho \in L(\mathcal{A}_{lc}^{\mathcal{Z},k})$. Further, by Lemma 6.1, G_ρ satisfies $(C_{\mathcal{Z}})$ and hence $\rho \in L(\mathcal{A}_\varphi^{\mathcal{Z}})$. Thus $\rho \in L(\mathcal{A}_\varphi^{\mathrm{LTL}}) \cap L(\mathcal{A}_{lc}^{\mathcal{Z},k}) \cap L(\mathcal{A}_k^{\mathcal{Z}})$. Hence $\rho \in L(\mathcal{A}_\varphi^{\mathcal{Z}})$.

Conversely, suppose $\mathcal{A}_\varphi^{\mathcal{Z}}$ accepts a word ρ. Then it must accept an ultimately periodic word ρ' (by nature of the acceptance condition). Since $\rho' \in L(\mathcal{A}_\varphi^{\mathcal{Z}})$, we know that ρ' is such that $\rho' \models_{\mathrm{LTL}} \varphi$, ρ' is locally consistent, and satisfies $(C_{\mathcal{Z}})$. By Lemma 6.2, ρ' admits a $\mathbb{Z}$-valuation sequence σ. Further, since $\rho' \models_{\mathrm{LTL}} \varphi$, by Lemma 3.1, $\sigma \models \varphi$. Thus φ is CLTL($\mathcal{Z}$) satisfiable. $\square$

This lets us conclude that

Theorem 6.4 *The satisfiability problem for* CLTL($\mathcal{Z}$) *is decidable.*



$\rho^{\mathcal{Z}}_{bad}$
$L^{\mathcal{Z}}_{fs}(\varphi)$
$L(\mathcal{A}^{\mathcal{Z}}_{\varphi})$
$C_{\mathcal{Z}} + lc$
$L_{\mathrm{LTL}}(\varphi)$

Fig. 7. The languages $L^{\mathcal{Z}}_{fs}(\varphi)$ and $L(\mathcal{A}^{\mathcal{Z}}_{\varphi})$.

Proof Given $\varphi \in \mathrm{CLTL}(\mathcal{Z})$ we can effectively construct the automaton $\mathcal{A}^{\mathcal{Z}}_{\varphi}$ and check it for emptiness. $\Box$

The decision procedure can be shown to run in PSPACE. Before that, as a consequence of Lemma 6.2 and the existence of the automaton $\mathcal{A}^{\mathcal{Z}}_{\varphi}$, we observe that:

Corollary 6.5 *There are formulas φ in* CLTL($\mathcal{Z}$) *for which $L^{\mathcal{Z}}_{fs}(\varphi)$ is* not *ω-regular.*

Proof Consider the formula $\varphi = \Diamond((x < Oy) \wedge (y < z))$. We will show that $L^{\mathcal{Z}}_{fs}(\varphi)$ is not ω-regular.

Let us suppose that $L^{\mathcal{Z}}_{fs}(\varphi)$ was ω-regular. Then, since ω-regular languages are closed under boolean operations, $L = ((2^{atc(2,\mathcal{Z})})^{\omega} - L^{\mathcal{Z}}_{fs}(\varphi)) \cap L(\mathcal{A}^{\mathcal{Z}}_{\varphi})$ is also ω-regular. This is the lightly shaded region in Fig. 7. Note that L is disjoint from $L^{\mathcal{Z}}_{fs}(\varphi)$.

Now consider the frame sequence $\rho^{\mathcal{Z}}_{bad}$ of Fig. 2. $\rho^{\mathcal{Z}}_{bad}$ satisfies $(C_{\mathcal{Z}})$ and is a locally consistent 2-frame sequence. Further it satisfies φ as an LTL formula (i.e. $\rho^{\mathcal{Z}}_{bad} \models_{\mathrm{LTL}} \varphi$). Hence it belongs to $L(\mathcal{A}^{\mathcal{Z}}_{\varphi})$. However, as observed earlier, $\rho^{\mathcal{Z}}_{bad}$ does not admit a $\mathbb{Z}$-valuation sequence, and hence $\rho^{\mathcal{Z}}_{bad} \notin L^{\mathcal{Z}}_{fs}(\varphi)$. Hence $\rho^{\mathcal{Z}}_{bad} \in L$.

But since L is ω-regular, there must exist an ultimately periodic word ρ' in L. $\rho' \in L(\mathcal{A}^{\mathcal{Z}}_{\varphi})$ and hence is locally consistent, satisfies $(C_{\mathcal{Z}})$, and satisfies φ as an LTL formula. By Lemma 6.2, ρ' admits a $\mathbb{Z}$-valuation sequence. Since we also have that $\rho' \in L(\mathcal{A}^{\mathrm{LTL}}_{\varphi})$, by Lemma 3.1 we must have $\rho \in L^{\mathcal{Z}}_{fs}(\varphi)$. But this contradicts the fact that L and $L^{\mathcal{Z}}_{fs}(\varphi)$ are disjoint. Hence $L^{\mathcal{Z}}_{fs}(\varphi)$ could not have been ω-regular. $\Box$

Complexity of the decision procedure. We now want to argue that the satisfiability problem for CLTL($\mathcal{Z}$) can be solved in PSPACE. Given a CLTL($\mathcal{Z}$) formula φ of O-length k, we show that the decision procedure above runs in space polynomial in $n = |\varphi|$.

$$\begin{tikzcd}
\mathcal{B} \arrow[d, dash] & & \\
\mathcal{C} \arrow[d, dash] & & \\
\mathcal{A}_k^{\mathcal{Z}} \arrow[dr, dash] & \mathcal{A}_\varphi^{\mathrm{LTL}} \arrow[d, dash] & \mathcal{A}_{lc}^{\mathcal{Z},k} \arrow[dl, dash] \\
& \mathcal{A}_\varphi^{\mathcal{Z}} &
\end{tikzcd}$$

Fig. 8. $\mathcal{A}_\varphi^{\mathcal{Z}}$ and component automata

Recall that we need to check if the language accepted by the automaton $\mathcal{A}_\varphi^{\mathcal{Z}} = \mathcal{A}_\varphi^{\mathrm{LTL}} \cap \mathcal{A}_{lc}^{\mathcal{Z},k} \cap \mathcal{A}_k^{\mathcal{Z}}$, is empty.

We briefly describe the construction of the automaton $\mathcal{A}_k^{\mathcal{Z}}$. It is obtained from the automaton $\mathcal{B}$ above by Safra's construction [46] to complement a Büchi automaton. Note that the automaton $\mathcal{B}$ has a number of states which is polynomial in n, say $p(n)$. From this Safra's method constructs a deterministic Streett automaton $\mathcal{C}$ which accepts the complement of the language accepted by $\mathcal{B}$, and has $2^{O(p(n)\log n)}$ states. This automaton can then be converted to an equivalent Büchi automaton $\mathcal{A}_k^{\mathcal{Z}}$ with the same order of states.

Though the automaton $\mathcal{A}_\varphi^{\mathcal{Z}}$ is exponential in n (note that both component automata $\mathcal{A}_k^{\mathcal{Z}}$ and $\mathcal{A}_\varphi^{\mathrm{LTL}}$ are exponential in n), we can still check its nonemptiness non-deterministically in PSPACE in n, as argued below. Using Savitch's theorem, it then follows that its nonemptiness can be checked deterministically in PSPACE in n.

