Verification of qualitative Z constraints

Stéphane Demri, Régis Gascon

Verification of qualitative $\mathbb{Z}$ constraints

Stéphane Demri and Régis Gascon

LSV, ENS Cachan, CNRS, INRIA
email: {demri,gascon}@lsv.ens-cachan.fr

Abstract. We introduce an LTL-like logic with atomic formulae built over a constraint language interpreting variables in $\mathbb{Z}$. The constraint language includes periodicity constraints, comparison constraints of the form $x = y$ and $x < y$, is closed under Boolean operations and admits a restricted form of existential quantification. Such constraints are used for instance in calendar formalisms or abstractions of counter automata by using congruences modulo some power of two. Indeed, various programming languages perform arithmetic operators modulo some integer. We show that the satisfiability and model-checking problems (with respect to an appropriate class of constraint automata) for this logic are decidable in polynomial space improving significantly known results about its strict fragments. This is the largest set of qualitative constraints over $\mathbb{Z}$ known so far, shown to admit a decidable LTL extension.

1 Introduction

Model-checking infinite-state systems. The verification of systems with an infinite number of states has benefited from the numerous decidable model-checking problems for infinite-state systems, including timed automata [AD94], infinite transition graphs [MS85,Cau03], or subclasses of counter systems (see e.g. [CJ98]). There exist numerous techniques to prove decidability such as finite partition of the infinite domain, well-structured systems, Presburger definable reachability sets or reduction to the monadic second-order theory of the binary tree... Since it is often possible to find a reduction from the halting problem for Minsky machines [Min67], undecidability is often easy to prove. Decidability can sometimes be regained by naturally restricting the class of models (see e.g. the flatness condition in [CJ98,FL02]) or by considering fragments of the specification language (for instance reachability questions). Symbolic representations of infinite sets of states are often the key argument to get decidability (see e.g. [HMR05]).

Systems with variables interpreted in $\mathbb{Z}$. Counter machines are operational models that have found numerous applications in the verification of infinite-state systems, including broadcast protocols (see e.g. [EFM99,FL02]) and programs with pointer variables [BFLS06,BBH$^+$06]. They consist of a structure with a finite set of control states augmented with a finite set of variables interpreted either in $\mathbb{Z}$ or $\mathbb{N}$ (counters). Though this class of automata has numerous undecidable model-checking problems such as the reachability problem, many subclasses have been shown to be decidable:

1. reversal-bounded multicounter machines [Iba78],
2. flat counter systems with affine update functions forming a finite monoid (see e.g. [Boi98,FL02,BFLP03]),
3. flat counter systems [CJ98,BIL06] (weaker class of Presburger guards but no condition on the monoid),
4. admissible Presburger counter systems [DFGvD06],
5. constraint automata with qualitative constraints on $\mathbb{Z}$ [DD07].

Our motivation. Constraint automata with qualitative constraints on $\mathbb{Z}$ are quite attractive operational models since they can be viewed as abstractions of counter automata where increments and decrements are abstracted by operations modulo some power of two. Indeed, common programming languages perform arithmetic operators for integers modulo 2^k [MOS05], where k is typically equal to 32 or 64. So, such an abstraction is well-suited to check safety properties about the original counter system. In this paper, we introduce a class of constraint automata with a language of qualitative constraints as rich as possible and a companion LTL-like logic in order to perform model-checking on these models. Our framework should be able to deal both with modulo abstractions (see e.g. [CGL94,LS01]) and with integer periodicity constraints used in logical formalisms to deal with calendars [LM01]. By qualitative constraint, we mean constraints that are interpreted as a non-deterministic binary relation, like $x < y$ or $x \equiv_{2^k} y + 5$ (the relationship between x and y is not sharp).

Our contribution. We introduce CLTL(IPC*) as an extension of LTL over the constraint language IPC*, whose expressions are Boolean combinations of IPC^{++} constraints from [Dem06] and constraints of the form $x < y$. The language inherits from IPC^{++} closure under Boolean operators and first-order quantification. We impose that no constraint of the form $x < y$ occurs in the scope of a quantifier; otherwise incrementation is definable and this leads to undecidability. In this paper, we show that adding the single type of constraints $x < y$ leads to many technical complications but not to undecidability. We also introduce the class of IPC*-automata defined as finite-state automata with transitions labelled by CLTL(IPC*) formula à la Wolper [Wol83]. Such structures can be viewed as labelled transition systems obtained by abstraction of counter automata.

Constraint LTL over IPC^{++} is already known to be in PSPACE in [Dem06] whereas constraint LTL over constraints of the form either $x = y$ or $x < y$ is shown to be also in PSPACE in [DD07]. Though both proofs use reductions to the nonemptiness problem for Büchi automata, following the approach in [VW94], they are of different nature: in [Dem06] the complexity upper bound is obtained by a finite model property argument whereas in [DD07] approximations of classes of symbolic models are considered because some formulae can generate non ω-regular classes of symbolic models. We show that the model-checking and satisfiability problems for the logic CLTL(IPC*) are still PSPACE-complete which generalizes and unifies these results. We improve what is done for constraint LTL over the domain $\langle \mathbb{Z}, <, = \rangle$ by considering both new constraints of the form $x \leq d$

with $d \in \mathbb{Z}$ and periodicity constraints. The optimal treatment of the constants introduced in this language is our main technical contribution. As a corollary, we establish that LTL model-checking over integral relational automata [Čer94] is PSPACE-complete. Hence, even though IPC* is a powerful language of qualitative constraints, the PSPACE upper bound is preserved in CLTL(IPC*). Moreover, past-time operators can be added for free in our formalism thanks to [GK03] (the PSPACE bound is still preserved).

Related work. Reachability problems for subclasses of counter systems have been addressed in numerous works [Iba78,CJ98,FS00,FL02,BFLP03] (see also richer questions in [BEM97,DI02,DPK03,JKMS04]). In this paper, we consider a full LTL-like language used as a specification language which is not restricted to reachability questions, and we have no restriction on the structure of the models unlike [CJ98,DFGvD06]. However, the atomic formulae of the specification language are restricted to qualitative constraints. If we give up the decidability requirement, other extensions of LTL with Presburger constraints can be found in [BEH95,CC00,ID01].

Extensions of LTL over concrete domains, not only restricted to variables interpreted in $\mathbb{Z}$, have also been considered in [WZ00,BC02,DD07,GKK$^+$03,Dem06] where often PSPACE-completeness results are shown. The idea of building LTL over a language of constraints, although already present in first-order temporal logics, stems from the use of concrete domains for description logics [BH91,Lut04]. The language CLTL(IPC*) extends the different LTL-like fragments from the works [Čer94,LM01,Dem06].

The underlying constraint language of CLTL(IPC*) includes integer periodicity constraints, a special class of Presburger constraints that have found applications in many logical formalisms such as abstractions with congruences modulo an integer of the form 2^k (see e.g. [CGL94,MOS05]), logical formalisms dealing with calendars (see e.g. [LM01,Pup06]) and temporal reasoning in database access control [BBFS98]. Such constraints can also be found in real-time logics, see e.g. [AH94]. Our approach of constraint LTL makes explicit the constraints on variables, similarly to the explicit clock approach from [HLP90]. Furthermore, the class of IPC*-automata we introduce generalizes the class of integral relational automata from [Čer94] (see details in Appendix A).

Finally, the concept of symbolic models used in the paper has similarities with untimed languages recognized by some classes of timed machines. In [Bér95], sufficient conditions to get regular untimed languages from timed machines are exhibited.

Plan of the paper. The rest of the paper is organized as follows. In Section 2, we introduce the logic CLTL(IPC*) and the class of IPC*-automata. We present the model-checking and satisfiability problems and discuss expressiveness issues. In Section 3, we analyze the computational complexity of the satisfiability problem of the underlying constraint language IPC*. We also provide a symbolic representation of the valuations that is used later in the decidability proof. Section 4 contains a characterization of the sequences of symbolic valuations that

admit concrete models (valuation sequence). We show that testing the existence of some concrete model is an ω-regular property when considering ultimately periodic sequences. In Section 5, we show that given a CLTL(IPC*) formula ϕ, one can build a Büchi automaton $\mathcal{A}_\phi$ over the alphabet of symbolic valuations such that ϕ is CLTL(IPC*) satisfiable iff $\mathrm{L}(\mathcal{A}_\phi)$ is non-empty. Moreover, we establish that nonemptiness of $\mathrm{L}(\mathcal{A}_\phi)$ can be checked in polynomial space in $|\phi|$. Section 6 contains concluding remarks.

This paper is a completed version of [DG05].

2 The logic CLTL(IPC*)

2.1 Language of constraints

Let $\mathrm{VAR} = \{x_0, x_1, \ldots\}$ be a countably infinite set of variables (in some places for ease of presentation, VAR will denote a particular finite set of variables). The language of constraints IPC* is defined by the following grammar:

$$\xi ::= \theta \mid x < y \mid \xi \wedge \xi \mid \neg\xi$$

$$\theta ::= \; x \equiv_k [c_1, c_2] \mid x \equiv_k y + [c_1, c_2] \mid x = y \mid x < d \mid x = d \mid \theta \wedge \theta \mid \neg\theta \mid \exists x \; \theta$$

where $x, y \in \mathrm{VAR}$, $k \in \mathbb{N}\setminus\{0\}$, $c_1, c_2 \in \mathbb{N}$ and $d \in \mathbb{Z}$. In the following, the symbol $\sim$ is used to mean either $=$ or $<$. We write IPC^{++} to denote the restriction of the language to constraints ranged over by θ, Z^c to constraints of the form either $x \sim y$ or $x \sim d$ and Z to constraints of the form $x \sim y$ where $x, y \in \mathrm{VAR}$, $d \in \mathbb{Z}$ and $\sim \in \{<, =\}$. A valuation $v : \mathrm{VAR} \to \mathbb{Z}$ is a map that associates a value to each variable and the satisfaction relation $v \models_\star \xi$ is defined in the standard way:

- $v \models_\star x \sim y$ iff $v(x) \sim v(y)$;
- $v \models_\star x \sim d$ iff $v(x) \sim d$;
- $v \models_\star x \equiv_k [c_1, c_2]$ iff $v(x) - c = kd$ for some $c_1 \leq c \leq c_2$ and $d \in \mathbb{Z}$;
- $v \models_\star x \equiv_k y + [c_1, c_2]$ iff $v(x) - v(y) - c = kd$ for some $c_1 \leq c \leq c_2$ and $d \in \mathbb{Z}$;
- $v \models_\star \xi \wedge \xi'$ iff $v \models_\star \xi$ and $v \models_\star \xi'$;
- $v \models_\star \neg\xi$ iff $v \not\models_\star \xi$;
- $v \models_\star \exists x \; \xi$ iff there is $d \in \mathbb{Z}$ such that $v[x \leftarrow d] \models_\star \xi$ where $v[x \leftarrow d](x') = v(x')$ if $x \neq x'$ and $v[x \leftarrow d](x) = d$.

We will shortly write $x \equiv_k c$ instead of $x \equiv_k [c, c]$ and $x \equiv_k y + c$ instead of $x \equiv_k y + [c, c]$. Given a set of IPC*-constraints X, we note $v \models_\star X$ whenever $v \models_\star \xi$ for every $\xi \in X$. A constraint ξ is satisfiable iff there is a valuation v such that $v \models_\star \xi$. Two constraints are equivalent iff they are satisfied by the same valuations.

Lemma 1. **(I)** *The satisfiability problem for* IPC* *is* PSPACE*-complete.*
(II) *For every constraint in* IPC* *there exists an equivalent quantifier-free constraint in* IPC**.*

Proof. (I) Satisfiability for IPC^{++} is PSPACE-complete [Dem06] whereas satisfiability for Z is NLOGSPACE-complete. Since constraints in IPC* are Boolean combinations of IPC^{++} and Z constraints, IPC* satisfiability is in PSPACE by simply adapting the proof of [Dem06, Theorem 3]. PSPACE-hardness is a consequence of the PSPACE-hardness of IPC^{++}.
(II) IPC^{++} admits quantifier elimination [Dem06] and therefore so does IPC* since Z is quantifier-free. □

2.2 Logical language

We consider the extension of the linear-time temporal logic LTL whose atomic formulae are defined from constraints in IPC* (denoted by CLTL(IPC*)). So, the language includes boolean operators as well as the classical temporal operators next (X) and until (U) of LTL. The atomic formulae are of the form $\xi[x_1 \leftarrow \mathsf{X}^{l_1} x_{j_1}, \ldots, x_r \leftarrow \mathsf{X}^{l_r} x_{j_r}]$, where ξ is a constraint of IPC* with free variables $x_1 \ldots x_r$. We substitute each occurrence of the variable x_i by $\mathsf{X}^{l_i} x_{j_i}$, which corresponds to the variable x_{j_i} preceded by l_i next symbols. Each expression of the form $\mathsf{X}^l x_j$ is called a term and represents the value of the variable x_j at the l^{th} next state. For example, $\mathsf{X}y \equiv_{2^{32}} x + 1$ and $x < \mathsf{X}y$ are atomic formulae of the logic.

