Verification of qualitative Z constraints
Stéphane Demri, Régis Gascon

To cite this version:
Stéphane Demri, Régis Gascon. Verification of qualitative Z constraints. CONCUR 2005 - 16th International Conference on Concurrency Theory, Martin Abadi; Luca de Alfaro, Aug 2005, San Francisco, United States. pp.518-532, 10.1007/11539452_39 . hal-03203570

HAL Id: hal-03203570
https://hal.science/hal-03203570
Submitted on 20 Apr 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Verification of qualitative $\mathbb{Z}$ constraints

Stéphane Demri and Régis Gascon

LSV/CNRS UMR 8643 & INRIA Futurs projet SECSI & ENS Cachan
61, av. Pdt. Wilson, 94235 Cachan Cedex, France
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 $x < y$, it is closed under Boolean operations and it admits a restricted form of existential quantification. This is the largest set of qualitative constraints over $\mathbb{Z}$ known so far, shown to admit a decidable LTL extension. Such constraints are those used for instance in calendar formalisms or in 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. As a by-product, LTL model-checking over integral relational automata is proved complete for polynomial space which contrasts with the known undecidability of its CTL counterpart.

1 Introduction

Model-checking infinite-state systems. The verification of systems with an infinite amount of states has benefited from the numerous decidable model-checking problems for infinite-state systems, including timed automata [AD94], infinite transition graphs [Cau03], or subclasses of counter systems (see e.g. [CJ98]). Even though decidability can be obtained via numerous proof techniques (finite partition of the infinite domain, well-structured systems, Presburger definable reachability sets, reduction to the second-order theory of the binary tree), showing undecidability of model-checking for some classes of infinite-state systems is often easy. After all, the halting problem for Minsky machines is already undecidable. Decidability is more difficult to establish and it can be sometimes regained by naturally restricting the class of models (see e.g. the flatness condition in [CJ98]) or by considering fragments of the specification language (to consider only reachability or repeated reachability for instance).

Systems with variables interpreted in $\mathbb{Z}$. Structures with a finite set of control states augmented with a finite set of variables interpreted either in $\mathbb{Z}$ or in $\mathbb{N}$ (counters) are operational models of numerous infinite-state systems, including broadcast protocols (see e.g. [EFM99,FL02]). The class of counter machines
has numerous undecidable model-checking problems such as the reachability problem but many classes of counter systems have been shown to be decidable: reversal-bounded multicounter machines [Iba78], flat counter systems with affine update functions forming a finite monoid [Boi98,FL02,BFLP03], flat counter systems [CJ98] (weaker class of Presburger guards but no condition on the monoid) and constraint automata with qualitative constraints on $Z$ [DD03].

Our motivation. Constraint automata with qualitative constraints on $Z$ are quite attractive operational models since they can be viewed as abstractions of counter automata where incrementations and decrements are abstracted by operations modulo some power of two. Common programming languages perform arithmetic operators for integer types modulo $2^k$ [MOS05], typically $k$ is either 32 or 64. For example, $x = y + 1$ can be abstracted by $x \equiv 2^k y + 1 \land y < x$. Such an abstraction is well-suited to check safety properties about the original counter system. In the paper, we study a class of constraint automata with a language of qualitative constraints as rich as possible and a companion LTL-like logic to perform model-checking on such operational models. Our framework should be able to deal both with abstractions modulo (see e.g. [CGL94,LS01]) and with integer periodicity constraints used in logical formalisms to deal with calendars [LM01], i.e. constraints of the form $x \equiv_k y + c$. By a qualitative constraint, we mean for instance a constraint that is interpreted as a non-deterministic binary relation, like $x < y$ and $x \equiv 2^k y + 5$ (the relationship between $x$ and $y$ is not sharp).