To argue that $\mathcal{A}_\varphi^{\mathcal{Z}}$ can be checked for nonemptiness non-deterministically in PSPACE, we note that $\mathcal{A}_\varphi^{\mathcal{Z}}$ has an implicitly defined transition relation that can be checked in PSPACE in n. Fig. 8 depicts the fact that the transition relation of $\mathcal{A}_\varphi^{\mathcal{Z}}$ is implicitly defined in terms of the transition relations of its component automata. Each of the component automata have implicitly defined transition relations, that can be computed in PSPACE. For $\mathcal{A}_\varphi^{\mathrm{LTL}}$ the argument is standard and can be found in [47]. For $\mathcal{A}_k^{\mathcal{Z}}$ one can verify in [46] that the construction of the automata $\mathcal{C}$ and $\mathcal{A}_k^{\mathcal{Z}}$ are indeed such that their states can be described in space polynomial in n and their transition relation can be checked in space polynomial in n.

Thus we can conclude that checking nonemptiness of the automaton $\mathcal{A}_\varphi^{\mathcal{Z}}$ can be done in PSPACE in n. PSPACE-hardness follows from that of classical LTL as in the proof of Theorem 4.5. Hence we have:

Theorem 6.6 *The satisfiability problem for* CLTL($\mathcal{Z}$) *is* PSPACE*-complete.*

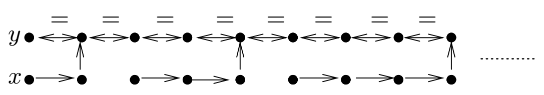


Fig. 9. $\rho_{bad}^{\mathcal{N}}$ satisfies $(C_{\mathcal{N}})$ but does not admit an $\mathbb{N}$-valuation sequence.

7 Satisfiability for $\mathrm{CLTL}(\mathbb{N}, <, =)$

In this section we outline a decision procedure for the satisfiability problem for CLTL over the constraint system $\mathcal{N} = (\mathbb{N}, <, =)$. The decidability of this logic actually follows from the fact that $\mathrm{CLTL}(\mathcal{Z})$ extended with constants is decidable (cf. Sec. 9). However for reasons of independent interest, we sketch a decision procedure for this logic, which in fact is very much along the lines of the one for $\mathrm{CLTL}(\mathcal{Z})$ in the previous section. It is hoped that this will emphasize the fact that a simple-minded approach to rule out infinite backward chains will not work, and further that the languages corresponding to these formulas could in fact be non-ω-regular.

We begin with a notion similar to that of *slen* in the previous section. For a vertex u in G_ρ, we define $sdlen(u)$ to be the supremum of $slen(p)$ over directed paths p from some vertex v to u in G_ρ.

Lemma 7.1 *Let ρ be a locally consistent k-frame sequence. Then ρ admits an $\mathbb{N}$-valuation sequence iff for all $u \in G_\rho$, $sdlen(u) < \omega$.*

Proof If ρ admits an $\mathbb{N}$-valuation σ, then clearly $sdlen(u) \leq l(u)$, where l is the $\mathbb{N}$-labelling of G_ρ corresponding to σ. Thus, there cannot exist a vertex u with $sdlen(u) = \omega$.

Conversely, if G_ρ satisfies the given condition, then one can verify that the $\mathbb{N}$ labelling l given by $l(u) = sdlen(u)$ is an edge-respecting labelling of G_ρ. Hence ρ admits an $\mathbb{N}$-valuation. $\square$

Let $(C_{\mathcal{N}})$ denote the following condition on a locally consistent k-frame sequence ρ: G_ρ satisfies $(C_{\mathcal{Z}})$ *and* it does not contain a strict backward path.

Condition $(C_{\mathcal{N}})$ is once again necessary but *not* sufficient to ensure that a locally consistent k-frame sequence admits an $\mathbb{N}$-valuation sequence. One can see that it is not sufficient by considering the 2-frame sequence $\rho_{bad}^{\mathcal{N}}$ of Fig. 9. It clearly satisfies condition $(C_{\mathcal{N}})$, but does not admit an $\mathbb{N}$-valuation sequence because the vertex $u = (y, 0)$ is such that $sdlen(u) = \omega$.

However, we can assert that:

Lemma 7.2 *Let ρ be an ultimately periodic, locally consistent, k-frame sequence. Then ρ admits an $\mathbb{N}$-valuation sequence iff ρ satisfies $(C_{\mathcal{N}})$.*

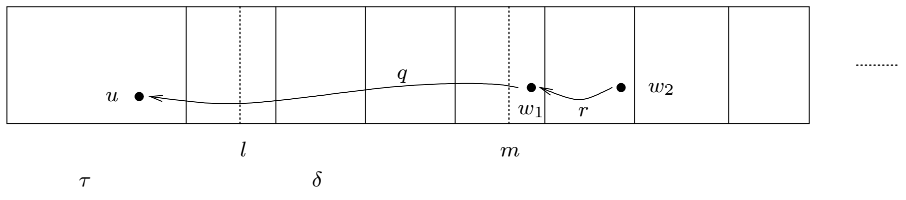


Fig. 10. The case (a)

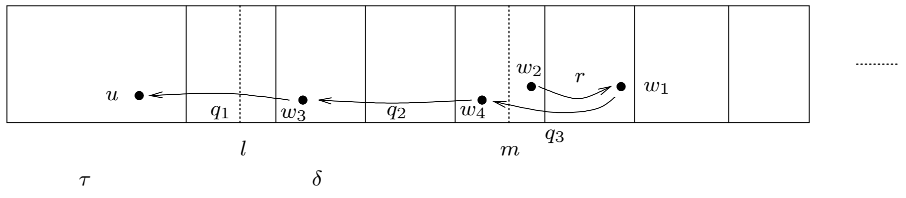


Fig. 11. The case (b)

Proof Suppose ρ does not admit an $\mathbb{N}$-valuation. Then by Lemma 7.1 there must exist a vertex u with unbounded strict descending length. We now choose the levels l and m, and a backward path p from u of suitable strict length, as in the $\mathcal{Z}$ case. Once again, we must visit two isomorphic vertices w_1 and w_2 (in that order, as we go along the path in the *backward* direction), with both w_1 and w_2 to the right of level m, and with a strict path between them. Again, two cases arise: either w_1 is to the left of w_2, or w_2 is the left of w_1. If w_1 is to the left of w_2, then we have the situation in Fig. 10, and we have an infinite backward path from u given by q followed by r ad infinitum. This contradicts condition $(C_{\mathcal{N}})$.

If w_2 is visited before w_1, then the situation in Fig. 11 arises. We now have a strict infinite forward path d from w_2 given by r followed by r ad infinitum. We also have an infinite backward path e from w_3 given by q_2 repeated ad infinitum. As in the case of $\mathcal{Z}$ we can argue that for any two vertices w and w' on d and e respectively, there is a strict edge from w to w'. This picture violates $(C_{\mathcal{Z}})$ and hence $(C_{\mathcal{N}})$. $\square$

We can now define an automaton $\mathcal{A}_{\varphi}^{\mathcal{N}}$ as in the $\mathbb{Z}$ case, so that the CLTL($\mathcal{N}$) formula φ is satisfiable iff $L(\mathcal{A}_{\varphi}^{\mathcal{N}})$ is non-empty. We can define $\mathcal{A}_{\varphi}^{\mathcal{N}}$ to be the intersection of $\mathcal{A}_{\varphi}^{\mathcal{Z}}$ and an automaton that rejects k-frame sequences which contain a vertex with a strict backward path from it.