The set of CLTL(IPC*) formulae ϕ is defined by the grammar below

$$\phi ::= \xi[x_1 \leftarrow \mathsf{X}^{i_1} x_{j_1}, \ldots, x_r \leftarrow \mathsf{X}^{i_r} x_{j_r}] \mid \neg\phi \mid \phi \wedge \phi \mid \mathsf{X}\phi \mid \phi \mathsf{U} \phi,$$

where ξ belongs to IPC*. The integers are encoded with a binary representation (this is important for complexity considerations).

A one-step constraint is a constraint where all the terms are of the form x and $\mathsf{X}x$ only (for some $x \in$ VAR). Given a set of constraints X included in IPC*, we write CLTL(X) to denote the restriction of CLTL(IPC*) in which the atomic constraints are built over elements of X.

A model $\sigma : \mathbb{N} \times \text{VAR} \to \mathbb{Z}$ for CLTL(IPC*) is an ω-sequence of valuations. The satisfaction relation is defined as follows:

- $\sigma, i \models \xi[x_1 \leftarrow \mathsf{X}^{i_1} x_{j_1}, \ldots, x_r \leftarrow \mathsf{X}^{i_r} x_{j_r}]$
  iff $[x_1 \leftarrow \sigma(i + i_1, x_{j_1}), \ldots, x_r \leftarrow \sigma(i + i_r, x_{j_r})] \models_\star \xi$;
- $\sigma, i \models \phi \wedge \phi'$ iff $\sigma, i \models \phi$ and $\sigma, i \models \phi'$;
- $\sigma, i \models \neg\phi$ iff $\sigma, i \not\models \phi$;
- $\sigma, i \models \mathsf{X}\phi$ iff $\sigma, i + 1 \models \phi$;
- $\sigma, i \models \phi \mathsf{U} \phi'$ iff there is $j \geq i$ s.t. $\sigma, j \models \phi'$ and for every $i \leq l < j,\ \sigma, l \models \phi$.

2.3 Satisfiability and model-checking problems

We define below the problems we consider in this paper. The definition of the satisfiability problem is standard.

Satisfiability problem :
Given a CLTL(IPC*) formula ϕ, is there a model σ such that $\sigma, 0 \models \phi$?

Note that if we extend IPC* to allow constraints of the form $x < y$ in the scope of an existential quantifier, then the satisfiability problem for the corresponding extension of CLTL(IPC*) is undecidable since the relation $x = y + 1$ is then definable and the halting problem for Minsky machines can be easily encoded.

The model-checking problem which is defined in a moment rests on a particular class of constraint automata [Rev02]. An IPC*-automaton is defined as a Büchi automaton over a finite alphabet made of CLTL(IPC*) formulae. In such an automaton, letters on transitions may induce constraints between the variables of the current state and the variables of the next state as done in [CC00]. Hence, guards and update functions are expressed in the same formalism. We are however a bit more general since we allow formulae on transitions as done in [Wol83]. Formally, an IPC*-automaton is a structure $\mathcal{A} = \langle Q, I, F, \delta \rangle$ such that:

- Q is a finite set of locations,
- $I \subseteq Q$ is a set of initial locations and $F \subseteq Q$ a set of final locations,
- $\delta \subseteq Q \times \Sigma \times Q$ where Σ is a finite set of IPC*-constraints.

We say that $\mathcal{A}$ is a restricted IPC*-automaton when Σ is a set of Boolean combination of one-step constraints. A run is an infinite sequence $q_0 \xrightarrow{\phi_0} q_1 \xrightarrow{\phi_1} q_2 \xrightarrow{\phi_2} \dots$ such that for every $i \in \mathbb{N}$, $\langle q_i, \phi_i, q_{i+1} \rangle \in \delta$. Such a run is accepting iff there is a state $q_f \in F$ such that $q_i = q_f$ for infinitely many $i \in \mathbb{N}$. In this case, the word $\phi_0 \cdot \phi_1 \cdot \phi_2 \cdot \dots$ is accepted by $\mathcal{A}$ and we write $\mathrm{L}(\mathcal{A})$ to denote the language recognized by $\mathcal{A}$ made of ω-sequences $\phi_1 \cdot \phi_2 \cdots \in \Sigma^\omega$ obtained from accepting runs. We say that a valuation sequence σ realizes a word $\phi_0 \cdot \phi_1 \cdot \phi_2 \cdot \dots$ in Σ^ω iff for every $i \geq 0$, $\sigma, i \models \phi_i$.

As an illustration, we present a (restricted) IPC*-automaton in Figure 1 which is an abstraction of the pay-phone controller from [CC00, Example 1] (x is the number of quarters that have been inserted and y measures the total communication time). An increment of a variable z is abstracted by $\mathsf{X}z \equiv_{2^{32}} z + 1 \wedge \mathsf{X}z > z$. The formula $\phi_=$ denotes $\mathsf{X}x = x \wedge \mathsf{X}y = y$. Messages are omitted because they are irrelevant here (simplifications are then possible).

Model-checking problem :
Given an IPC*-automaton $\mathcal{A}$ and a CLTL(IPC*) formula ϕ, are there an ω-word $\phi_0 \cdot \phi_1 \cdot \dots$ accepted by $\mathcal{A}$ and a model σ for ϕ that realizes $\phi_0 \cdot \phi_1 \cdot \dots$.

Note that the equivalence problem for Extended Single-String automata [LM01] can be encoded as a model-checking problem for CLTL(IPC*) (see [Dem06]).

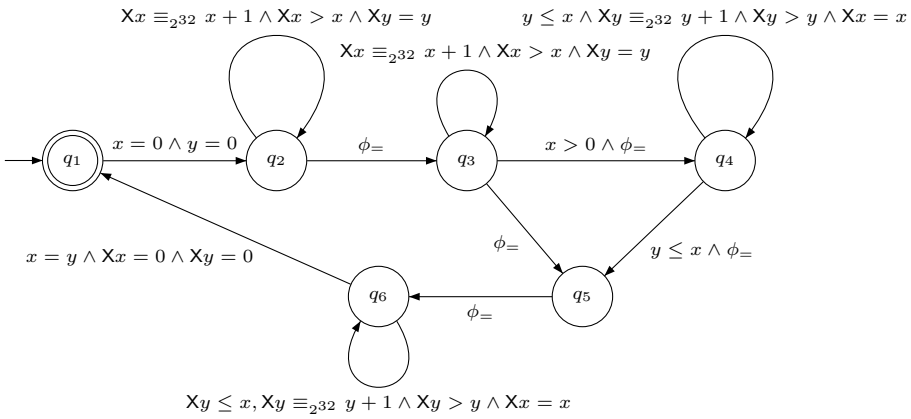

Fig. 1. A restricted IPC*-automaton

The satisfiability problem and the model-checking problem are reducible to each other in logspace following techniques from [SC85]. Indeed, satisfiability can be seen as a particular case of model-checking problem since one can build an IPC*-automaton such that every valuation sequence realizes some execution of this automaton (consider for instance the automaton that accepts the sequence $\top^\omega$). The converse reduction relies on a standard encoding of the executions of an IPC*-automaton by a CLTL(IPC*) formula, possibly introducing a new variable to encode the control states of the automaton. In the following, we only refer to the satisfiability problem but the results we prove also hold for the model-checking problem.

Let CCTL*(IPC*) denote the CTL* extension of CLTL(IPC*). The model-checking problem of the LTL fragment of the logic introduced in [Čer94] against integral relational automata is a subproblem of the model-checking problem for CLTL(IPC*). Full CCTL*(IPC*) model-checking can be shown to be undecidable by using developments in Appendix A and [Čer94]. This is actually true even for its CTL-like fragment. However in [BG06], it has been shown that its existential and universal fragments are decidable even though the proof does not allow to obtain any tight bound on the computational complexity of the problem. Here, we show that the LTL fragment is decidable in polynomial space, a result not captured by these two works.

2.4 Expressive power and conciseness of the language

By definition, CLTL(IPC*)-models interpret variables but not propositional variables. However, it is not difficult to encode propositional variables by using atomic formulae of the form $x = 0$ where x is a new variable introduced for this purpose. The model-checking problems for CLTL(IPC^{++}) and CLTL(Z) are shown to be PSPACE-complete respectively in [Dem06] and in [DD07]. However, the proof for IPC^{++} uses an ω-regular property of the set of models that

does not hold when introducing constraints of the form $x < y$. The problem for CLTL(Z^c) is shown to be in EXPSPACE in [DD07] by translation into CLTL(Z) that increases exponentially the size of formulae with a binary encoding of the integers.

Let WIPC* (weak IPC*) denote the restriction of IPC* to constraints of the form either $x \sim y$, $x \sim d$ or $x \equiv_k c$ where $x, y \in$ VAR, $\sim \in \{<, =\}$, $d \in \mathbb{Z}$ and $k, c \in \mathbb{N}$. Though WIPC* is a fragment of IPC*, the logic CLTL(WIPC*) is as expressive as CLTL(IPC*).

Lemma 2. *For every $\phi \in$ CLTL(IPC*), there exists an equivalent formula $\psi \in$ CLTL(WIPC*).*

Proof. This a direct consequence of the facts below:

- IPC* admits quantifier-elimination.
- $x \equiv_k [c_1, c_2]$ is equivalent to

$$\bigvee_{c_1 \leq c' \leq c_2} x \equiv_k c$$

- $x \equiv_k y + [c_1, c_2]$ is equivalent to

$$\bigvee_{c_1 \leq c' \leq c_2} \Big(\bigvee_{\{\langle c'_1, c'_2 \rangle \in \{0,\ldots,k-1\}^2 \mid c'_1 + c'_2 \equiv_k c'\}} (x \equiv_k c'_1 \wedge y \equiv_k c'_2) \Big).$$

□

The size of ψ is exponential in the size of ϕ in the worst case (for an infinite amount of formulae ϕ). In spite of this exponential blow-up, we shall prove that both CLTL(WIPC*) and CLTL(IPC*) have PSPACE-complete model-checking problems.

Note also that adding constraints of the form $ax + by \equiv_k c$ with $a, b, c \in \mathbb{Z}$ in CLTL(IPC*) does not add expressiveness since we can translate such constraints in CLTL(IPC*). Let $S = \{\langle c_x, c_y \rangle \in \{0, \ldots, k-1\}^2 \mid c_x + c_y \equiv_k c\}$. We have the following logical equivalence:

$$ax + by \equiv_k c \Leftrightarrow \bigvee_{\langle c_x, c_y \rangle \in S} (ax \equiv_k c_x \wedge by \equiv_k c_y).$$

Then we can translate the constraints of the form $ax \equiv_k c_x$ into CLTL(IPC*) by solving a simple diophantine equation. The constraint $ax \equiv_k c_x$ reduces either to false if $gcd(a, k)$ does not divide c_x or to $x \equiv_{k'} c'$ with $k' \times gcd(a, k) = k$ for some c' that can be computed in polynomial-time in the respective sizes of a, k and c_x. The addition of such constraints may cause a gain of conciseness. However, because the sizes of k' and c' are bounded by the maximum of the sizes of a, k and c and k is a multiple of k', the forthcoming PSPACE upper bounds for CLTL(IPC*) problems can be also obtained when constraints of the form $\Sigma_i a_i x_i \equiv_k c$ are added.

3 Properties of the constraint language

In this section, we establish results about the constraint language underlying the logic CLTL(IPC*). We define a symbolic representation of the valuations in order to build automata that recognize symbolic representations of CLTL(IPC*)-models. Given a finite set X of IPC* constraints, typically the set of constraints occurring in a given CLTL(IPC*) formula, we introduce the following notations:

- K is the least common multiple of the integers $k_1, \ldots, k_n$ such that periodicity constraints with relations $\equiv_{k_1}, \ldots, \equiv_{k_n}$ occur in X.
- CONS is the finite set of constants d occurring in the constraints of X of the form $x \sim d$ (where $\sim \in \{<, =\}$).
- m is the minimal element of CONS and M is its maximal element.
- CONS$'$ denotes the set of constants $\{m, m+1, \ldots, M\}$.
- VAR is the finite set of variables occurring in X.

In the remaining, we consider that the above objects are always defined (possibly by adding dummy valid constraints in order to make the sets non-empty). In the sequel, we write $|\mathtt{O}|$ to denote the size of the finite object $\mathtt{O}$ for some reasonably succinct encoding (in particular binary encoding of integers). Observe that $|K|$ is in $\mathcal{O}(|k_1| + \cdots + |k_n|)$ and the cardinality of CONS is polynomial in the size of X. The cardinality of CONS$'$ is in $\mathcal{O}(2^{|M|})$ and each element of CONS$'$ can be encoded in binary representation with $\mathcal{O}(|M|)$ bits.

A maximally consistent set Y of Z^c constraints with respect to VAR and CONS is a set of Z^c constraints using only the variables from VAR and the constants from CONS such that there is a valuation v verifying $v \models_\star Y$ and for any proper extension Z of Y, there is no valuation v' verifying $v' \models_\star Z$. A valuation is abstracted by three disjoint finite sets of IPC* constraints similar to regions for timed automata.