Our contribution. We introduce a version of constraint LTL over the constraint language $IPC^*$, whose expressions are Boolean combinations of $IPC^{++}$ constraints from [Dem04] and constraints of the form $x < y$. The language $IPC^{++}$ is already closed under Boolean operators and first-order quantification. No constraint of the form $x < y$ occurs in the scope of a quantifier. Otherwise incrementation is definable and it leads to undecidability of the logic. So, as shown in this paper, adding the single type of constraints $x < y$ leads to many technical complications, but not to undecidability. We call CLTL($IPC^*$) the specification language built over $IPC^*$ constraints. 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 shown to be in $\text{PSPACE}$ in [Dem04] whereas constraint LTL over constraints of the form either $x = y$ or $x < y$ is also shown to be in $\text{PSPACE}$ in [DD03]. Both proofs use reductions to the emptiness problem for Büchi automata following the approach in [VW94]. However, the proofs are of different nature: in [Dem04] the complexity upper bound is obtained by a finite model property argument whereas in [DD03] approximations of classes of symbolic models are considered because some formulae can generate non-ω-regular classes of symbolic models. We show that model-checking and satisfiability problems for the logic CLTL($IPC^*$) are decidable (which was open so far) and moreover in $\text{PSPACE}$ ($\text{PSPACE}$-hardness is easy). The proof substantially generalizes what is done for constraint LTL over the domain $(\mathbb{Z}, <, =)$ by con-
considering both new constraints of the form \( x \leq d, d \in \mathbb{Z} \) and integer periodicity constraints. The optimal treatment of constants occurring in such constraints is our main technical contribution. As a corollary, we establish that LTL model-checking over integral relational automata [Čer94] is \( \text{pspace} \)-complete. Hence, even though IPC* is a powerful language of qualitative constraints, the \( \text{pspace} \) upper bound is preserved in CLTL(IPC*). To our opinion, we provide a definite complexity characterization of LTL with qualitative constraints over \( \mathbb{Z} \).

Related work. Reachability problems for subclasses of counter systems have been addressed for instance in [iba78,CJ98,FL02,BFLP03] (see also richer questions in [BEM97,JKMS04]). In our work, we have a full LTL-like language, not restricted to reachability questions, used as a specification language and no restriction on the structure of the models. However, atomic formulae of the specification language are qualitative constraints. If we give up the decidability requirement, LTL over Presburger constraints can be found in [BEH95,CC00].

Constraint LTL over concrete domains (not only restricted to \( \mathbb{Z} \)) has been considered in [WZ00,BC02,DD03,GKK03,Dem04] where often \( \text{pspace} \)-completeness is 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, see e.g. [Lut04]. The language CLTL(IPC*) extends the different LTL-like fragments from [Čer94,LM01,DD03,Dem04] (past-time operators can be added for free in our formalism thanks to [GK03]). The class of IPC*-automata introduced in the paper generalizes the class of integral relational automata from [Čer94] (see details in [DG05]).

Integer periodicity constraints, a special class of Presburger constraints, have found applications in many 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]), DAT ALOG with integer periodicity constraints [TC98] and in real-time logics [AH94].

Omitted proofs can be found in [DG05].

2 The logic CLTL(IPC*)

2.1 Language of constraints

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

\[
p ::= p_{\text{mod}} \mid x < y \mid p \land p \mid \neg p
\]

\[
p_{\text{mod}} ::= x \equiv_k [c_1,c_2] \mid x \equiv y + [c_1,c_2] \mid x = y \mid x < d \mid x = d \mid p_{\text{mod}} \land p_{\text{mod}} \mid \neg p_{\text{mod}} \mid \exists x \ p_{\text{mod}}
\]

where \( x, y \in V, k \in \mathbb{N} \setminus \{0\}, c_1, c_2 \in \mathbb{N} \) and \( d \in \mathbb{Z} \). This language is denoted by IPC*. We write IPC++ to denote its restriction to constraints ranged over by
and $Z'$ its restriction to constraints of the form either $x \sim y$ or $x \sim d$. The symbol $\sim$ is used to mean either $=$ or $<$. The language $Z$ is the restriction of $Z'$ to constraints of the form $x \sim y$. We define a valuation $v$ as a map $v : V \rightarrow Z$ and the satisfaction relation $v \models \phi$ is defined as follows in the standard way:

$$v \models x = y \iff v(x) = v(y); \quad v \models x \sim d \iff v(x) \sim d;$$

$$v \models x \equiv_k [c_1, c_2] \iff v(x) = c \text{ modulo } k \text{ for some } c_1 \leq c \leq c_2;$$

$$v \models x \equiv_k y + [c_1, c_2] \iff v(x) - v(y) = c \text{ modulo } k \text{ for some } c_1 \leq c \leq c_2;$$

$$v \models p \land p' \iff v \models p \text{ and } v \models p'; \quad v \models \neg p \iff v \not\models p;$$

$$v \models \exists x \ p \iff \text{there is } z \in Z \text{ such that } v[x \leftarrow z] \models p$$

where $v[x \leftarrow z](x') = v(x')$ if $x \neq x'$ and $v[x \leftarrow z](x) = z$.