In a similar manner to the $\mathcal{Z}$ case, we can thus conclude that:

Theorem 7.3 *The satisfiability problem for* CLTL($\mathcal{N}$) *is* PSPACE*-complete.*

Finally, the frame sequence ρ_{bad}^{N} of Fig. 9 allows us to argue that there is a CLTL($\mathcal{N}$) formula φ (for example $x < Ox$) for which $L_{fs}^{\mathcal{N}}(\varphi)$ is not ω-regular. Once again the argument is similar to the $\mathcal{Z}$ case.

8 Model-Checking

A natural model-checking problem for CLTL($\mathcal{D}$) one may study would be to consider a Kripke structure $\mathcal{M}$ whose states are labelled by D-valuations, and a CLTL($\mathcal{D}$) formula φ; and to ask if every valuation sequence generated by $\mathcal{M}$ satisfies φ. However this problem easily reduces to the classical LTL model-checking problem, since we can transform $\mathcal{M}$ into a Kripke structure $\mathcal{M}_k$ labelled by k-frames (where k is the O-length of φ) corresponding to the k-frame induced by k-length valuation sequences in $\mathcal{M}$. Whether $\mathcal{M}$ satisfies φ is now equivalent to checking whether $\mathcal{M}_k$ satisfies φ as a classical LTL formula.

Hence, in this section we focus on a more interesting model-checking problem which has been proposed earlier in the literature in the context of counter automata [13] (see also [48,49] and [50, Chap. 6]). As in the case of classical LTL, this model-checking problem will be shown to be equivalent to the satisfiability problem for the logic.

Let $\mathcal{D}$ be a constraint system. A *$\mathcal{D}$-automaton* $\mathcal{A}$ is a Büchi automaton $\mathcal{A}$ over an alphabet comprising CLTL($\mathcal{D}$) formulae. Thus transitions in $\mathcal{A}$ are of the form $q \xrightarrow{\varphi} q'$, see similar structures in [51]. Fig. 12 shows a $\mathcal{N}$-automaton with variables $\{x, y, z\}$.

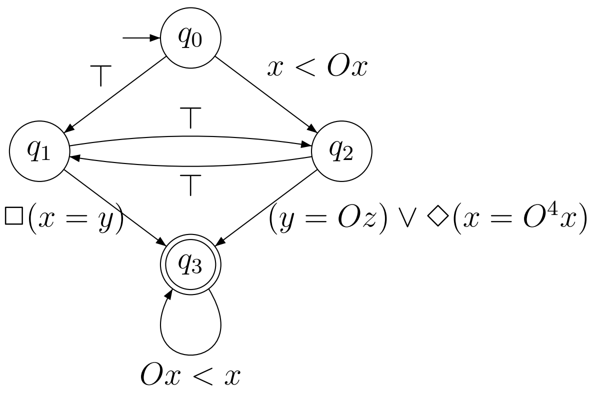


Fig. 12. An example of a $\mathcal{D}$-automaton

Viewed as a Büchi automaton over CLTL($\mathcal{D}$) formulas, a $\mathcal{D}$-automaton $\mathcal{A}$ accepts a language $L(\mathcal{A})$ of ω-words over CLTL($\mathcal{D}$) formulas. Elements of $L(\mathcal{A})$ can be understood as symbolic models. However, with each ω-word $\alpha = \varphi_0\varphi_1\cdots$ in $L(\mathcal{A})$, we can associate a set of D-valuation sequences σ which satisfy $\sigma[i,\infty] \models \varphi_i$ for each i. Such sequences can be viewed as concrete models. Let $L_{vs}(\mathcal{A})$ denote this set of D-valuation sequences: thus

$$L_{vs}(\mathcal{A}) = \{\sigma \mid \exists\alpha \in L(\mathcal{A}), \sigma[i,\infty] \models \alpha(i) \text{ for each } i\}.$$

The $\mathcal{N}$-automaton $\mathcal{A}$ in Fig. 12 is such that $L_{vs}(\mathcal{A}) = \emptyset$. However, if we consider it as an $\mathcal{Z}$-automaton, $L_{vs}(\mathcal{A})$ contains the $\mathbb{Z}$-valuation sequence

$$\langle 0,0,0\rangle, \langle 0,0,0\rangle, \langle -1,-1,0\rangle, \langle -2,-2,0\rangle, \langle -3,-3,0\rangle, \ldots$$

The model-checking problem for CLTL($\mathcal{D}$) is defined as follows: Given a $\mathcal{D}$-automaton $\mathcal{A}$ and a CLTL($\mathcal{D}$) formula φ, is there a $\sigma \in L_{vs}(\mathcal{A})$ that $\sigma \models \varphi$? (in symbols $\mathcal{A} \models_{\exists} \varphi$?)

Note that for convenience we have defined the existential version of the problem. The universal version of the problem can be phrased in terms of the version above. We also note that the nonemptiness problem for $\mathcal{D}$-automata is a restriction of the model-checking problem for CLTL($\mathcal{D}$), by taking $\varphi = \top$. The problems are in fact inter-reducible to each other.

We now show that the satisfiability and model-checking problems for CLTL($\mathcal{D}$) are inter-reducible. This is analogous to the situation for classical LTL.

Theorem 8.1 *Let $\mathcal{D}$ be a non-trivial constraint system. Then the model-checking and satisfiability problems for* CLTL($\mathcal{D}$) *are inter-reducible with respect to logspace transformations.*

Proof Let us reduce the satisfiability problem to the model-checking problem for CLTL($\mathcal{D}$). We only need to observe that φ is CLTL($\mathcal{D}$) satisfiable iff $\mathcal{A}_\top \models_{\exists} \varphi$ where $\mathcal{A}_\top$ is a one-state $\mathcal{D}$-automaton such that $q_0 \xrightarrow{\top} q_0$ and $Q_0 = F = \{q_0\}$.

To reduce the model-checking problem to the satisfiability problem, we use the idea behind the reduction from the model-checking problem to the satisfiability problem for classical LTL from [44]. Let $\mathcal{A} = (Q, Q_0, F, \rightarrow)$ be a $\mathcal{D}$-automaton and φ be a CLTL($\mathcal{D}$) formula. By non-triviality of $\mathcal{D}$, there is R of arity a such that $R \neq \emptyset$ or $R \neq D^a$. For every state $q \in Q$, we consider an exclusive set of new variables $x_1^q, \ldots, x_a^q$ from Var, distinct from those in V. In that way, for each $q \in Q$, $R(x_1^q, \ldots, x_a^q)$ behaves as an independent atomic formula that can be locally set to either true or false. Now, let us encode $\mathcal{A}$ by a CLTL($\mathcal{D}$) formula. For every $q \in Q$, let $next_q$ be the formula that encodes the successors of q:

$$next_q \stackrel{\text{def}}{=} \bigvee_{q \xrightarrow{\psi} q' \in \delta} (\psi \wedge R(Ox_1^{q'}, \ldots, Ox_a^{q'})).$$

Let uni be the formula stating the unicity of the current state:

$$uni \stackrel{\text{def}}{=} \bigvee_{q \in Q} (R(x_1^q, \ldots, x_a^q) \wedge \bigwedge_{q' \in Q \setminus \{q\}} \neg R(x_1^{q'}, \ldots, x_a^{q'})).$$

Let $\varphi_{\mathcal{A}}$ be the formula encoding the accepting runs of $\mathcal{A}$:

$$\begin{aligned}\varphi_{\mathcal{A}} \stackrel{\text{def}}{=} & \bigvee_{q \in Q_0} R(x_1^q, \ldots, x_a^q) \\ & \wedge \Box uni \\ & \wedge \Box(\bigwedge_{q \in Q} R(x_1^q, \ldots, x_a^q) \Rightarrow next_q) \\ & \wedge \Box\Diamond(\bigvee_{q \in F} R(x_1^q, \ldots, x_a^q)).\end{aligned}$$

One can show that $\mathcal{A} \models_{\exists} \varphi$ iff $\varphi \wedge \varphi_{\mathcal{A}}$ is CLTL($\mathcal{D}$) satisfiable. $\Box$

Observe that in the second part of the above proof, we use the fact that we have a countably infinite supply of variables in Var.