Definition 1. *Given a finite set X of* IPC* *constraints, a symbolic valuation sv is a triple $\langle Y_1, Y_2, Y_3\rangle$ such that*

- *Y_1 is a maximally consistent set of* Z^c *constraints with respect to* VAR *and* CONS*.*
- *Y_2 is a set of constraints of the form $x = d$ with $x \in$ VAR and $d \in$ CONS$' \setminus$ CONS. Moreover, we impose that for every $x \in$ VAR, $(x = d) \in Y_2$ for some unique $d \in$ CONS$' \setminus$ CONS iff for every $d' \in$ CONS, $(x = d') \notin Y_1$ and $\{m < x, x < M\} \subseteq Y_1$. Note that, each $x \in$ VAR occurs at most once in Y_2.*
- *Y_3 is a set of constraints of the form $x \equiv_K c$ with $x \in$ VAR and $c \in \{0, \ldots, K-1\}$. Each $x \in$ VAR occurs exactly once in Y_3.*

A consequence of Definition 1 is that in a symbolic valuation $sv = \langle Y_1, Y_2, Y_3\rangle$, no constraint occurs in more than one set. Given an IPC* constraint ξ, we write $\xi \in sv$ instead of $\xi \in Y_1 \cup Y_2 \cup Y_3$. A symbolic valuation is satisfiable iff there is a valuation $v : \text{VAR} \to \mathbb{Z}$ such that $v \models_\star Y_1 \cup Y_2 \cup Y_3$.

Lemma 3. *Let X be a finite set of* IPC* *constraints and $sv = \langle Y_1, Y_2, Y_3\rangle$ be a triple composed of* IPC* *constraints such that Y_1 is a set of* Z^c *constraints built over* VAR *and* CONS*, Y_2 is a set of* Z^c *constraints of cardinality at most* |VAR| *built over* VAR *and* CONS$'\setminus$CONS*, Y_3 is a set of constraints of the form $x \equiv_K c$ of cardinality* |VAR|*. Checking whether sv is a satisfiable symbolic valuation can be done in polynomial-time in the sum of the respective size of X and sv.*

To prove this result, we have to check several things. Maximal consistency of Y_1 can be checked in polynomial-time by using developments from [Čer94, Lemma 5.5]. A set Y_1 of Z^c constraints is maximally consistent w.r.t. VAR and CONS iff the associated graph $G_{Y_1} = \langle \mathrm{VAR} \cup \mathrm{CONS}, \xrightarrow{=}, \xrightarrow{\leq}\rangle$ such that $n \xrightarrow{\sim} n' \overset{\text{def}}{\Leftrightarrow} (n \sim n') \in Y_1$ (where $\sim \in \{<, =\}$) satisfies the conditions below:

(MC1) For all n, n', either $n \xrightarrow{\sim} n'$ or $n' \xrightarrow{\sim} n$ for some $\sim \in \{<, =\}$.

(MC2) $\xrightarrow{=}$ is a congruence relation compatible with $\xrightarrow{\leq}$.

(MC3) There is no path $n_0 \xrightarrow{\sim_0} n_1 \xrightarrow{\sim_1} \ldots \xrightarrow{\sim_{\alpha-1}} n_\alpha$ such that $n_0 = n_\alpha$ and one of the symbols $\sim_0, \ldots, \sim_{\alpha-1}$ is equal to $<$ (no strict cycle).

(MC4) For all $d_1, d_2 \in$ CONS, $d_1 \sim d_2$ implies $d_1 \xrightarrow{\sim} d_2$.

(MC5) For all d_1, d_2 with $d_1 \leq d_2$, there is no path $n_0 \xrightarrow{\sim_0} n_1 \xrightarrow{\sim_1} \ldots \xrightarrow{\sim_{\alpha-1}} n_\alpha$ with $n_0 = d_1$ and $n_\alpha = d_2$ such that the number of occurrences of the symbol $<$ in $\sim_0, \cdots, \sim_{\alpha-1}$ is strictly greater than $d_2 - d_1$.

Note that this graph does not take into account the constraints in Y_2 and Y_3.

Checking that a variable occurs at most once in Y_2 or exactly once in Y_3 can be done in linear time. It is also easy to verify that the equality relations of Y_2 do not contradict the constraints in Y_1.

Finally, we need to check that the set of congruence relations Y_3 is compatible with the sets of constraints Y_1 and Y_2:

(MC6) (a) For all n, n' such that $n \xrightarrow{=} n'$, for every $c \in \{0, \ldots, K-1\}$, $n \equiv_K c \in Y_3$ iff $n' \equiv_K c \in Y_3$.
(b) For all $x = d$ in Y_2, the corresponding constraint $x \equiv_K c$ belonging to Y_3 is such that $c \equiv_K d$.

As illustrated by the following lemma, the symbolic representations of valuations contain the relevant information to evaluate constraints.

Lemma 4. *Let X be a finite set of* IPC* *constraints.*

(I) *For every valuation v : VAR $\to \mathbb{Z}$ there is a unique symbolic valuation $\langle Y_1, Y_2, Y_3\rangle$ built w.r.t. X and denoted by $sv(v)$ such that $v \models_\star Y_1 \cup Y_2 \cup Y_3$.*

(II) *For all valuations v, v' such that $sv(v) = sv(v')$ and for every $\xi \in X$, $v \models_\star \xi$ iff $v' \models_\star \xi$.*

Proof. (I) Given a symbolic valuation sv, let V_{sv} be the set of tuples $\langle d_1, \ldots, d_n\rangle$ in $\mathbb{Z}^{|\mathrm{VAR}|}$ (viewed as maps VAR $\to \mathbb{Z}$) such that $\langle d_1, \ldots, d_n\rangle \models sv$. It is easy to show that $\{V_{sv} : sv$ is a symbolic valuation built w.r.t. X and $V_{sv} \neq \emptyset\}$ is a

partition of $\mathbb{Z}^{|\mathrm{VAR}|}$.

(II) By structural induction on ξ. The proof is similar to the proof of [Dem06, Lemma 1]. □

Note that by Lemma 4, a symbolic valuation is an equivalence class of valuations. Given a symbolic valuation sv and a constraint ξ, we write $sv \models_{\mathrm{symb}} \xi \stackrel{\mathrm{def}}{\Leftrightarrow}$ for every valuation v such that $sv(v) = sv$, $v \models_\star \xi$.

Lemma 5. *Given a symbolic valuation sv built w.r.t. a set of constraints X (fixing* VAR*,* CONS *and K) and a constraint $\xi \in X$, checking whether $sv \models_{\mathrm{symb}} \xi$ is* PSPACE*-complete.*

Proof. PSPACE-hardness can be obtained by reducing QBF. The only constants used are 0 and 1. Each QBF formula $\phi = Q_1\ p_1\ Q_2\ p_2\ \ldots Q_n\ p_n\ \phi'$ with ϕ' a propositional formula in CNF built over the propositional variables $\{p_1, \ldots, p_n\}$ and $\{Q_1, \ldots, Q_n\} \subseteq \{\forall, \exists\}$ is translated via the map t as follows

- $t(\exists\ p_i\ \phi_i) = \exists\ x_i\ (x_i = 0 \vee x_i = 1) \wedge t(\phi_i)$,
- $t(\forall\ p_i\ \phi_i) = \forall\ x_i\ (x_i = 0 \vee x_i = 1) \Rightarrow t(\phi_i)$,
- t is homomorphic for Boolean connectives,
- $t(p_i) = (x_i = 1)$.

One can show that ϕ is QBF satisfiable iff for all symbolic valuations sv, $sv \models_{\mathrm{symb}} t(\phi)$ which is equivalent to check that an arbitrary symbolic valuation symbolically satisfies $t(\phi)$, since $t(\phi)$ has no free variable.

The proof for the upper bound is similar to the proof of the PSPACE upper bound for first-order model-checking [CM77]. We can define a function $\mathrm{MC}(sv, \xi', \xi)$ where

- ξ' is a subconstraint occurring in the IPC* constraint ξ,
- sv is a symbolic valuation over the syntactic resources of ξ' (VAR, CONS, K defined above),

which returns true iff $sv \models_{\mathrm{symb}} \xi'$. Observe that if a variable occurs in sv but is not free in ξ' then the satisfaction of $sv \models_{\mathrm{symb}} \xi'$ is independent of its value. The function MC is defined as a case analysis on the form of ξ'. For instance, $\mathrm{MC}(sv, \exists\ x\ \xi', \xi)$ returns true iff there is a satisfiable symbolic valuation sv' extending sv by addition of x-constraints, such that $\mathrm{MC}(sv', \xi', \xi)$ returns true. The symbolic valuations $sv' = \langle Y_1', Y_2', Y_3' \rangle$ and $sv = \langle Y_1, Y_2, Y_3 \rangle$ are related as follows:

- sv' is a satisfiable symbolic valuation over the free variables of ξ'. This can be checked in polynomial-time in $|\xi|$.
- $Y_1 \subseteq Y_1'$, $Y_2 \subseteq Y_2'$ and $Y_3 \subseteq Y_3'$.
- The only variable in sv' but not in sv is x.

Even if the number of symbolic valuations over the free variables of ξ' is exponential in $|\xi|$, it is possible to enumerate them in polynomial space in order to check the existence of some sv' verifying the above conditions. □

4 Satisfiable ω-sequences of symbolic valuations

Given a CLTL(IPC*) formula ϕ, we write IPC$^\star(\phi)$ to denote the set of IPC* constraints ξ such that some atomic formula of the form $\xi[x_1 \leftarrow \mathsf{X}^{i_1}x_{j_1}, \ldots, x_r \leftarrow \mathsf{X}^{i_r}x_{j_r}]$ occurs in ϕ. We associate to IPC$^\star(\phi)$ the objects relative to any finite set of IPC* constraints: the set VAR of variables and the set CONS of constants occurring in ϕ, and K the least common multiple of the integers k_i that occur in the congruence relations. This induces a unique set $\mathrm{CONS}' = \{m, \ldots, M\}$ where m is the minimal element of CONS and M the maximal element.

We define the X-length of ϕ, denoted by $|\phi|_{\mathsf{X}}$, as the maximal number i such that a term of the form $\mathsf{X}^i x$ occurs in ϕ. In the following, we assume that $\mathrm{VAR} = \{x_1, \ldots, x_s\}$ and $|\phi|_{\mathsf{X}} = l \geq 1$. We write Terms($\phi$) to denote the set of terms of the form $\mathsf{X}^\beta x_\alpha$ with $\beta \in \{0, \ldots, l\}$ and $\alpha \in \{1, \ldots, s\}$.

Let VAR$'$ be a set of fresh variables of cardinality |Terms(ϕ)|. For technical convenience, we need to introduce a bijection $f : \mathrm{Terms}(\phi) \to \mathrm{VAR}'$ such that f and f^{-1} can be computed in polynomial time. By extension, for every subformula ψ of ϕ, $f(\psi)$ is obtained from ψ by replacing each occurrence of $\mathsf{X}^\beta x_\alpha$ by $f(\mathsf{X}^\beta x_\alpha)$. The map f^{-1} is used in a similar fashion. A symbolic valuation with respect to ϕ is a symbolic valuation built over the set of variables VAR$'$, CONS and K.

We say that a pair $\langle\langle Y_1, Y_2, Y_3\rangle, \langle Y_1', Y_2', Y_3'\rangle\rangle$ of symbolic valuations with respect to ϕ is one-step consistent iff for every $j, j' \geq 1$, $x_i, x_{i'} \in \mathrm{VAR}$, $d \in \mathrm{CONS}'$ and $c \in \{0, \ldots, K-1\}$ we have

1. $f(\mathsf{X}^j x_i) \sim f(\mathsf{X}^{j'} x_{i'}) \in Y_1$ iff $f(\mathsf{X}^{j-1} x_i) \sim f(\mathsf{X}^{j'-1} x_{i'}) \in Y_1'$,
2. $f(\mathsf{X}^j x_i) \sim d \in Y_1 \cup Y_2$ iff $f(\mathsf{X}^{j-1} x_i) \sim d \in Y_1' \cup Y_2'$,
3. $f(\mathsf{X}^j x_i) \equiv_K c \in Y_3$ iff $f(\mathsf{X}^{j-1} x_i) \equiv_K c \in Y_3'$.

An ω-sequence ρ of satisfiable symbolic valuations w.r.t. ϕ is one-step consistent iff for every $j \in \mathbb{N}$, $\langle \rho(j), \rho(j+1)\rangle$ is one-step consistent, where $\rho(j)$ denotes the jth symbolic valuation of the sequence. We say that a symbolic valuation sequence ρ is satisfied by a CLTL(IPC*)-model σ iff for all $j \in \mathbb{N}$ and $\xi \in \rho(j)$, $\sigma, j \models f^{-1}(\xi)$. In order to simplify the future developments, we will write ρ_f (or sometimes $\rho_{f^{-1}}$) to denote the ω-sequence of IPC*-constraints obtained from ρ by substituting each occurrence of x by $f^{-1}(x)$ for every variable x used in ρ.