As a consequence of Theorem 8.1, the PSPACE-completeness results for the satisfiability problem carry over for the model-checking problem too. In particular, from Theorem 4.5 we get that the model-checking problem for CLTL($\mathcal{D}$) when $\mathcal{D}$ satisfies completion, non-triviality, and allows frame-checking in PSPACE (and hence for all the constraint systems in Corollary 5.4), is PSPACE-complete. Also, from Theorems 6.6 and 7.3 we have that the model-checking problem for $\mathcal{Z}$ and $\mathcal{N}$ are also PSPACE-complete.

9 Decidable Extensions

In this section we outline some extensions of the logic CLTL($\mathcal{D}$) for which the satisfiability and model checking problems remain decidable.

9.1 Monadic second order logic

We define a Constraint Monadic Second Order Logic, parameterized by the constraint system $\mathcal{D}$, and denoted CMSO($\mathcal{D}$). Here the underlying logic is generalized from LTL to MSO [15].

We assume a supply of individual variables $\mathsf{x}, \mathsf{y}, \ldots$, and set variables $\mathsf{X}, \mathsf{Y}, \ldots$. These variables will range over positions (respectively sets of positions) of a given sequence. The syntax of CMSO($\mathcal{D}$) is given by:

$$\varphi ::= Q_c(\mathsf{x}) \mid \mathsf{x} \in \mathsf{X} \mid \mathsf{x} \prec \mathsf{y} \mid \neg\varphi \mid (\varphi \vee \varphi) \mid \exists \mathsf{x}\varphi \mid \exists \mathsf{X}\varphi.$$

Here c is an atomic $\mathcal{D}$-term constraint, and Q_c is a unary predicate with the semantics below.

A structure for the logic will be a pair $(\sigma, \mathbb{J})$ where σ is a D-valuation sequence, and $\mathbb{J}$ is an assignment of individual variables to a position of σ (i.e. an element of $\mathbb{N}$), and set variables to a set of positions of σ. The predicate '$\prec$' is interpreted as the usual ordering on $\mathbb{N}$. The satisfaction relation $\sigma, \mathbb{J} \models \varphi$ for atomic formulas φ is given below.

$$\sigma, \mathbb{J} \models Q_c(\mathsf{x}) \text{ iff } \sigma[\mathbb{J}(\mathsf{x}), \infty] \models_{\mathcal{D}} \mathsf{c}$$

$$\sigma, \mathbb{J} \models \mathsf{x} \in \mathsf{X} \text{ iff } \mathbb{J}(\mathsf{x}) \in \mathbb{J}(\mathsf{X})$$

$$\sigma, \mathbb{J} \models \mathsf{x} \prec \mathsf{y} \text{ iff } \mathbb{J}(\mathsf{x}) < \mathbb{J}(\mathsf{y})$$

The operators $\neg$, $\vee$, and the existential quantifiers $\exists\mathsf{x}$ and $\exists\mathsf{X}$ are interpreted in the usual manner. In particular the quantifier $\exists\mathsf{X}$ is interpreted as follows. Let $i \in \mathbb{N}$. We will use the notation $\mathbb{J}[i/\mathsf{x}]$ to denote the assignment which maps x to i and agrees with $\mathbb{J}$ on all other individual and set variables. Similarly, for a subset S of $\mathbb{N}$, the notation $\mathbb{J}[S/\mathsf{X}]$ will denote the interpretation which sends X to S, and agrees with $\mathbb{J}$ on all other variables. Then:

$$\sigma, \mathbb{J} \models \exists\mathsf{X}\varphi \text{ iff there exists } S \subseteq \mathbb{N} \text{ such that } \sigma, \mathbb{J}[S/\mathsf{X}] \models \varphi.$$

As an illustration, the CMSO($\mathcal{N}$) formula $\forall\mathsf{x}\, Q_{x<Oy}\,(\mathsf{x})$ rephrases the CLTL($\mathcal{N}$) formula $\Box(x < Oy)$. By analogy to CLTL($\mathcal{D}$), the O-length of some CMSO($\mathcal{D}$) φ is the maximal O-length of $\mathcal{D}$-terms occurring in unary predicates of φ.

Recall that classical monadic second order logic over a set of propositions P is defined similar to CMSO($\mathcal{D}$) above, except that instead of Q_c we have predicates of the form Q_p for each $p \in P$. A model for the logic is an ω-word α over 2^P, and the formula $Q_p(\mathsf{x})$ is interpreted as: $\alpha, \mathbb{J} \models_{\text{MSO}} Q_p(\mathsf{x})$ iff $p \in \alpha(\mathbb{J}(\mathsf{x}))$. Just as in Lemma 3.1 it is easy to show the following result.

Lemma 9.1 *Let φ be an* CMSO($\mathcal{D}$) *formula of O-length k. Let σ be a D-valuation sequence, and let $\rho = k\text{-}fs_{\mathcal{D}}(\sigma)$. Then $\sigma \models \varphi$ iff $\rho \models_{\text{MSO}} \varphi$.* $\Box$

Using Büchi's well-known result that for every classical MSO formula we can construct a Büchi automaton $\mathcal{A}_{\varphi}^{\text{MSO}}$ which accepts precisely the models of φ, we can now follow the same route as in the previous sections to conclude that:

Theorem 9.2 *The satisfiability (and model-checking) problem for* CMSO($\mathcal{D}$) *is decidable for $\mathcal{D} = \mathcal{N}$, $\mathcal{Z}$, and for any $\mathcal{D}$ which satisfies completion, non-triviality, and has a decidable frame checking problem.* $\Box$

We can extend the language of constraints by allowing *constants* to be used, leading to term constraints of the form $(Ox < 3)$. The model checking problem for this extension can be handled as in Sec. 8, so we focus here on the satisfiability problem. We restrict ourselves to domains of the form $(D, <, =)$ introduced in Sec. 5, and denote the version of the logic extended with constants by $\mathrm{CLTL}^{con}(D, <, =)$. For reasons of effectiveness, we assume the constants are from $D \cap \mathbb{Q}$.

Consider first the case of domains which are dense and open (or equivalently those which satisfy the completion property). Let φ be a formula with constants $K_1, \ldots, K_n$. Without loss of generality we assume that $K_1 \sim_1 K_2 \sim_2 \ldots \sim_{n-1} K_n$ with $\sim_i \in \{<, =\}$. Consider the $\mathrm{CLTL}(\mathbb{R}, <, =)$ formula φ':

$$\varphi[K_1 \leftarrow y_1, \ldots, K_n \leftarrow y_n] \wedge \bigwedge_{i=1}^{n-1} (y_i \sim_i y_{i+1}) \wedge \Box(\bigwedge_{i=1}^{n} y_i = Oy_i)$$

where $y_1, \ldots, y_n$ are new variables not occurring in φ and $K_i \leftarrow y_i$ denotes the operation of replacing every occurrence of K_i by y_i. φ' is now satisfiable iff φ is. To see this, note that if φ is satisfiable, the same valuation extended by labelling y_i's by K_i's satisfies φ'. Conversely, given a D valuation sequence σ which satisfies φ', we can re-label the variables $y_1, \ldots, y_n$ with $K_1, \ldots, K_n$, and repeatedly use the completion property to label the remaining variables so that we get an edge-respecting labelling of the underlying frame sequence. This gives us a valuation sequence which satisfies φ.

Corollary 9.3 *The satisfiability problem for* $\mathrm{CLTL}^{con}(D, <, =)$ *when D is dense and open, is solvable in* PSPACE.

We now consider $\mathrm{CLTL}^{con}(\mathcal{Z})$. Once again we can reduce the satisfiability for $\mathrm{CLTL}^{con}(\mathcal{Z})$ to that of $\mathrm{CLTL}(\mathcal{Z})$. Let φ be a $\mathrm{CLTL}^{con}(\mathcal{Z})$ formula, and let m and M be respectively the minimum and maximum constants used in φ. Let $x_m, x_{m+1}, \ldots, x_M$ be new variables that do not occur in $voc(\varphi)$. Consider the $\mathrm{CLTL}(\mathcal{Z})$ formula φ' which is the conjunction of:

(1) $\varphi[m \leftarrow x_m, \ldots, M \leftarrow x_M]$
(2) $\Box \bigwedge_{i=m}^{M} (x_i = Ox_i)$
(3) $(x_m < \ldots < x_M)$
(4) $\bigwedge_{y \in voc(\varphi)} \Box((y < x_m) \vee (y > x_M) \vee (y = x_m) \vee \ldots \vee (y = x_M))$.

The claim is now that φ is satisfiable iff φ' is. Given a model for φ we can extend it to a model for φ' by assigning the value K to x_K for each K between m and M. Conversely, consider a model σ for φ'. Then σ may not be a model for φ for two reasons: first, the values assigned to $x_m, \ldots, x_M$ may not be contiguous, and secondly, they may not begin at m. The first problem can be taken care of by observing that the values assigned to $x_m, \ldots, x_M$ in σ can be "compressed" without contradicting the formula. Thus, if $d_m, \ldots, d_M$ were the values assigned to $x_m, \ldots, x_M$ in σ, then we relabel all vertices which are labelled with these values, by the values $d_m, d_m + 1, \ldots, d_m + (M - m)$ respectively. Clause 4 of the formula ensures that this is still a model of the formula. It is now easy to see that the values of all vertices can be shifted down uniformly by the value $(d_m - m)$ to get a model for φ' and hence also for φ.

Assuming a binary encoding for the constants, the size of φ' can be exponential in φ. Hence we have:

Corollary 9.4 *The satisfiability problem for* $\mathrm{CLTL}^{con}(\mathbb{Z}, <, =)$ *is solvable in* EXPSPACE.

Few remarks about the logic $\mathrm{CLTL}^{con}(\mathbb{Z}, <, =)$ are in order.

(1) We conjectured in [2,52] that satisfiability for $\mathrm{CLTL}^{con}(\mathbb{Z}, <, =)$ is indeed still in PSPACE. By an appropriate extension of the notion of symbolic models, we have shown in [31] that it is the case.
(2) We note that the satisfiability problem for $\mathrm{CLTL}(\mathcal{N})$ and $\mathrm{CLTL}^{con}(\mathcal{N})$ can be reduced to that of $\mathrm{CLTL}^{con}(\mathcal{Z})$ by adding the constraint

$$\Box(\bigwedge_{x \in voc(\varphi)} (x > 0)).$$

Thus this gives us an alternate solution to the satisfiability problem for $\mathrm{CLTL}(\mathcal{N})$ and provides a PSPACE upper bound.
(3) A CTL variant of $\mathrm{CLTL}^{con}(\mathbb{Z}, <, =)$ is known to be undecidable [20, Theorem 1] and our complexity upper bound in Corollary 9.4 refines results in [20].

9.3 Other extensions

The satisfiability results of this paper can be easily extended to handle the past-time operators O^{-1} (previous) and S (since) in the temporal language. Indeed, since we use automata-based techniques, the treatment of LTL with those past-time operators from [53] can be used in our framework to obtain a PSPACE upper bound. This answers an open question from [1].

Actually, for any extension L of LTL obtained by adding a finite amount of MSO-definable temporal operators, one can show that $\mathrm{L}(\mathcal{D})$ satisfiability is in PSPACE as soon as $\mathcal{D}$ satisfies completion, non-triviality, and allows frame-checking in PSPACE. Indeed, instead of considering the Büchi automaton $\mathcal{A}^{\mathrm{LTL}}_{\varphi}$, one builds the automaton $\mathcal{A}^{\mathrm{L}}_{\varphi}$ based on the developments in [54].

10 Undecidable Extensions

As seen in the previous section, the decidability results from Secs. 4–7 are quite robust when the logical language is enriched. However, adding even basic *quantitative* relations (like $=_{+1}$) to the constraint system quickly leads to undecidability.

More generally, we define below three abstract conditions for a constraint system to admit implicitly a counting mechanism which leads to undecidability. A constraint system $\mathcal{D}$ is said to admit an *implicit counting mechanism* if the following conditions are met:

(1) $\mathcal{D}$ contains the equality predicate,
(2) $\mathcal{D}$ contains a binary relation R such that
   (a) $R = \{(x, y) \in D^2 : f(x) = y\}$ for some injective map $f : D \to D$,
   (b) (D, R) is a DAG.

Observe that for any d in D, the set $\{f^i(d) : i \in \mathbb{N}\}$ with the relation R (i.e. $d \xrightarrow{R} f^1(d) \xrightarrow{R} \ldots f^i(d) \xrightarrow{R} \ldots$) is isomorphic to $(\mathbb{N}, <)$.

For every $D \in \{\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}\}$ and for every $i \in D \setminus \{0\}$, the constraint system $(D, =, =_{+i})$ has an implicit counting mechanism, where $n =_{+i} n'$ iff $n = n' + i$. Similarly, the constraint system $(D \setminus \{0\}, =, =_{\times i})$ admits an implicit counting mechanism where $n =_{\times i} n'$ iff $n = n' \times i$ with $i \neq -1, 1, 0$. Similar conditions are used to encode natural numbers in classical predicate logic, see e.g. [55, Sect. 4.3.1].

Theorem 10.1 *The satisfiability problem for* CLTL($\mathcal{D}$) *is undecidable for every constraint system $\mathcal{D}$ with an implicit counting mechanism.*

Proof The proof is based on the proof of [56, Theorem 2, Case 2] (see also [57, Theorem 5] and [13, Theorem 3]) reducing the recurring computation problem for nondeterministic 2-counter machines.

A nondeterministic 2-counter machines consists of two counters C_0 and C_1, and a sequence of n instructions. The lth instruction is either of the form

- *add*$\langle l, C_i, l', l''\rangle$ meaning "add 1 to the counter C_i and go to either instruction l' or instruction l''" or of the form
- *sub*$\langle l, C_i, l', l''\rangle$ meaning "if $C_i \neq 0$, then subtract 1 from the counter C_i and go to the instruction l', otherwise go to the instruction l''".