One-step consistent ω-sequences of symbolic valuations w.r.t. ϕ define abstractions of models for ϕ. We represent a one-step consistent sequence ρ by an infinite labeled structure $G_\rho = \langle (\mathrm{VAR} \cup \mathrm{CONS}') \times \mathbb{N}, \stackrel{=}{\to}, \stackrel{<}{\to}, mod\rangle$ where $mod : (\mathrm{VAR} \cup \mathrm{CONS}') \times \mathbb{N} \to \{0, \ldots, K-1\}$ and for all $\sim \in \{<, =\}$, $x, y \in \mathrm{VAR}$, $d', d_1, d_2 \in \mathrm{CONS}'$ and $i, j \in \mathbb{N}$ such that $|i - j| \leq l$ we have:

$\langle x, i\rangle \stackrel{\sim}{\to} \langle y, j\rangle$ iff either $i \leq j$ and $x \sim \mathsf{X}^{j-i} y \in \rho_f(i)$ or $i > j$ and $\mathsf{X}^{i-j} x \sim y \in \rho_f(j)$,

$\langle x, i\rangle \stackrel{=}{\to} \langle d, j\rangle$ iff $x = d \ \in \rho_f(i)$,

$\langle d, i\rangle \stackrel{=}{\to} \langle x, j\rangle$ iff $x = d \ \in \rho_f(j)$,

$\langle x, i\rangle \stackrel{<}{\to} \langle d, j\rangle$ iff there is $d' \sim d$ s.t. $x \sim' d' \ \in \rho_f(i)$ for some $\sim' \in \{<, =\}$ and either $\sim$ or $\sim'$ is equal to $<$,

$\langle d, i\rangle \overset{<}{\to} \langle x, j\rangle$ iff there is $d \sim d'$ s.t. $d' \sim' x \ \in \rho_f(j)$ for some $\sim' \in \{<, =\}$ and either $\sim$ or $\sim'$ is equal to $<$,

$\langle d_1, i\rangle \overset{\sim}{\to} \langle d_2, j\rangle$ iff $d_1 \sim d_2$,

$mod(\langle x, i\rangle) = c$ iff $x \equiv_K c \in \rho_f(i)$,

$mod(\langle d, i\rangle) = c$ iff $d \equiv_K c$.

By construction of G_ρ, the variables and constants are treated in a similar fashion. It is worth observing that G_ρ is well-defined because ρ is one-step consistent. Moreover, the construction ensures that the "local" representation of every $\rho(i)$ verifies the conditions (MC1)–(MC6) introduced Section 3.

We say that a vertex represents the constant $d \in \mathrm{CONS}'$ if it is of the form $\langle d, i\rangle$ for some $i \in \mathbb{N}$. The level of a node $n = \langle a, t\rangle$ in G_ρ is defined by $\mathrm{lev}(n) = t$. There is some redundancy in G_ρ for the nodes of the form $\langle d, i\rangle$ but this turn out to be helpful to establish tight relationships between ρ and G_ρ.

As an example, assume that $\mathrm{VAR} = \{y\}$, $\mathrm{CONS} = \{2, 4\}$, $K = 2$, $l = 1$, $\mathrm{VAR}' = \{x, x'\}$ ($f(y) = x$ and $f(\mathsf{X}y) = x'$) and $l = 1$. We consider the sequence $\rho = sv^0 \cdot (sv^1 \cdot sv^2)^\omega$ where

- $sv^0 = \langle Y_1^0, Y_2^0, Y_3^0\rangle$ with
  - $Y_1^0 = \{x = x, x' = x', x < x', 2 < x, x < 4, 2 < x', x' = 4\}$,
  - $Y_2^0 = \{x = 3\}$,
  - $Y_3^0 = \{x \equiv_2 1, x' \equiv_2 0\}$,
- $sv^1 = \langle Y_1^1, Y_2^1, Y_3^1\rangle$ with
  - $Y_1^1 = \{x = x, x' = x', x < x', 2 < x, x = 4, 2 < x', 4 < x'\}$,
  - $Y_2^1 = \emptyset$,
  - $Y_3^1 = \{x \equiv_2 0, x' \equiv_2 1\}$,
- $sv^2 = \langle Y_1^2, Y_2^2, Y_3^2\rangle$ with
  - $Y_1^2 = \{x = x, x' = x', x < x', 2 < x, 4 < x, 2 < x', 4 < x'\}$,
  - $Y_2^2 = \emptyset$,
  - $Y_3^2 = \{x \equiv_2 1, x' \equiv_2 1\}$.

The graph G_ρ is presented in Figure 2. In order to simplify the representation, closure by transitivity for $\overset{<}{\to}$ and the fact that $\overset{=}{\to}$ is a congruence are omitted. The function *mod* is directly encoded in the node label.

A path in G_ρ is a sequence (possible infinite) of the form $n_0 \overset{\sim_0}{\to} n_1 \overset{\sim_1}{\to} n_2 \overset{\sim_2}{\to} \ldots$ (each $\sim_i$ belongs to $\{<, =\}$). A finite path $w = n_0 \overset{\sim_0}{\to} n_1 \overset{\sim_1}{\to} n_2 \overset{\sim_2}{\to} \ldots \overset{\sim_{\alpha-1}}{\longrightarrow} n_\alpha$ such that $n_0 = n_\alpha$ is called a cycle. For any finite path $w = n_0 \overset{\sim_0}{\to} n_1 \overset{\sim_1}{\to} n_2 \overset{\sim_2}{\to} \ldots \overset{\sim_{\alpha-1}}{\longrightarrow} n_\alpha$, its strict length $slen(w)$ is the number of indices $i \in \{0, \ldots, \alpha - 1\}$ such that $\sim_i$ is equal to $<$. We say that w is strict if $slen(w) > 0$. The strict length between two nodes n_1 and n_2, written $slen(n_1, n_2)$, is the least upper bound (possibly equal to ω) of the strict lengths of finite paths between n_1 and n_2. By convention, if there is no path between from n_1 to n_2, $slen(n_1, n_2)$ takes the value $-\infty$. For example, in Figure 2 $slen(\langle 2, 2\rangle, \langle y, 3\rangle) = 4$ (see the transitions in boldface).

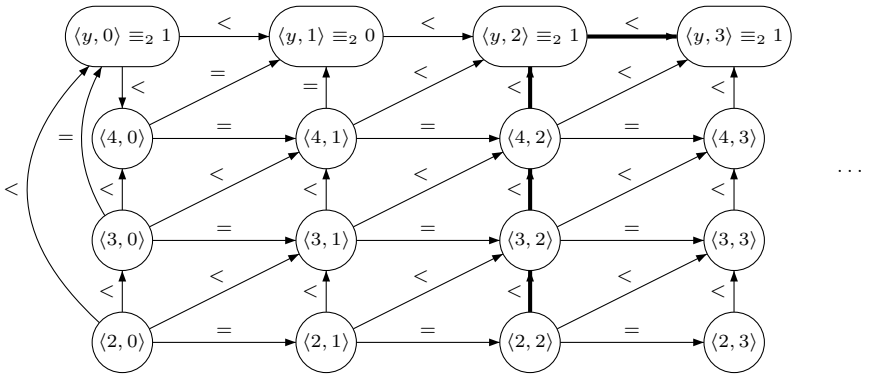

Fig. 2. A graph G_ρ

The one-step consistency of ρ implies global constraints on its graph representation that already hold true locally. By a global constraint, we mean a constraint on the whole graph and not only on the local representation of a single symbolic valuation or on two successive satisfiable symbolic valuations.

Lemma 6. *Let ρ be a one-step consistent symbolic valuation sequence. The following properties hold for G_ρ.*

(I) *G_ρ has no strict cycle.*
(II) *If there is a finite path w starting at $\langle d, i\rangle$ and ending at the node n of level j, then: if w is strict then $\langle d, j\rangle \xrightarrow{<} n$, otherwise $\langle d, j\rangle \xrightarrow{=} n$.*
(III) *If there is a finite path w starting at the node n of level j and ending at $\langle d, i\rangle$, then: if w is strict then $n \xrightarrow{<} \langle d, j\rangle$, otherwise $n \xrightarrow{=} \langle d, j\rangle$.*
(IV) *For every pair of nodes n, n' in G_ρ such that $slen(n, n') = 0$, we have $mod(n) = mod(n')$.*

Proof. (I) We show that for every path $w = n_0 \overset{\sim_0}{\to} n_1 \overset{\sim_1}{\to} n_2 \overset{\sim_2}{\to} \ldots \xrightarrow{\sim_{\alpha-1}} n_\alpha$, $n_0 = n_\alpha$ implies $slen(w) = 0$. The proof is by induction on α. The base case with either $\alpha = 1$ or $\alpha = 2$ is by an easy verification since the subgraph corresponding to the symbolic valuation $\rho(min\{\mathrm{lev}(n_i) : 1 \leq i \leq \alpha\})$ contains all the nodes of the path and satisfies (MC3). In the induction step, suppose that w is a cycle with $\alpha > 2$ and let n_β be a node of the path with the greatest level. Without any loss of generality, we can assume that $0 < \beta < \alpha$ since one can choose the first node of the cycle. As $\mathrm{lev}(n_\beta)$ is maximal, $|\mathrm{lev}(n_{\beta-1}) - \mathrm{lev}(n_{\beta+1})| \leq l$ and so the subgraph corresponding to $\rho(min\{\mathrm{lev}(n_{\beta-1}), \mathrm{lev}(n_{\beta+1})\})$ contains the nodes $n_{\beta-1}, n_\beta$ and $n_{\beta+1}$. Since this subgraph satisfies (MC2), this implies that $n_{\beta-1} \xrightarrow{<} n_{\beta+1}$ iff $n_{\beta-1} \xrightarrow{<} n_\beta$ or $n_\beta \xrightarrow{<} n_{\beta+1}$. As a consequence, $slen(w) = 0$ iff $slen(w') = 0$ where the path w' is obtained from w by replacing the subpath from $n_{\beta-1}$ to $n_{\beta+1}$ by the edge $n_{\beta-1} \xrightarrow{\sim} n_{\beta+1}$. By induction hypothesis, $slen(w') = 0$ and therefore $slen(w) = 0$.

(II) We show that for every path $w = n_0 \stackrel{\sim_0}{\to} n_1 \stackrel{\sim_1}{\to} n_2 \stackrel{\sim_2}{\to} \ldots \stackrel{\sim_{\alpha-1}}{\longrightarrow} n_\alpha$ such that $n_0 = \langle d, i\rangle$, we have $\langle d, \mathrm{lev}(n_\alpha)\rangle \stackrel{\sim}{\to} n_\alpha$ with $\sim$ equal to $<$ if w is strict and to $=$ otherwise. The proof is by induction on α. The base case with $\alpha = 1$ is obvious. Let us consider a path $w = n_0 \stackrel{\sim_0}{\to} n_1 \stackrel{\sim_1}{\to} n_2 \stackrel{\sim_2}{\to} \ldots \stackrel{\sim_{\alpha-1}}{\longrightarrow} n_\alpha$ with $n_0 = \langle d, i\rangle$ and $\alpha > 1$. Suppose that $\langle d, \mathrm{lev}(n_{\alpha-1})\rangle \stackrel{\sim}{\to} n_{\alpha-1}$ with $\sim$ equals to $<$ if $n_0 \stackrel{\sim_0}{\to} n_1 \stackrel{\sim_1}{\to} n_2 \stackrel{\sim_2}{\to} \ldots \stackrel{\sim_{\alpha-2}}{\longrightarrow} n_{\alpha-1}$ is strict and $\sim$ equals to $=$ otherwise. We only treat the case $\mathrm{lev}(n_{\alpha-1}) \leq \mathrm{lev}(n_\alpha)$, the other case is similar. Since $\mathrm{lev}(n_\alpha) - \mathrm{lev}(n_{\alpha-1}) \leq l$, the nodes $n_{\alpha-1}$ and n_α belong to the subgraph corresponding to $\rho(\mathrm{lev}(n_{\alpha-1}))$. This subgraph satisfies (MC2) and so $\langle d, \mathrm{lev}(n_{\alpha-1})\rangle \stackrel{\sim'}{\to} n_\alpha$ with $\sim'$ equal to $<$ if either $\langle d, \mathrm{lev}(n_{\alpha-1})\rangle \stackrel{<}{\to} n_{\alpha-1}$ or $n_{\alpha-1} \stackrel{<}{\to} n_\alpha$. Hence, by using induction hypothesis we obtain that $\langle d, \mathrm{lev}(n_{\alpha-1})\rangle \stackrel{\sim'}{\to} n_\alpha$ and $\sim'$ is equal to $<$ if w is strict. Since $\langle d, \mathrm{lev}(n_{\alpha-1})\rangle \stackrel{=}{\to} \langle d, \mathrm{lev}(n_\alpha)\rangle$ and using the property (MC2), we get $\langle d, \mathrm{lev}(n_\alpha)\rangle \stackrel{\sim'}{\to} n_\alpha$ and $\sim'$ equals $<$ if w is strict (and $=$ otherwise).

(III) Similar to (II).