A configuration is a triple $\langle ic, n_0, n_1 \rangle \in \{1, \ldots, n\} \times \mathbb{N} \times \mathbb{N}$ where ic is the instruction counter and n_i is the value of the counter C_i. A computation is an infinite sequence of configurations starting at the initial configuration $\langle 1, 0, 0 \rangle$ and such that two successive configurations are admissible according to the instructions of the machine. A computation is recurring iff it contains infinitely many configurations with the value counter being 1 (Büchi acceptance condition).

In order to encode the configurations, we shall use the variables ic, $\mathsf{n_1}$, and $\mathsf{n_2}$. However, since $\mathcal{D}$ does not necessarily have constant values, we shall introduce additional variables with constant values throughout the models:

- $\mathsf{d_1}, \ldots, \mathsf{d_n}$ to encode the n different possible values of the instruction counter;
- $\mathsf{d_{n+1}}$ and $\mathsf{d_{n+2}}$ the initial values of the counters C_0 and C_1, respectively.

Now, we define a formula that enforces that the interpretation of $\mathsf{d_1}$, $\ldots$, $\mathsf{d_n}$ are distinct and constant throughout the model. The formula φ_{inst} does the job:

$$\varphi_{inst} \stackrel{\text{def}}{=} \overbrace{\bigwedge_{1 \leq i < j \leq n} (\neg(\mathsf{d}_i = \mathsf{d}_j))}^{d_1,\ldots,d_n \text{ are distinct}} \wedge \Box(\bigwedge_{1 \leq i \leq n} \mathsf{d}_i = O\mathsf{d}_i).$$

In a similar way, we define a formula that enforces that $\mathsf{d_{n+1}}$ and $\mathsf{d_{n+2}}$ are constant values. The formula $\varphi_{init\text{-c}}$ does the job:

$$\varphi_{init\text{-}c} \stackrel{\text{def}}{=} \Box(\mathsf{d}_{n+1} = O\mathsf{d}_{n+1} \wedge \mathsf{d}_{n+2} = O\mathsf{d}_{n+2}).$$

Observe that for these values of the counters, we do not require that $\mathsf{d_{n+1}} \neq \mathsf{d_{n+2}}$.

Now we define a formula stating what is the first instruction counter and what are the initial values of the counters:

$$\varphi_{init} \stackrel{\text{def}}{=} (\mathsf{ic} = \mathsf{d}_1) \wedge (\mathsf{n}_0 = \mathsf{d}_{n+1}) \wedge (\mathsf{n}_1 = \mathsf{d}_{n+2}).$$

It remains to specify how the values of the current configurations evolve along the ω-sequence of states. We need to specify how to encode the relationships between two successive configurations. For each instruction l, we will define a formula ψ_l that specifies its effects on the instruction counter and on the counters. For instance, adding one to a counter C_i will be encoded by the atomic formula $\mathsf{R}(\mathsf{n}_i, O\mathsf{n}_i)$.

If the lth instruction is of the form $add\langle l, C_i, l', l'' \rangle$, the formula μ_l is defined as follows:

$$\overbrace{((\mathsf{d}_{l'} = O\mathsf{ic}) \vee (\mathsf{d}_{l''} = O\mathsf{ic}))}^{\text{go to either } l' \text{ or } l''} \wedge \overbrace{\mathsf{R}(\mathsf{n}_i, O\mathsf{n}_i)}^{\text{add 1 to } C_i} \wedge \overbrace{(O\mathsf{n}_{1-i} = \mathsf{n}_{1-i})}^{C_{1-i} \text{ does not change}} .$$

Similarly, if the lth instruction is of the form $sub\langle l, C_i, l', l''\rangle$, the formula μ_l is defined as follows:

$$(\overbrace{\neg(\mathsf{n}_i = \mathsf{d}_{n+1+i})}^{C_i \neq 0} \Rightarrow \overbrace{(\mathsf{d}_{l'} = O\mathsf{ic})}^{\text{go to } l'} \wedge \overbrace{\mathsf{R}(O\mathsf{n}_i, \mathsf{n}_i)}^{\text{sub 1 from } C_i} \wedge \overbrace{(O\mathsf{n}_{1-i} = \mathsf{n}_{1-i})}^{C_{1-i} \text{ does not change}})$$

$$\wedge(\overbrace{(\mathsf{n}_i = \mathsf{d}_{n+1+i})}^{C_i = 0} \Rightarrow \overbrace{(\mathsf{d}_{l''} = O\mathsf{ic})}^{\text{go to } l''} \wedge \overbrace{(O\mathsf{n}_0 = \mathsf{n}_0)}^{C_0 \text{ does not change}} \wedge \overbrace{(O\mathsf{n}_1 = \mathsf{n}_1)}^{C_1 \text{ does not change}}).$$

It remains to express that a computation is recurring with the formula φ_{rec} below:

$$\Box\Diamond(\mathsf{ic} = \mathsf{d}_1).$$

We define the final formula φ_M to be the formula below:

$$\varphi_{inst} \wedge \varphi_{init\text{-}c} \wedge \wedge\varphi_{rec} \wedge \Box(\bigwedge_{1 \leq l \leq n} ((\mathsf{ic} = \mathsf{d}_l) \Rightarrow \mu_l)).$$

Using the fact that $\mathcal{D}$ has an implicit counting mechanism, it is easy to see that M has a recurring computation iff φ_M is CLTL($\mathcal{D}$) satisfiable. $\Box$

Corollary 10.2 *For each $D \in \{\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{Q}_+\mathbb{R}, \mathbb{R}_+\}$ and for every $i \in D \setminus \{0\}$, satisfiability for* CLTL$(D, =, =_{+i})$ *is undecidable.*

The above Σ^1_1-hardness results implies that CLTL($\mathcal{D}$) is not recursively enumerable and therefore there cannot be complete proof systems for it.

A slight adaptation of the proof of Theorem 10.1 allows us to show the following result.

Corollary 10.3 *For every $D \in \{\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{Q}_+, \mathbb{R}, \mathbb{R}_+\}$ and for every $i \in D \setminus \{0, 1, -1\}$, satisfiability* CLTL$(D, =, =_{\times i})$ *is undecidable.*

Proof The only difference with the proof of Theorem 10.1 is that we have to enforce that d_{n+1} and d_{n+2} are not equal to zero which can be expressed by the formula $\neg(\mathsf{d}_{n+1} =_{\times i} \mathsf{d}_{n+1}) \wedge \neg(\mathsf{d}_{n+2} =_{\times i} \mathsf{d}_{n+2})$. $\Box$

Finally, we show that Theorem 10.1 holds even when we restrict ourselves to just two variables. This follows from the general result below.

Lemma 10.4 *Let $\mathcal{D} = (D, =, \ldots)$ be a concrete domain equipped with equality. Then* CLTL($\mathcal{D}$)*-satisfiability can be reduced in logarithmic space to* CLTL($\mathcal{D}$)*-satisfiability restricted to formulas with at most two variables.*

Proof Let φ be a CLTL($\mathcal{D}$) formula of O-length m with variables $x_1, \ldots, x_n$ for some $n \geq 3$. We set $M = n \times m$. We shall build a CLTL($\mathcal{D}$) formula φ' (in logarithmic space in $|\varphi|$) such that φ is CLTL($\mathcal{D}$) satisfiable iff φ' is CLTL($\mathcal{D}$) satisfiable.