(IV) Suppose that two nodes n and n' are such that $slen(n, n') = 0$. So there is a path, $w = n \stackrel{=}{\to} n_0 \stackrel{=}{\to} \ldots \stackrel{=}{\to} n_{\alpha-1} \stackrel{=}{\to} n_\alpha$ with $n_\alpha = n'$. We proceed by induction on α. If $\alpha \leq 1$ then the local representation of $\rho(min\{\mathrm{lev}(n_0), \mathrm{lev}(n_\alpha))\}$ contains both n_0 and n_α and we can conclude using (MC6). Otherwise, suppose that $\alpha > 1$ and $mod(n_0) = mod(n_{\alpha-1})$. Since $n_{\alpha-1} \stackrel{=}{\to} n_\alpha$ the local representation of $\rho(min\{\mathrm{lev}(n_{\alpha-1}), \mathrm{lev}(n_\alpha))\}$ contains both $n_{\alpha-1}$ and n_α. Since, this symbolic valuation satisfies (MC6), we have $mod(n_\alpha) = mod(n_{\alpha-1}) = mod(n_0)$. □

Corollary 1. *Let ρ be a one-step consistent sequence of symbolic valuations and G_ρ its graph representation. Then, for all nodes $\langle d_1, i\rangle$ and $\langle d_2, j\rangle$ in G_ρ representing constants such that $d_1 \leq d_2$, $slen(\langle d_1, i\rangle, \langle d_2, j\rangle) = d_2 - d_1$.*

Proof. Let $\langle d_1, i\rangle$ and $\langle d_2, j\rangle$ be vertices of G_ρ representing respectively the constants d_1 and d_2. Without any loss of generality, we can assume that $i \leq j$ (the case $j > i$ has a similar treatment). Obviously, $slen(\langle d_1, i\rangle, \langle d_2, j\rangle) \geq d_2 - d_1$ as witnessed by the path below of strict length $d_2 - d_1$:

$$\langle d_1, i\rangle \stackrel{=}{\to} \langle d_1, i{+}1\rangle \stackrel{=}{\to} \langle d_1, i{+}2\rangle \cdots \langle d_1, j\rangle \stackrel{<}{\to} \langle d_1{+}1, j\rangle \stackrel{<}{\to} \langle d_1{+}2, j\rangle \cdots \stackrel{<}{\to} \langle d_2, j\rangle.$$

Now suppose that there is a path w between $\langle d_1, i\rangle$ and $\langle d_2, j\rangle$ such that $slen(w) > d_2 - d_1$. Consider the restriction of the transitive closure of $\stackrel{=}{\to}$ to the nodes appearing in w. This relation is an equivalence relation having exactly $slen(w)+1$ equivalence classes. Let $X_0, \ldots, X_{slen(w)}$ be an enumeration of these equivalence classes. As a consequence of Lemma 6(II,III), every node n in w of level j is such that $\langle d_1, j\rangle \stackrel{\sim}{\to} n \stackrel{\sim'}{\to} \langle d_2, j\rangle$ for some $\sim, \sim' \in \{<, =\}$. By definition of CONS′ (which contains all the integers of the interval $[m, M]$) and by maximal consistency of the local representations, for every $i \in \{0, \ldots, slen(w)\}$, there is $d'_i \in$ CONS′ such that $d_1 \leq d'_i \leq d_2$ and every node n of level j in X_i has an outgoing edge $n \stackrel{=}{\to} \langle d'_i, j\rangle$. Moreover the constants $d'_0, \ldots, d'_{slen(w)}$ should be

mutually distinct since all the X_i represents distinct equivalence classes. This leads to a contradiction since the cardinality of $\{d_1, \ldots, d_2\}$ is $(d_2 - d_1) + 1$. $\square$

So far, we have stated properties about the graph G_ρ. Below, we establish simple conditions on G_ρ equivalent to the existence of a CLTL(IPC*) model satisfying ρ. An edge-respecting labeling for G_ρ is a map $lab : (\mathrm{VAR} \cup \mathrm{CONS}') \times \mathbb{N} \to \mathbb{Z}$ such that

- for all nodes n_1, n_2 and $\sim \in \{<, =\}$, we have $n_1 \overset{\sim}{\to} n_2$ implies $lab(n_1) \sim lab(n_2)$,
- for every node n, $lab(n) \equiv_K mod(n)$.

Additionally, lab is said to be strict if for every $\langle d, i\rangle$ in G_ρ, $lab(\langle d, i\rangle) = d$.

Lemma 7. *A one-step consistent sequence of symbolic valuations ρ has a model iff G_ρ has an edge-respecting labeling.*

Proof. Let σ be a model for ρ and $lab : (\mathrm{VAR} \cup \mathrm{CONS}') \times \mathbb{N} \to \mathbb{Z}$ be the map defined as follows:

$$lab(\langle x, i\rangle) = \sigma(i, x); \quad lab(\langle d, i\rangle) = d \quad \text{for all } x \in \mathrm{VAR}, d \in \mathrm{CONS}' \text{ and } i \in \mathbb{N}.$$

It is not difficult to show that lab is a *strict* edge-respecting labeling for G_ρ. For instance, we have implications between the propositions below ($x, y \in \mathrm{VAR}$, $i, j \in \mathbb{N}$, $\sim \in \{<, =\}$):

- $\langle x, i\rangle \overset{\sim}{\to} \langle y, j\rangle$ and $i \leq j$,
- $x \sim \mathsf{X}^{j-i} y \in \rho_f(i)$ (by definition of G_ρ),
- $f(x \sim \mathsf{X}^{j-i} y) \in \rho(i)$ (by definition of ρ_f),
- $\sigma, i \models f^{-1}(f(x \sim \mathsf{X}^{j-i} y))$ (since σ is a model for ρ),
- $\sigma, i \models x \sim \mathsf{X}^{j-i} y$ (f is a bijection),
- $\sigma(i, x) \sim \sigma(j, y)$ (by definition of $\models$),
- $lab(\langle x, i\rangle) \sim lab(\langle y, j\rangle)$ (by definition of lab).

Hence, $\langle x, i\rangle \overset{\sim}{\to} \langle y, j\rangle$ and $i \leq j$ implies $lab(\langle x, i\rangle) \sim lab(\langle y, j\rangle)$. Satisfaction of periodicity constraints is based on the same development ($x \in \mathrm{VAR}$, $c \in \{0, \ldots, K-1\}$):

- $mod(\langle x, i\rangle) = c$,
- $x \equiv_K c \in \rho_f(i)$ (by definition of G_ρ),
- $\sigma, i \models x \equiv_K c$ (arguments as above),
- $\sigma(i, x) \equiv_K c$ (by definition of $\models$),
- $lab(\langle x, i\rangle) \equiv_K c$ (by definition of lab).

Conversely, let lab be an edge-respecting labeling of G_ρ. First, we build from lab a strict edge-respecting labeling lab' of G_ρ. The values greater than M are divided in consecutive blocks of K consecutive values in such a way that if $lab(n) - lab(\langle M, \mathrm{lev}(n)\rangle) = \beta > 0$ then $lab'(n)$ takes its value in the βth block.

$$\overbrace{M}^{\text{block } 0}\ \overbrace{M+1\cdots M+K}^{\text{block } 1}\ \overbrace{M+K+1\cdots M+2K}^{\text{block } 2}\ldots\overbrace{M+(\gamma-1)K+1\cdots M+\gamma K}^{\text{block } \gamma}\ldots$$

Then the constraint $mod(n)$ insures the unicity of $lab'(n)$ such that $lab'(n) \equiv_K mod(n)$. A similar division is performed for the values smaller than m.

- For every $n = \langle x, i\rangle$ such that $\langle x, i\rangle \xrightarrow{\leq} \langle m, i\rangle$, then $lab'(\langle x, i\rangle) \stackrel{\text{def}}{=} \alpha$ with
  - $\alpha \equiv_K mod(\langle x, i\rangle)$,
  - $m - (lab(\langle m, i\rangle) - lab(\langle x, i\rangle)) \times K \leq \alpha \leq m - ((lab(\langle m, i\rangle) - lab(\langle x, i\rangle) - 1) \times K - 1$.
- For every $\langle x, i\rangle$ such that $\langle M, i\rangle \xrightarrow{\leq} \langle x, i\rangle$, then $lab'(\langle x, i\rangle) \stackrel{\text{def}}{=} \alpha$ with
  - $\alpha \equiv_K mod(\langle x, i\rangle)$,
  - $M + ((lab(\langle x, i\rangle) - lab(\langle M, i\rangle) - 1) \times K + 1 \leq \alpha \leq M + (lab(\langle x, i\rangle) - lab(\langle M, i\rangle)) \times K$.

  In both above cases, α is unique since it belongs to an interval of length K with a periodicity constraint that forces a unique value in this interval.
- For every $\langle x, i\rangle$ such that $\langle x, i\rangle \xrightarrow{=} \langle d, i\rangle$ for some $d \in \text{CONS}'$, $lab'(\langle x, i\rangle) = d$.
- For every $\langle d, i\rangle$, $lab'(\langle d, i\rangle) = d$.

lab' is well-defined because ρ is a sequence of satisfiable symbolic valuations with respect to ϕ. Moreover, lab' is a strict edge-respecting labeling. By way of example, suppose that $n \xrightarrow{\leq} n'$, $\text{lev}(n) \leq \text{lev}(n')$, $\langle M, \text{lev}(n)\rangle \xrightarrow{\leq} n$ and $\langle M, \text{lev}(n')\rangle \xrightarrow{\leq} n'$. We have $lab(\langle M, \text{lev}(n)\rangle) < lab(n) < lab(n')$ because lab is edge-respecting and so, since the values of the $(lab(n) - lab(\langle M, \text{lev}(n)\rangle)$th block after M are greater than the values of $(lab(n') - lab(\langle M, \text{lev}(n)\rangle)$th block, $lab'(n) < lab'(n')$.

Now, we show that the model σ defined by $\sigma(i, x) = lab'(\langle x, i\rangle)$ for all $x \in \text{VAR}$ and $i \in \mathbb{N}$ satisfies ρ. By way of example, we show that $\mathsf{X}^j x \sim \mathsf{X}^k y \in \rho_f(i)$ and $j \leq k$ implies $\sigma, i \models \mathsf{X}^j x \sim \mathsf{X}^k y$. We have implications between the propositions below:

- $\mathsf{X}^j x \sim \mathsf{X}^k y \in \rho_f(i)$ and $j \leq k$,
- $\langle x, i + j\rangle \xrightarrow{\sim} \langle y, i + k\rangle$ in G_ρ (by definition of G_ρ),
- $lab(\langle x, i + j\rangle) \sim lab'(\langle y, i + k\rangle)$ (lab' is edge-respecting),
- $\sigma(i + j)(x) \sim \sigma(i + k)(y)$ (by definition of σ),
- $\sigma, i \models \mathsf{X}^j x \sim \mathsf{X}^k y$ (by definition of $\models$).

□

Lemmas 7 states correspondences between ρ and its graphical representation G_ρ. We now define a more abstract characterization of the one-step consistent sequences admitting a model.

Lemma 8. *Let ρ be a one-step consistent sequence. The graph G_ρ has an edge-respecting labeling iff for all nodes n_1, n_2 in G_ρ, $slen(n_1, n_2) < \omega$.*

Note that, by construction of G_ρ, for all nodes $\langle d_1, i\rangle$ and $\langle d_2, j\rangle$ representing constants such that $d_1 \leq d_2$ we have $slen(\langle d_1, i\rangle, \langle d_2, j\rangle) = d_2 - d_1$ (see Corollary 1). That is why it suffices to consider nodes n_1 and n_2 that are not both constants.

Proof. If G_ρ has an edge-respecting labeling lab, then one can easily show that for all nodes n_1, n_2 in G_ρ $slen(n_1, n_2) \leq lab(n_2) - lab(n_1)$.

Conversely, if for all nodes n_1, n_2 in G_ρ, $slen(n_1, n_2) < \omega$, we define the following map $lab : (\mathrm{VAR} \cup \mathrm{CONS}') \times \mathbb{N} \to \mathbb{Z}$:

- $lab(\langle d, i\rangle) \stackrel{\text{def}}{=} d$.
- If $\langle x, i\rangle \xrightarrow{=} \langle d, i\rangle$ then $lab(\langle x, i\rangle) \stackrel{\text{def}}{=} d$.
- Otherwise,
  - If $\langle x, i\rangle \xrightarrow{<} \langle m, i\rangle$ then $lab(\langle x, i\rangle) \stackrel{\text{def}}{=} \alpha$ with
    1. $\alpha \equiv_K mod(\langle x, i\rangle)$.
    2. $m - slen(\langle x, i\rangle, \langle m, i\rangle) \times K \leq \alpha \leq m - (slen(\langle x, i\rangle, \langle m, i\rangle) - 1) \times K - 1$.
  - If $\langle M, i\rangle \xrightarrow{<} \langle x, i\rangle$ then $lab(\langle x, i\rangle) \stackrel{\text{def}}{=} \alpha$ with
    1. $\alpha \equiv_K mod(\langle x, i\rangle)$.
    2. $M + (slen(\langle M, i\rangle, \langle x, i\rangle) - 1) \times K + 1 \leq \alpha \leq M + slen(\langle M, i\rangle, \langle x, i\rangle) \times K$.

Similarly to the proof of Lemma 7, each α is uniquely defined since it belongs to an interval of length K and the map mod forces a unique value in this interval.

We now show that lab is a strict edge-respecting labeling of G_ρ. If the labeling is not edge-respecting, then one of the following cases arises:

- Suppose $n \xrightarrow{=} n'$. We treat the case $\mathrm{lev}(n) < \mathrm{lev}(n')$ (the symmetrical case has an analogous treatment).

  Case 1: $\langle M, \mathrm{lev}(n)\rangle \xrightarrow{<} n$ and $\langle M, \mathrm{lev}(n')\rangle \xrightarrow{<} n'$.
  Since $n \xrightarrow{=} n'$ and $\langle M, \mathrm{lev}(n)\rangle \xrightarrow{=} \langle M, \mathrm{lev}(n')\rangle$, we have

  $$slen(\langle M, \mathrm{lev}(n)\rangle, n) = slen(\langle M, \mathrm{lev}(n')\rangle, n').$$

  Hence $lab(n)$ and $lab(n')$ belong to the same block of size K after M. Moreover, $\rho(\mathrm{lev}(n))$ is maximally consistent which entails $mod(n) = mod(n')$ (by (MC6)) and $lab(n) = lab(n')$.