If D is a singleton, every variable in φ can be replaced by x leading to φ'. In the sequel, we assume that D has at least two elements.

A sequence $v_0, \ldots, v_{m-1}$ of valuations of the form $\{x_1, \ldots, x_n\} \rightarrow D$, viewed as a sequence of m states in a model for φ, will be encoded by M successive valuations $u_1, \ldots, u_M$ over $\{x, y\}$ such that

- for every $1 \leq i \leq M$, $u_i(x) = v_j(x_{k+1})$ where j and k are such that $i = j \times n + k$ and $k < n$.
- $u_1(y) = u_1(x)$ and for every $2 \leq i \leq M$, $u_i(y) \neq u_i(x)$.

The second variable y has a unique function: to be able to count the positions of the states modulo M with the satisfiability of the atomic constraint $x = y$. The first formula that we consider allows us to access states whose positions are multiple of M. Basically, the first states of $n \times m$ states sequences are exactly the states satisfying $x = y$. The formula φ_{mult} does the job:

$$\varphi_{mult} \stackrel{\text{def}}{=} (x = y) \wedge \bigwedge_{1 \leq i \leq M-1} \neg(O^i x = O^i y) \wedge \Box(x = y \Leftrightarrow O^M x = O^M y).$$

Indeed, we have $\sigma \models \varphi_{mult}$ iff for all i, ($\sigma[i, \infty] \models x = y$ iff $i = 0 \bmod M$). The formula φ_{shift} below states that for $2 \times M$ consecutive valuations over $\{x, y\}$ $u_1, \ldots, u_M, u'_1, \ldots, u'_M$, $u_{n+1}, \ldots, u_M = u'_1, \ldots, u'_{M-n}$:

$$\Box(x = y \Rightarrow \bigwedge_{i=1}^{M-n} O^{n+i} x = O^{M+i} x)$$

We now define the map t from subformulas of φ to formulas over $\{x, y\}$:

- t is homomorphic with respect to the Boolean operators;
- $t(R(O^{j_1} x_{i_1}, \ldots, O^{j_\alpha} x_{i_\alpha})) = R(O^{(j_1 - 1) \times n + i_1} x, \ldots, O^{(j_\alpha - 1) \times n + i_\alpha} x)$;
- $t(O\psi) = O(\neg(x = y) U((x = y) \wedge t(\psi)))$;
- $t(\psi U \mu) = ((x = y) \Rightarrow t(\psi)) U((x = y) \wedge t(\mu))$.

Now one can easily show that φ is CLTL($\mathcal{D}$) satisfiable iff $\varphi_{mult} \wedge t(\varphi)$ is CLTL($\mathcal{D}$) satisfiable. $\Box$

When only one variable is involved, undecidability can still be established following [33].

11 Conclusion

We have studied some natural satisfiability and model-checking problems related to a simple constraint temporal logic. The technique used in this paper is automata-theoretic and its novelty lies in its applicability in the face non-regularity of the set of models described by formulas in the logics. The automata-theoretic approach also leads to transparent generalizations of the techniques for some extensions of the logic.

Among the issues we have not paid much attention to are applications of these results, particularly in the area of automated program verification. For example one could consider model-checking properties of finitely presented infinite state programs like timed automata [58] (where clocks can be modelled as real-valued variables) or integer-valued programs [59]. Another application one could investigate would be in the area of program synthesis. As mentioned in the the introduction, the theory developed here can in principle be used to synthesize programs satisfying a given CLTL specification.

Another interesting problem we would like to investigate is the decidability of the logic when we allow more expressive quantifiers like $y = \Diamond x$ expressing that some future value of x is equal to the current value of y. These quantifiers are related to the "freeze quantifiers" used in some real-time logics [57] (see also recent developments in [35,36]), and one may be able to exploit the techniques used there.

References

[1] P. Balbiani, J. Condotta, Computational complexity of propositional linear temporal logics based on qualitative spatial or temporal reasoning, in: Frontiers of Combining Systems (FroCoS'02), Vol. 2309 of Lecture Notes in Computer Science, Springer, 2002, pp. 162–176.

[2] S. Demri, D. D'Souza, An automata-theoretic approach to constraint LTL, in: FST&TCS'02, Kanpur, Vol. 2256 of Lecture Notes in Computer Science, Springer, Berlin, 2002, pp. 121–132.

[3] A. Pnueli, The temporal logic of programs, in: Proc. 18th IEEE Symposium on Foundation of Computer Science, 1977, pp. 46–57.

[4] B. Bennett, F. Wolter, M. Zakharyaschev, Multi-dimensional modal logic as a framework for spatio-temporal reasoning, Applied Intelligence 17 (3) (2002) 239–251.

[5] F. Wolter, M. Zakharyaschev, Qualitative spatio-temporal representation and reasoning: a computational perspective, in: G. Lakemeyer, B. Nebel (Eds.), Exploring Artificial Intelligence in the New Millenium, Morgan Kaufmann, 2002, pp. 175–216.

[6] F. Baader, P. Hanschke, A scheme for integrating concrete domains into concept languages, in: 12th International Joint Conference on Artificial Intelligence, Sydney, Australia, 1991, pp. 452–457.

[7] C. Lutz, Interval-based temporal reasoning with general TBoxes, in: 17th International Joint Conference on Artificial Intelligence (IJCAI'01), Morgan-Kaufmann, 2001, pp. 89–94.

[8] C. Lutz, Reasoning with the description logic ALCF(D), Logic Journal of the IGPL 10 (5) (2002) 535–568.

[9] J. F. Allen, Maintaining knowledge about temporal intervals, Communications of the ACM 26 (11) (1983) 832–846.

[10] D. Randell, Z. Cui, A. Cohn, A spatial logic based on regions and connection, in: 3rd Conference on Knowledge Representation and Reasoning, Morgan Kaufman, 1992, pp. 165–176.

[11] B. Bennett, Spatial reasoning with propositional logic, in: 4th Conference on Knowledge Representation and Reasoning, Morgan Kaufman, 1994, pp. 51–62.

[12] F. Wolter, M. Zakharyaschev, Spatio-temporal representation and reasoning based on RCC-8, in: KR'00, Morgan Kaufmann, 2000, pp. 3–14.

[13] H. Comon, V. Cortier, Flatness is not a weakness, in: 14 Int. Workshop Computer Science Logic, Vol. 1862 of Lecture Notes in Computer Science, Springer, 2000, pp. 262–276.

[14] M. Vardi, P. Wolper, Reasoning about infinite computations, Information and Computation 115 (1994) 1–37.

[15] J. Büchi, On a decision method in restricted second order arithmetic, in: E. et al (Ed.), ICM'60, Stanford University Press, 1960, pp. 1–11.

[16] H. Comon, Y. Jurski, Multiple counters automata, safety analysis and Presburger analysis, in: Computer-Aided Verification'98, Vol. 1427 of Lecture Notes in Computer Science, Springer, Berlin, 1998, pp. 268–279.

[17] O. Ibarra, J. Su, Z. Dang, T. Bultan, A. Kemmerer, Counter machines: Decidable properties and applications to verification problems, in: MFCS'00, Vol. 1893 of Lecture Notes in Computer Science, Springer, 2000, pp. 426–435.

[18] A. Finkel, J. Leroux, How to compose Presburger accelerations: Applications to broadcast protocols, in: M. Agrawal, A. Seth (Eds.), FST&TCS'02, Kanpur, Vol. 2256 of Lecture Notes in Computer Science, Springer, Berlin, 2002, pp. 145–156.

[19] Z. Dang, P. S. Pietro, R. Kemmerer, Presburger liveness verification of discrete timed automata, Theoretical Computer Science 299 (2003) 413–438.

[20] K. Čerāns, Deciding properties of integral relational automata, in: ICALP, Vol. 820 of Lecture Notes in Computer Science, Springer, 1994, pp. 35–46.

[21] A. Bouajjani, R. Echahed, P. Habermehl, On the verification problem of nonregular properties for nonregular processes, in: LICS'95, 1995, pp. 123–133.

[22] V. Bruyère, E. Dall'Olio, J. Raskin, Durations, parametric model-checking in timed automata with presburger arithmetic, in: STACS'03, Vol. 2607 of Lecture Notes in Computer Science, Springer, 2003, pp. 687–698.

[23] D. Gabbay, A. Kurucz, F. Wolter, M. Zakharyaschev, Many-dimensional modal logics: theory and applications, Cambridge University Press, 2003.

[24] I. Hodkinson, R. Kontchakov, A. Kurucz, F. Wolter, M. Zakharyaschev, On the computational complexity of decidable fragments of first-order linear temporal logics, in: 10th Int. Symp. Temporal Representation and Reasoning and 4th Int. Conf. Temporal Logic (TIME-ICTL), IEEE, 2003, pp. 91–98.

[25] D. Gabelaia, R. Kontchakov, A. Kurucz, F. Wolter, M. Zakharyaschev, On the computational complexity of spatio-temporal logics, in: FLAIRS'03, St Augustine, Florida, 2003, pp. 460–464.

[26] C. Lutz, M. Miličić, A tableau algorithm for description logics with concrete domains and GCIs, in: TABLEAUX'05, Vol. 3702 of Lecture Notes in Computer Science, Springer, 2005, pp. 201–216.

[27] R. Dechter, From local to global consistency, Artificial Intelligence (1992) 87–107.

[28] C. Lutz, Description logics with concrete domains—a survey, in: Advances in Modal Logics Volume 4, King's College Publications, 2003, pp. 265–296.

[29] C. Lutz, NEXPTIME-complete description logics with concrete domains, ACM Transactions on Computational Logic 5 (4) (2004) 669–705.

[30] S. Demri, LTL over integer periodicity constraints, Theoretical Computer Science 360 (1–3) (2006) 96–123.

[31] S. Demri, R. Gascon, Verification of qualitative $\mathbb{Z}$-constraints, in: CONCUR'05, Vol. 3653 of Lecture Notes in Computer Science, Springer, 2005, pp. 518–532.

[32] L. Bozzelli, R. Gascon, Branching-time temporal logic extended with Presburger constraints, in: LPAR'06, Lecture Notes in Computer Science, Springer, 2006, to appear.

[33] S. Demri, R. Gascon, The effects of bounding syntactic resources on Presburger LTL, Tech. Rep. LSV-06-5, LSV, 36 pages (February 2006).

[34] S. Demri, Linear-time temporal logics with Presburger constraints: An overview, Journal of Applied Non-Classical Logics To appear.

[35] S. Demri, R. Lazić, D. Nowak, On the freeze quantifier in constraint LTL: decidability and complexity, Information and Computation To appear in a special issue dedicated to selected papers from TIME'05.

[36] S. Demri, R. Lazić, LTL with the freeze quantifier and register automata, in: LICS'06, IEEE, 2006, pp. 17–26.

[37] R. Lazić, Safely freezing LTL, in: FSTTCS'06, Lecture Notes in Computer Science, Springer, 2006, to appear.

[38] M. Bojańczyk, A. Muscholl, T. Schwentick, L. Segoufin, C. David, Two-variable logic on words with data, in: LICS, IEEE, 2006, pp. 7–16.

[39] A. Lisitsa, I. Potapov, Temporal logic with predicate λ-abstraction, in: TIME, IEEE, 2005, pp. 147–155.

[40] A. Deutsch, L. Sui, V. Vianu, Specification and verification of data-driven web services, in: PODS'04, Paris, 2004, pp. 71–82.

[41] M. Vardi, P. Wolper, An automata theoretic approach to automatic program verification, in: Logic in Computer Science, IEEE, 1986, pp. 332–334.

[42] B. Bérard, Untiming timed languages, Information Processing Letters 55 (1995) 129–135.

[43] W. Thomas, Automata on infinite objects, in: J. van Leeuwen (Ed.), Handbook of Theoretical Computer Science, Volume B, Formal Models and Semantics, Elsevier, 1990, pp. 133–191.

[44] P. Sistla, E. M. Clarke, The complexity of propositional linear temporal logics, Journal of the Association for Computing Machinery 32 (3) (1985) 733–749.

[45] M. Vilain, H. Kautz, Constraint propagation algorithms for temporal reasoning, in: T. Kehler, S. Rosenschein (Eds.), 5th National Conference on Artificial Intelligence (AAAI'86), Morgan Kaufmann, 1986, pp. 372–382.

[46] S. Safra, On the complexity of ω-automata, Proc. 29th IEEE Symp. on Foundations of Computer Science (1988) 319–327.

[47] M. Vardi, An automata-theoretic approach to linear temporal logic, in: F. Moller, G. Birtwistle (Eds.), Logics of Concurrency: Structure versus Automata, Lecture Notes in Computer Science, Vol. 1043. Springer, Berlin, 1996, pp. 238–266.

[48] L. Fribourg, M. V. Peixoto, Concurrent constraint automata, Tech. Rep. LIENS-93-10, Ecole Normale Supérieure (1993).

[49] M. Roger, J. Goubault-Larrecq, Log auditing through model-checking, in: 14th Computer Security Foundations Workshop (CSFW'01), Cape Breton, Nova Scotia, Canada, IEEE Computer Society Press, 2001, pp. 220–236.

[50] P. Revesz, Introduction to Constraint Databases, Springer, New York, 2002.

[51] P. Wolper, Temporal logic can be more expressive, Information and Computation 56 (1983) 72–99.

[52] S. Demri, D. D'Souza, An automata-theoretic approach to constraint LTL, Tech. Rep. LSV-03-11, LSV, 40 pages. (August 2003).

[53] N. Markey, F. Laroussinie, P. Schnoebelen, Temporal logic with forgettable past, in: 17th IEEE Symp. Logic in Computer Science (LICS'2002), Copenhagen, Denmark, IEEE Comp. Soc. Press, 2002, pp. 383–392.

[54] P. Gastin, D. Kuske, Satisfiability and model checking for MSO-definable temporal logics are in PSPACE, in: CONCUR'03, Marseille, France, Vol. 2761 of Lecture Notes in Computer Science, Springer, 2003, pp. 222–236.

[55] E. Börger, E. Grädel, Y. Gurevich, The Classical Decision Problem, Perspectives in Mathematical Logic, Springer, 1997.

[56] R. Alur, T. Henzinger, Real-time logics: complexity and expressiveness, Information and Computation 104 (1) (1993) 35–77.

[57] R. Alur, T. Henzinger, A really temporal logic, Journal of the Association for Computing Machinery 41 (1) (1994) 181–204.

[58] R. Alur, D. Dill, A theory of timed automata, Theoretical Computer Science 126 (1994) 183–235.

[59] M. Müller-Olm, H. Seidl, Computing interprocedurally valid relations in affine programs, Tech. rep., University of Trier (2003).