  Case 2: $n \xrightarrow{<} \langle m, \mathrm{lev}(n)\rangle$ and $n' \xrightarrow{<} \langle m, \mathrm{lev}(n')\rangle$.
  Similar to Case 1.

  Case 3: $\langle M, \mathrm{lev}(n)\rangle \xrightarrow{<} n$ and $n' \xrightsquigarrow{} \langle M, \mathrm{lev}(n')\rangle$.
  Since $\rho(\mathrm{lev}(n))$ is maximally consistent, by (MC2) we have $n \xrightsquigarrow{} \langle M, \mathrm{lev}(n')\rangle$.
  So we obtain the path

  $$\langle M, \mathrm{lev}(n)\rangle \xrightarrow{<} n \xrightsquigarrow{} \langle M, \mathrm{lev}(n')\rangle \xrightarrow{=} \langle M, \mathrm{lev}(n)\rangle,$$

  which leads to a contradiction using (MC3).

  Case 4: $n \xrightarrow{<} \langle m, \mathrm{lev}(n)\rangle$ and $\langle m, \mathrm{lev}(n')\rangle \xrightsquigarrow{} n'$.

Similar to Case 3.

Case 5: $n \xrightarrow{=} \langle d, \mathrm{lev}(n)\rangle$ and $n' \xrightarrow{=} \langle d', \mathrm{lev}(n')\rangle$ with $d < d'$.
Since $\rho(\mathrm{lev}(n))$ is maximally consistent, by (MC2) and (MC3) we have $d = d'$ which leads to a contradiction.

– The case $n \xrightarrow{<} n'$ can be done in a similar fashion.

□

So we have a characterization of the set of sequences having a model but what we really want is to recognize them with automata. The main difficulty rests on the fact that the set of satisfiable one-step consistent ω-sequences of satisfiable symbolic valuations is not ω-regular, as shown in [DD07] for the fragment CLTL(Z). However, we can define an ω-regular condition such that every one-step consistent ultimately periodic sequence ρ is satisfiable iff G_ρ satisfies this condition. An infinite word is ultimately periodic if it is of the form $\tau \cdot \delta^\omega$ for some finite words τ and δ. We will see in the following section that this approximation condition is enough for our purpose since satisfiable CLTL(IPC*) formulas always have a ultimately periodic (symbolic) model. Let ρ be a one-step consistent symbolic valuation sequence and G_ρ its graphical representation. An infinite forward (resp. backward) path in G_ρ is defined as a sequence $w : \mathbb{N} \to (\mathrm{VAR} \cup \mathrm{CONS}') \times \mathbb{N}$ such that:

– for every $i \in \mathbb{N}$, we have $w(i) \xrightarrow{\sim} w(i+1)$ (resp. $w(i+1) \xrightarrow{\sim} w(i)$) in G_ρ,
– for every $i \in \mathbb{N}$, we have $\mathrm{lev}(w(i)) < \mathrm{lev}(w(i+1))$.

The path w is infinitely often strict iff for every $i \geq 0$, there is $j \geq i$ such that $w(j) \xrightarrow{<} w(j+1)$ (resp. $w(j+1) \xrightarrow{<} w(j)$).

Definition 2. *A graph G_ρ satisfies the condition ($\mathcal{C}$) iff there* **do not** *exist vertices n_1 and n_2 in G_ρ with $|\mathrm{lev}(n_1) - \mathrm{lev}(n_2)| \leq l$ satisfying*

(AP1) *there is an infinite forward path w_{for} from n_1,*
(AP2) *there is an infinite backward path w_{back} from n_2,*
(AP3) *either w_{for} or w_{back} is infinitely often strict, and*
(AP4) *for all $i, j \in \mathbb{N}$, whenever $|\mathrm{lev}(w_{\mathrm{for}}(i)) - \mathrm{lev}(w_{\mathrm{back}}(j))| \leq l$, $w_{\mathrm{for}}(i) \xrightarrow{<} w_{\mathrm{back}}(j)$ in G_ρ.*

A graph representation of some ρ not satisfying ($\mathcal{C}$) is presented in Figure 3 where n_1 is a constant node.

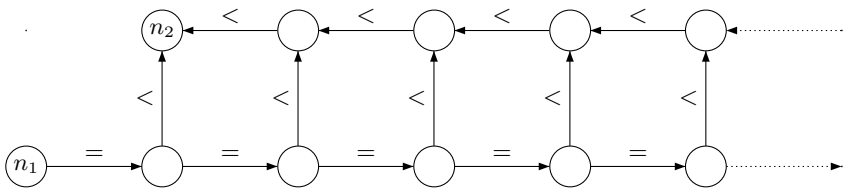

Fig. 3. G_ρ does not satisfy ($\mathcal{C}$)

If ρ admits a model, then necessarily G_ρ satisfies $(\mathcal{C})$. Indeed, if G_ρ does not satisfy $(\mathcal{C})$, then $slen(n_1, n_2) = \omega$ which entails that ρ has no model by Lemma 7 and 8. The converse does not hold in general. However, when ρ is ultimately periodic, the condition $(\mathcal{C})$ is sufficient.

Lemma 9. *Let ρ be a one-step consistent ω-sequence of satisfiable symbolic valuations which is ultimately periodic. Then ρ admits a model iff G_ρ satisfies $(\mathcal{C})$.*

Thanks to the way G_ρ is built from ρ, $(\mathcal{C})$ does not explicitly refer to the constants in CONS and the constraints of the form $x \equiv_K c$. Hence, Lemma 9 can be proved as [DD07, Lemma 6.2]: the map *mod* in G_ρ is ignored and a uniform treatment for all nodes in $(\mathrm{VAR} \cup \mathrm{CONS}') \times \mathbb{N}$ is provided.

Let $\rho = \tau \cdot \delta^\omega$ be an ultimately periodic one-step consistent ω-sequence. If ρ admits a model then by Lemma 8 it satisfies the condition $(\mathcal{C})$. Conversely, if ρ has no model then by Lemma 8 there exist two vertices n_1 and n_2 such that $slen(n_1, n_2) = \omega$. Intuitively, the proof uses this property to claim the existence between these two nodes of a finite path w long enough so that two paths w_{for} and w_{back} satisfying the conditions (AP1)–(AP4) can be deduced. The construction of w_{for} and w_{back} from w uses the periodicity of ρ by repeating infinitely finite subpaths and can be done smoothly by using the properties established in this section (see e.g. Lemma 6). This witnesses that G_ρ does not satisfy $(\mathcal{C})$.

As the proof is not essentially different from [DD07, Lemma 6.2] modulo slight changes mentioned above, we omit it here (see details in [Gas07]).

5 Büchi automata and PSPACE upper bound

Based on the previous results and following the approach in [VW94], we show that given a CLTL(IPC*) formula ϕ, one can build a standard Büchi automaton $\mathcal{A}_\phi$ such that ϕ is CLTL(IPC*) satisfiable iff $\mathrm{L}(\mathcal{A}_\phi)$ is non-empty. Moreover, we establish that nonemptiness of $\mathrm{L}(\mathcal{A}_\phi)$ can be checked in polynomial space in the size of ϕ (denoted by $|\phi|$). The automaton $\mathcal{A}_\phi$ is precisely the intersection of three Büchi automata and its construction can be done quite smoothly thanks to the previous results. In the following, VAR, VAR$'$, CONS and CONS$'$ are the sets of variables and constants associated to ϕ as defined in Section 4. Moreover, K, m and M are constants with their usual meaning and we use the map $f : \mathrm{Terms}(\phi) \to \mathrm{VAR}'$ as previously.

Unlike LTL, the language recognized by the Büchi automaton $\mathcal{A}_\phi$ is not a set of models but rather a set of symbolic models. The alphabet Σ of this automaton is the set of symbolic valuations w.r.t. ϕ. As a consequence, a symbolic model for ϕ is an ω-sequence $\rho : \mathbb{N} \to \Sigma$. We naturally extend the symbolic satisfaction to sequences. The relation $\models'$ is defined as $\models$ except at the atomic level: $\rho, i \models' \xi \stackrel{\text{def}}{\Leftrightarrow} \rho(i) \models_{\text{symb}} f(\xi)$ where $\models_{\text{symb}}$ is the satisfaction relation between symbolic valuations and constraints (see Section 3).

By Lemma 5 and using standard techniques for LTL [VW94], checking whether there is a symbolic model ρ satisfying $\rho \models' \phi$ can be done in PSPACE (see more details below). Because every model for ϕ generates a unique symbolic model for ϕ (consequence of Lemma 4), we obtain the result below.

Lemma 10. *A* CLTL(IPC*) *formula ϕ is satisfiable iff there is a one-step consistent symbolic valuation ρ such that $\rho \models' \phi$ and ρ has a model.*

Proof. Let σ be a model that satisfies ϕ. Consider the symbolic valuation sequence ρ defined by: $\rho(i) = sv(v_i)$ where for every $i \in \mathbb{N}$, v_i is the valuation such that $v_i(f(\mathsf{X}^j x)) = \sigma(i+j)(x)$. By construction, we have $\sigma \models \rho$. Using Lemma 4(II) we can show that for every v such that $sv(v) = \rho(i)$ we have $\sigma(i) \models \xi$ iff $v \models \xi$ for every atomic subformula ξ of ϕ. By definition of the symbolic satisfaction relation, this implies that if $\sigma \models \xi$ then $\rho \models' \xi$. Consequently, $\rho \models' \phi$ (induction on the structure of ϕ).

Conversely, suppose that $\rho \models' \phi$ and $\sigma \models \rho$ for some σ and ρ. Since for every $i \in \mathbb{N}$ we have $\sigma, i \models \rho(i)$, $\rho(i)$ is the symbolic valuation corresponding to the valuation v_i such that $v_i(f(\mathsf{X}^j x)) = \sigma(i+j)(x)$. By definition of $\models'$, this implies that for every atomic subformula ξ of ϕ, if $\rho, i \models' \xi$ then $\sigma, i \models \xi$. Thus, we can show that $\rho \models \phi$ and $\sigma \models \rho$ imply $\sigma \models \phi$. □

The automaton $\mathcal{A}_\phi$ is formally defined as the intersection $\mathcal{A}_{\mathrm{LTL}} \cap \mathcal{A}_{1\mathrm{cons}} \cap \mathcal{A}_{\mathcal{C}}$ of Büchi automata where

- $\mathrm{L}(\mathcal{A}_{\mathrm{LTL}})$ is the set of symbolic models satisfying ϕ,
- $\mathrm{L}(\mathcal{A}_{1\mathrm{cons}})$ is the set of one-step consistent sequences of symbolic valuations,
- $\mathrm{L}(\mathcal{A}_{\mathcal{C}})$ is the set of sequences of symbolic valuations verifying $(\mathcal{C})$.

We briefly explain below how these different automata are built. All of them are built over the alphabet Σ which is of exponential size in $|\phi|$. The automaton $\mathcal{A}_{\mathrm{LTL}}$ is obtained from [VW94] with a difference for atomic formulae. We define $cl(\phi)$ the closure of ϕ as usual, and an atom of ϕ is a maximally consistent subset of $cl(\phi)$. We define $\mathcal{A}_{\mathrm{LTL}} = (Q, Q_0, \delta, F)$ as the generalized Büchi automaton below:

- Q is the set of atoms of ϕ and $Q_0 = \{X \in Q : \phi \in X\}$,
- $X \xrightarrow{sv} Y$ is in δ iff
  (atomic constraints) for every atomic formula ξ in X, $sv \models_{\mathrm{symb}} f(\xi)$,
  (one step) for every $\mathsf{X}\psi \in cl(\phi)$, $\mathsf{X}\psi \in X$ iff $\psi \in Y$,
- let $\{\phi_1 \mathsf{U} \psi_1, \ldots, \phi_r \mathsf{U} \psi_r\}$ be the set of until formulas in $cl(\phi)$. We define F as the set $\{F_1, \ldots, F_r\}$ such that for every $i \in \{1, \ldots, r\}$,

$$F_i = \{X \in Q : \phi_i \mathsf{U} \psi_i \notin X \text{or } \psi_i \in X\}.$$

By Lemma 5, the condition about atomic formulae can be checked in PSPACE and so the transition relation can also be computed in PSPACE.

We define $\mathcal{A}_{1\mathrm{cons}} = \langle Q, Q_0, \delta, F \rangle$ as a Büchi automaton such that $Q = Q_0 = F = Q = \Sigma$ and the transition relation satisfies: $sv \xrightarrow{sv''} sv'$ is in $\delta \overset{\text{def}}{\Leftrightarrow} \langle sv, sv' \rangle$ is one-step consistent and $sv' = sv''$. Since checking whether a triple of sets of IPC*-constraints is a symbolic valuation and checking whether a pair of symbolic valuations is one-step consistent can both be done in polynomial time (see

Lemma 3), the transition relation of $\mathcal{A}_{1\text{cons}}$ can be computed in polynomial time.

It remains to define $\mathcal{A}_{\mathcal{C}}$ that recognizes ω-sequences of symbolic valuations satisfying $(\mathcal{C})$. As done in [DD07], instead of building $\mathcal{A}_{\mathcal{C}}$, it is easier to construct the Büchi automaton $\mathcal{A}_{\mathcal{C}}^{\neg}$ that recognizes the complement language of $\mathrm{L}(\mathcal{A}_{\mathcal{C}})$. The automaton $\mathcal{A}_{\mathcal{C}}^{\neg}$ is essentially the automaton $\mathcal{B}$ defined in [DD07, Sect.6] except that we work with an extended alphabet. We need to consider vertices in the graph that represent constants in CONS' and equality between constants does not need to be explicitly present in the symbolic valuations. Apart from this point, the variables in VAR and the constants in CONS have a uniform treatment in the definition of $\mathcal{A}_{\mathcal{C}}^{\neg}$.

The automaton $\mathcal{A}_{\mathcal{C}}^{\neg}$ non-deterministically guesses in the first part of the run the vertices n_1, n_2 and which path among w_{for} and w_{back} is infinitely often strict. Then it checks that the sequence fails to meet $(\mathcal{C})$. The Büchi acceptance condition guarantees that $<$-labeled edges are infinitely often visited. We store all these pieces of information in the locations. For instance, if the automaton is in the location $\langle a, i, b, j, \mathit{for}, 0\rangle$ at the position $\alpha \geq 0$ of the run, this means that:

- the position of the current vertex of the forward path is $\langle a, \alpha + i\rangle$,
- the position of the current vertex of the backward path is $\langle b, \alpha + j\rangle$,
- the forward path is infinitely often strict.

The last component is only used to note when the forward path visits a strict edge. It takes the value 1 (respectively 0) when the previous transition is (respectively is not) a $<$-transition.

Before defining formally $\mathcal{A}_{\mathcal{C}}^{\neg}$, for $a, a' \in \mathrm{VAR} \cup \mathrm{CONS}'$, $\sim \in \{<, =\}$ and $i, j \in \{0, \ldots, l\}$, we write $(\mathsf{X}^i a \overset{\sim}{\rightarrow} \mathsf{X}^j a') \in G_{sv}$ if there is an edge from the node representing $\mathsf{X}^i a$ at the current position to the node representing $\mathsf{X}^j a'$ (according to the definition of the edge relation in Section 4) which means that one of the following cases arises:

- $a, a' \in \mathrm{VAR}$ and $f(\mathsf{X}^i a) \sim f(\mathsf{X}^j a') \in sv$ (see definition of f in Section 4).
- $a \in \mathrm{VAR}$, $a' \in \mathrm{CONS}$ and $f(\mathsf{X}^i a) \sim a' \in sv$.
- $a' \in \mathrm{VAR}$, $a \in \mathrm{CONS}$ and $a \sim f(\mathsf{X}^j a') \in sv$.
- $a, a' \in \mathrm{CONS}'$ and $a \sim a'$.
- $a \in \mathrm{CONS}' \setminus \mathrm{CONS}$, $a' \in \mathrm{VAR}$ and
  either $\sim$ is equality and $a = f(\mathsf{X}^j a') \in sv$
  or there is $d \in \mathrm{CONS}$ such that $a < d$ and $d \sim f(\mathsf{X}^j a') \in sv$
  or there is $d \in \mathrm{CONS}'$ such that $a < d$ and $d = f(\mathsf{X}^j a') \in sv$.
- $a' \in \mathrm{CONS}' \setminus \mathrm{CONS}$, $a \in \mathrm{VAR}$ and
  either $\sim$ is equality and $f(\mathsf{X}^i a) = a' \in sv$
  or there is $d \in \mathrm{CONS}$ such that $d < a'$ and $f(\mathsf{X}^i a) \sim d \in sv$
  or there is $d \in \mathrm{CONS}'$ such that $d < a'$ and $f(\mathsf{X}^i a) = d \in sv$.

Formally, $\mathcal{A}_{\mathcal{C}}^{\neg} = \langle Q, Q_0, \rightarrow, F\rangle$ is defined as follows:

- $Q = \{q_0\} \uplus \{(\mathrm{VAR} \cup \mathrm{CONS}') \times \{0, \ldots, l\} \times (\mathrm{VAR} \cup \mathrm{CONS}') \times \{0, \ldots, l\} \times \{\mathit{for}, \mathit{back}\} \times \{0, 1\}\}$ where $l = |\phi|_{\mathsf{X}}$,

– $I = \{q_0\}$,

– The transition relation $\rightarrow$ is defined as follows.

(a) $q_0 \xrightarrow{sv} q_0$ for every $sv \in \Sigma$.

(b) $q_0 \xrightarrow{sv} \langle a, i, b, j, for, 0\rangle$ and $q_0 \xrightarrow{sv} \langle a, i, b, j, back, 0\rangle$ for every $a, b \in \mathrm{VAR} \cup \mathrm{CONS}'$, $i, j \in \{0, \ldots, l\}$, $sv \in \Sigma$ and $\mathsf{X}^i a < \mathsf{X}^j b \in G_{sv}$.

(c) $\langle a, i, b, j, p, bin\rangle \xrightarrow{sv} \langle a, i-1, b, j-1, p, bin\rangle$ for every $p \in \{for, back\}$, $bin \in \{0, 1\}$, $i, j \geq 1$ and $sv \in \Sigma$.

(d) $\langle a, 0, b, j, for, bin\rangle \xrightarrow{sv} \langle a', i'-1, b, j-1, for, bin'\rangle$ s.t. $i' > 0$, $j > 0$ and

- $(a \xrightarrow{\leq} \mathsf{X}^{i'} a') \in G_{sv}$ and $bin' = 1$;
  or $(a \xrightarrow{=} \mathsf{X}^{i'} a') \in G_{sv}$ and $bin' = 0$;
- $(\mathsf{X}^{i'} a' \xrightarrow{\leq} \mathsf{X}^j b) \in G_{sv}$.

These rules just check that there is a forward edge from the current node to the next node of the forward path.

(e) $\langle a, i, b, 0, for, bin\rangle \xrightarrow{sv} \langle a, i-1, b, j'-1, for, bin'\rangle$ if $i > 0$, $j' > 0$ and the conditions of (d) are verified when doing the following substitutions

$$a \leftarrow b, \qquad a' \leftarrow b', \qquad b \leftarrow a.$$

This corresponds to check that there is a backward edge from the current node to the next node of the backward path.

(f) $\langle a, 0, b, 0, for, bin\rangle \xrightarrow{sv} \langle a', i', b', j', for, bin'\rangle$ if the obvious combination of the constraints (d) and (e) is verified.

(g) Similar conditions are needed to consider the case where the backward path is infinitely often strict.

– F is the set of states of the form $\langle a, i, b, j, p, 1\rangle$ for every $a, b \in \mathrm{VAR} \cup \mathrm{CONS}'$, $i, j \in \{0, \ldots, l\}$ and $p \in \{for, back\}$.

Lemma 11. *A* CLTL(IPC*) *formula* ϕ *is satisfiable iff* $\mathrm{L}(\mathcal{A}_\phi)$ *is nonempty.*

The proof of this lemma is similar to [DD07, Lemma 6.3]. The main trick is to observe that if $\mathrm{L}(\mathcal{A}_\phi)$ is nonempty then $\mathcal{A}_\phi$ accepts an ultimately periodic ω-sequence so that Lemma 9 can be applied. Since given a formula ϕ we can effectively construct $\mathcal{A}_\phi$ and check whether $\mathrm{L}(\mathcal{A}_\phi)$ is nonempty, the model-checking and satisfiability problems for CLTL(IPC*) are decidable. We also have all the arguments to establish the PSPACE upper bound by using arguments from [Saf88].

Theorem 1. *The satisfiability problem for* CLTL(IPC*) *is* PSPACE*-complete.*

Proof. PSPACE-hardness is a consequence of the PSPACE-hardness of LTL [SC85]. As far as the PSPACE upper bound is concerned, the automata $\mathcal{A}_{\mathrm{LTL}}$, $\mathcal{A}_{1\mathrm{cons}}$ and $\mathcal{A}_{\mathcal{C}}^{\neg}$ are of exponential size in $|\phi|$ and can be built in polynomial space in $|\phi|$.

The automaton $\mathcal{A}_{\mathcal{C}}$ is obtained from $\mathcal{A}_{\mathcal{C}}^{\neg}$ by Safra's construction [Saf88] to complement a Büchi automaton. $\mathcal{A}_{\mathcal{C}}^{\neg}$ has a number of states polynomial in $|\phi|$,

say $P(|\phi|)$. From this, we can build a deterministic Streett automaton which accepts the complement of the language accepted by $\mathcal{A}_{\mathcal{C}}^{\neg}$ and has $\mathcal{O}(2^{P(|\phi|)\times log(|\phi|)})$ states. These are the same arguments as in the proof of [DD07, Theorem 6.6]. This automaton can be converted to an equivalent Büchi automaton $\mathcal{A}_{\mathcal{C}}$ with the same order of states. Hence, $\mathcal{A}_{\mathcal{C}}$ can be built in polynomial space in $|\phi|$.

So, computing the intersection automaton $\mathcal{A}_\phi = \mathcal{A}_{\text{LTL}} \cap \mathcal{A}_{1\text{cons}} \cap \mathcal{A}_{\mathcal{C}}$ can be done in polynomial space in $|\phi|$. Since the emptiness problem for Büchi automata is NLOGSPACE-complete, by [BDG88, Corollary 3.36], we finally get that testing emptiness of $\text{L}(\mathcal{A}_\phi)$ can be done non-deterministically in polynomial space in $|\phi|$. As usual, by Savitch's theorem we get the PSPACE upper bound. □

Note that, all the temporal operators in CLTL(IPC*) are definable in monadic second order logic (MSO). By using [GK03], it is immediate that any extension of CLTL(IPC*) obtained by adding a finite amount of MSO-definable temporal operators remains in PSPACE. Only the automaton $\mathcal{A}_{\text{LTL}}$ needs to be updated.

Another corollary is that the model-checking of the linear-time fragment of the logic of [Čer94] against integral relational automata is in PSPACE (only decidability is established by [Čer94]).

Corollary 2. *The model-checking problem for integral relational automata restricted to the LTL fragment of CCTL* introduced in [Čer94] is in* PSPACE.

6 Conclusion

In this paper, we have introduced the logic CLTL(IPC*) extending formalisms in [Čer94,LM01,DD07,Dem06] and we have shown that both model-checking over IPC*-automata and satisfiability are decidable in polynomial space. The proof heavily relies on a translation into the nonemptiness problem for standard Büchi automata and on the approximation of non ω-regular sets of symbolic models. As a by-product, the model-checking problem over the integral relational automata defined in [Čer94] is also PSPACE-complete when restricted to its LTL fragment. The logic CLTL(IPC*) supports a rich class of constraints including those of the form $x < y$ unlike periodicity constraints from [Dem06] (which are quite useful to compare absolute dates) and comparison with constants unlike logics shown in PSPACE in [DD07]. Abstraction of counter automata by performing reasoning modulo can be encoded in CLTL(IPC*) thanks to the presence of integer periodicity constraints.

To conclude, we mention a few open problems that are worth investigating.

- The model checking of CTL* for integral relational automata is undecidable [Čer94] whereas we have shown that its LTL fragment is PSPACE-complete. Moreover, it is shown in [BG06] that the existential and universal fragments have a decidable model checking problem (complexity is not known). It would be interesting to design other decidable branching-time extensions of CLTL(IPC*).
- The decidability status of the satisfiability problem of the full CTL* extension is also an open question.

– The decidability status of constraint LTL over the domain $\langle \{0,1\}^*, \subseteq \rangle$ is open either with the subword relation or with the prefix relation. Note that constraint LTL over the domain $\langle \{0\}^*, \subseteq \rangle$ is already equivalent to constraint LTL over $\langle \mathbb{N}, <, = \rangle$ that is a strict fragment of CLTL(IPC*).

Acknowledgments. We would like to thank the anonymous referees for their numerous and useful suggestions, remarks, etc. improving significantly the quality of the paper.

References

[AD94] R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[AH94] R. Alur and Th. Henzinger. A really temporal logic. *Journal of the Association for Computing Machinery*, 41(1):181–204, 1994.

[BBFS98] E. Bertino, C. Bettini, E. Ferrari, and P. Samarati. An access control model supporting periodicity constraints and temporal reasoning. *ACM Transactions on Databases Systems*, 23(3):231–285, 1998.

[BBH$^+$06] A. Bouajjani, M. Bozga, P. Habermehl, R. Iosif, P. Moro, and T. Vojnar. Programs with lists are counter automata. In *CAV'06*, volume 4144 of *Lecture Notes in Computer Science*, pages 517–531. Springer, 2006.

[BC02] Ph. Balbiani and J.F. Condotta. Computational complexity of propositional linear temporal logics based on qualitative spatial or temporal reasoning. In *FroCoS'02*, volume 2309 of *Lecture Notes in Artificial Intelligence*, pages 162–173. Springer, 2002.

[BDG88] J. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I.* Springer, Berlin, 2nd edition, 1988.

[BEH95] A. Bouajjani, R. Echahed, and P. Habermehl. On the verification problem of nonregular properties for nonregular processes. In *LICS'95*, pages 123–133, 1995.

[BEM97] A. Bouajjani, J. Esparza, and O. Maler. Reachability analysis of pushdown automata: application to model-checking. In *CONCUR'97*, volume 1243 of *Lecture Notes in Computer Science*, pages 135–150. Springer, 1997.

[Bér95] B. Bérard. Untiming timed languages. *Information Processing Letters*, 55:129–135, 1995.

[BFLP03] S. Bardin, A. Finkel, J. Leroux, and L. Petrucci. FAST: Fast Acceleration of Symbolic Transition systems. In *CAV'03*, volume 2725 of *Lecture Notes in Computer Science*, pages 118–121. Springer, 2003.

[BFLS06] S. Bardin, A. Finkel, E. Lozes, and A. Sangnier. From pointer systems to counter systems using shape analysis. In *AVIS'06*, 2006.

[BG06] L. Bozzelli and R. Gascon. Branching-time temporal logic extended with presburger constraints. In *LPAR'06*, volume 4246 of *Lecture Notes in Computer Science*, pages 197–211. Springer, 2006.

[BH91] F. Baader and P. Hanschke. A scheme for integrating concrete domains into concept languages. In *IJCAI'91*, pages 452–457, 1991.

[BIL06] M. Bozga, R. Iosif, and Y. Lakhnech. Flat parametric counter automata. In *ICALP'06*, number 4052 in Lecture Notes in Computer Science, pages 577–588. Springer, 2006.

[Boi98] B. Boigelot. *Symbolic methods for exploring infinite state spaces*. PhD thesis, Université de Liège, 1998.

[Cau03] D. Caucal. On infinite transition graphs having a decidable monadic theory. *Theoretical Computer Science*, 290:79–115, 2003.

[CC00] H. Comon and V. Cortier. Flatness is not a weakness. In *CSL'00*, volume 1862 of *Lecture Notes in Computer Science*, pages 262–276. Springer, 2000.

[Čer94] K. Čerāns. Deciding properties of integral relational automata. In *ICALP*, volume 820 of *Lecture Notes in Computer Science*, pages 35–46. Springer, 1994.

[CGL94] E. Clarke, O. Grumberg, and D. Long. Model checking and abstraction. *ACM Transactions on Programming Languages and Systems*, 16(5):1512–1542, 1994.

[CJ98] H. Comon and Y. Jurski. Multiple counters automata, safety analysis and Presburger arithmetic. In *CAV'98*, volume 1427 of *Lecture Notes in Computer Science*, pages 268–279. Springer, 1998.

[CM77] A. Chandra and P. Merlin. Optimal implementation of conjunctive queries in relational databases. In *9th ACM Symposium on Theory of Computing*, pages 77–90, 1977.

[DD07] S. Demri and D. D'Souza. An automata-theoretic approach to constraint LTL. *Information and Computation*, 205(3):380–415, 2007.

[Dem06] S. Demri. LTL over integer periodicity constraints. *Theoretical Computer Science*, 360(1–3):96–123, 2006.

[DFGvD06] S. Demri, A. Finkel, V. Goranko, and G. van Drimmelen. Towards a model-checker for counter systems. In *Proceedings of the 4th International Symposium on Automated Technology for Verification and Analysis (ATVA'06)*, volume 4218 of *Lecture Notes in Computer Science*, pages 493–507. Springer, 2006.

[DG05] S. Demri and R. Gascon. Verification of qualitative $\mathbb{Z}$-constraints. In *Proceedings of the 16th International Conference on Concurrency Theory (CONCUR'05)*, volume 3653 of *Lecture Notes in Computer Science*, pages 518–532. Springer, August 2005.

[DI02] Z. Dang and O.H. Ibarra. The existence of ω-chains for transitive mixed linear relations and its applications. *International Journal of Foundations of Computer Science*, 13(6):911–936, 2002.

[DPK03] Z. Dang, P. San Pietro, and R.A. Kemmerer. Presburger liveness verification of discrete timed automata. *Theoretical Computer Science*, 1–3(299):413–438, 2003.

[EFM99] J. Esparza, A. Finkel, and R. Mayr. On the verification of broadcast protocols. In *LICS'99*, pages 352–359, 1999.

[FL02] A. Finkel and J. Leroux. How to compose Presburger accelerations: Applications to broadcast protocols. In *FST&TCS'02*, volume 2256 of *Lecture Notes in Computer Science*, pages 145–156. Springer, 2002.

[FS00] A. Finkel and G. Sutre. Decidability of reachability problems for classes of two counters automata. In *STACS'00*, volume 2256 of *Lecture Notes in Computer Science*, pages 346–357. Springer, 2000.

[Gas07] R. Gascon. *Spécification et vérification de propriétés quantitatives sur des automates à contraintes*. PhD thesis, ENS de Cachan, November 2007.

[GK03] P. Gastin and D. Kuske. Satisfiability and model checking for MSO-definable temporal logics are in PSPACE. In *CONCUR'03*, volume 2761 of *Lecture Notes in Computer Science*, pages 222–236. Springer, 2003.

[GKK+03] D. Gabelaia, R. Kontchakov, A. Kurucz, F. Wolter, and M. Zakharyaschev. On the computational complexity of spatio-temporal logics. In *FLAIRS'03*, pages 460–464, 2003.

[HLP90] E. Harel, O. Lichtenstein, and A. Pnueli. Explicit clock temporal logic. In *Proc. 5th IEEE Symp. Logic in Computer Science (LICS '90), Philadelphia, PA, USA, June 1990*, pages 400–413. IEEE Computer Society Press, 1990.

[HMR05] Th. Henzinger, R. Majumdar, and J.F. Raskin. A classification of symbolic transitions systems. *ACM Transactions on Computational Logic*, 6(1), 2005. to appear.

[Iba78] O. Ibarra. Reversal-bounded multicounter machines and their decision problems. *Journal of the Association for Computing Machinery*, 25(1):116–133, 1978.

[ID01] O. Ibarra and Z. Dang. On removing the stack from reachability constructions. In *ISAAC'01*, volume 2223 of *Lecture Notes in Computer Science*, pages 244–256. Springer, 2001.

[JKMS04] P. Jančar, A. Kučera, F. Moller, and Z. Sawa. DP lower bounds for equivalence-checking and model-checking of one-counter automata. *Information and Computation*, 1(188):1–19, 2004.

[LM01] U. Dal Lago and A. Montanari. Calendars, time granularities, and automata. In *Int. Symposium on Spatial and Temporal Databases*, volume 2121 of *Lecture Notes in Computer Science*, pages 279–298. Springer, Berlin, 2001.

[LS01] G. Logothetis and K. Schneider. Abstraction from counters: an application on real-time systems. In *TIME'01*, pages 214–223. IEEE, 2001.

[Lut04] C. Lutz. NEXPTIME-complete description logics with concrete domains. *ACM Transactions on Computational Logic*, 5(4):669–705, 2004.

[Min67] M. Minsky. *Computation, Finite and Infinite Machines*. Prentice Hall, 1967.

[MOS05] M. Müller-Olm and H. Seidl. Analysis of modular arithmetic. In *ESOP'05*, volume 3444 of *Lecture Notes in Computer Science*, pages 46–60. Springer, 2005.

[MS85] D. Muller and P. Schupp. The theory of ends, pushdown automata, and second-order logic. *Theoretical Computer Science*, 37:51–75, 1985.

[Pup06] G. Puppis. *Automata for Branching and Layered Temporal Structures*. PhD thesis, University of Udine, 2006.

[Rev02] P. Revesz. *Introduction to Constraint Databases*. Springer, New York, 2002.

[Saf88] S. Safra. On the complexity of ω-automata. *Proc. 29th IEEE Symp. on Foundations of Computer Science*, pages 319–327, 1988.

[SC85] A. Sistla and E. Clarke. The complexity of propositional linear temporal logic. *Journal of the Association for Computing Machinery*, 32(3):733–749, 1985.

[VW94] M. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115:1–37, 1994.

[Wol83] P. Wolper. Temporal logic can be more expressive. *Information and Computation*, 56:72–99, 1983.

[WZ00] F. Wolter and M. Zakharyaschev. Spatio-temporal representation and reasoning based on RCC-8. In *KR'00*, pages 3–14, 2000.

A Integral relational automata are restricted IPC*-automata

An integral relational automata (RA) is defined in [Čer94] as a program with a finite amount of variables interpreted in $\mathbb{Z}$. The set OP of operations is composed of the following instructions and guards:

- comparisons of the form $x < y$, $x < d$, $d < y$,
- assignments of the form $x \leftarrow y$, $x \leftarrow d$,
- input value $?x$,
- output value $!x$ or $!d$,
- dummy operation NOP,

where x, y are variables and d is a constant in $\mathbb{Z}$. A relational automaton $\mathcal{A} = \langle Q, \delta, op, g\rangle$ [Čer94] is a finite directed graph where

- Q is a finite set of control states,
- $\delta \subseteq Q \times Q$,
- $op : Q \to \mathrm{OP}$,
- $g : \delta \to \{+, -\}$.

Let $Var(\mathcal{A})$ and $Cons(\mathcal{A})$ be respectively the sets of variables and constants occuring in op. A configuration of $\mathcal{A}$ is a pair $\langle n, v\rangle$ where $n \in Q$ and v is a map $v : Var(\mathcal{A}) \cup Cons(\mathcal{A}) \to \mathbb{Z}$ equal to identity for its restriction to $Cons(\mathcal{A})$. The configuration graph of $\mathcal{A}$ is defined as the pair $\langle S, \to\rangle$ where S is the set of configurations of $\mathcal{A}$ and $\langle n, v\rangle \to \langle n', v'\rangle$ iff there exists $e = \langle n, n'\rangle \in \delta$ such that

1. v and v' are related depending on the nature of $op(n)$:
   - if $op(n)$ is $?x$, then for every $y \in Var(\mathcal{A}) \setminus \{x\}$, $v'(y) = v(y)$,
   - if $op(n)$ is an output value or the dummy operation, then $v' = v$,
   - if $op(n)$ is $x \leftarrow a$, then $v' = v[x \leftarrow v(a)]$,
   - if $op(n)$ is $a < b$, then $v = v'$ and
     either $g(e) = +$ and $v(a) < v(b)$,
     or $g(e) = -$ and $v(a) \geq v(b)$,

   where $a, b \in Var(\mathcal{A}) \cup Cons(\mathcal{A})$, $v(x)$ denotes the value of x in v and $v[x \leftarrow a]$ is such that $v[x \leftarrow z](y) = z$ if $x = y$ and $v[x \leftarrow z](y) = v(y)$ otherwise.

Observe that equality between variables can be tested by performing two negative tests on $a < b$ and $b < a$.

From a relational automaton $\mathcal{A} = \langle Q, \delta, op, g\rangle$, we build a restricted IPC*-automaton $\mathcal{A}'$, having an isomorphic configuration graph in a sense to be precised below. Given $\mathcal{A} = \langle Q, \delta, op, g\rangle$, the restricted IPC*-automaton $\mathcal{A}' = \langle Q', Q'_0, \delta', F'\rangle$ is defined as follows:

1. $Q' = Q'_0 = F' = Q$. Without any loss of generality, we can assume that Q is a finite set of $\mathbb{Z}$ constants not occurring in op.
2. To each $e = \langle n, n'\rangle$ in $\mathcal{A}$, we associate $n \xrightarrow{\phi_e} n'$ in δ', where ϕ_e is a conjunction of constraints defined as follows:

- $\mathsf{X}ic = n'$ is a conjunct of ϕ_e where ic is a new variable taking care of the instruction counter,
- if $op(n) = ?x$ then $\bigwedge_{y \in \mathrm{VAR}(\mathcal{A}) \setminus \{x\}} y = \mathsf{X}y$ belongs to ϕ_e,
- if $op(n)$ is an output or a dummy operation, then $\bigwedge_{y \in \mathrm{VAR}(\mathcal{A})} y = \mathsf{X}y$ belongs to ϕ_e,
- if $op(n) = x \leftarrow a$ then $\bigwedge_{y \in \mathrm{VAR}(\mathcal{A}) \setminus \{x\}} y = \mathsf{X}y \wedge \mathsf{X}x = a$ belongs to ϕ_e,
- if $op(n) = a < b$ then
  if $g(e) = +$ then $a < b \wedge \bigwedge_{y \in \mathrm{VAR}(\mathcal{A})} y = \mathsf{X}y$ belongs to ϕ_e,
  $g(e) = -$ then $a \geq b \wedge \bigwedge_{y \in \mathrm{VAR}(\mathcal{A})} y = \mathsf{X}y$ belongs to ϕ_e.

The configuration graphs of $\mathcal{A}$ and $\mathcal{A}'$ are isomorphic with the following property. The transition $\langle n, v \rangle \rightarrow \langle n', v' \rangle$ belongs to the configuration graph of $\mathcal{A}$ iff $\langle n, \overline{v} \rangle \rightarrow \langle n', \overline{v'} \rangle$ belongs to the configuration graph of $\mathcal{A}'$ where $\overline{v} : \mathit{Var}(\mathcal{A}) \cup \mathit{Cons}(\mathcal{A}) \cup \{ic\} \rightarrow \mathbb{Z}$ is a conservative extension of $v : \mathit{Var}(\mathcal{A}) \cup \mathit{Cons}(\mathcal{A}) \rightarrow \mathbb{Z}$ and $\overline{v}(ic) = n$. The map $\overline{v'}$ is defined similarly. As a corollary, the LTL fragment of CLTL* defined in [Čer94] where the atomic formulae are of the form n, $x < y$ and $x = y$, can be reduced to the model-checking problem for restricted IPC*-automata (just replace n by $ic = n$ to obtain CLTL(IPC*) formulae).