We recall that $x$ is equal to $y$ modulo $k$ if there is $z \in Z$ such that $x - y = k \times z$. We write $x \equiv_k c$ instead of $x \equiv_k [c, c]$, $x \equiv_k y + c$ instead of $x \equiv_k y + [c, c]$ and $v \models X$ where $X$ is a set of IPC*-constraints, whenever $v \models p$ for every $p \in X$.

A constraint $p$ is satisfiable iff there is a valuation $v$ such that $v \models p$. Two constraints are equivalent iff they are satisfied by the same valuations.

Lemma 1. (I) The satisfiability problem for IPC* is PSPACE-complete. (II) Every constraint in IPC* admits an equivalent quantifier-free constraint in IPC*.

Hence, IPC* is a quite well understood fragment of Presburger arithmetic.

2.2 Logical language

We consider the linear-time temporal logic $CLTL(IPC^*)$ whose atomic formulae are defined from constraints in IPC*. The atomic formulae are of the form $p[x_1 \leftarrow X^i x_{j_1}, \ldots, x_r \leftarrow X^i x_{j_r}]$, where $p$ is a constraint of IPC* with free variables $x_1 \ldots x_r$. We substitute each occurrence of the variable $x_i$ with $X^i x_{j_i}$, which corresponds to the variable $x_{j_i}$ preceded by $i_i$ next symbols. Each expression of the form $X^i x_\alpha$ is called a term and represents the value of the variable $x_\alpha$ at the $j^\text{th}$ next state. Here are examples of atomic formulae: $Xy \equiv_{2^{32}} x + 1$ and $x < Xy$.

The set of $CLTL(IPC^*)$ formulae $\phi$ is defined by

$$\phi ::= p[x_1 \leftarrow X^i x_{j_1}, \ldots, x_r \leftarrow X^i x_{j_r}] \mid \neg \phi \mid \phi \land \phi \mid X\phi \mid \phi U \phi,$$

where $p$ belongs to IPC*. The operators $X$ and until (U) are the classical operators used in temporal logics. In the language, all the integers are encoded with a binary representation (this is important for complexity considerations). 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 $\sigma : \mathbb{N} \times V \rightarrow Z$ for $CLTL(IPC^*)$ is an $\omega$-sequence of valuations. The satisfaction relation is defined as follows (we omit the Boolean cases):

$$\sigma, i \models p[x_1 \leftarrow X^i x_{j_1}, \ldots, x_r \leftarrow X^i x_{j_r}] \iff [x_1 \leftarrow \sigma(i + i_1, x_{j_1}), \ldots, x_r \leftarrow \sigma(i + i_r, x_{j_r})] \models \phi;$$

$$\sigma, i \models X\phi \iff \sigma, i + 1 \models \phi;$$

$$\sigma, i \models \phi U \phi' \text{ iff there is } j \geq i \text{ s.t. } \sigma, j \models \phi' \text{ and for every } i \leq l < j, \sigma, l \models \phi.$$
By definition, CLTL(IPC\textsuperscript{*})-models interpret variables but not propositional variables. However, it is easy to encode propositional variables by using atomic formulae of the form $x = 0$ where $x$ is a new variable introduced for this purpose.

### 2.3 Satisfiability and model-checking problems

We recall below the problems we are interested in.

**Satisfiability problem for CLTL(IPC\textsuperscript{*}):**

Given a CLTL(IPC\textsuperscript{*}) formula $\phi$, is there a model $\sigma$ such that $\sigma, 0 \models \phi$?

If we extend IPC\textsuperscript{*} to allow constraints of the form $x < y$ in the scope of $\exists$, then the satisfiability problem for the corresponding constraint LTL-like logic is undecidable since the successor relation is then definable and the halting problem for Minsky machines can be easily encoded.

The model-checking problem rests on IPC\textsuperscript{*}-automata which are constraint automata. An IPC\textsuperscript{*}-automaton $A$ is defined as a Büchi automaton over the infinite alphabet composed of CLTL(IPC\textsuperscript{*}) formulae. In an IPC\textsuperscript{*}-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].

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

**Model-checking problem for CLTL(IPC\textsuperscript{*}):**

Given an IPC\textsuperscript{*}-automaton $A$ and a CLTL(IPC\textsuperscript{*}) formula $\phi$, are there a symbolic $\omega$-word $v = \phi_0 \cdot \phi_1 \cdot \ldots$ accepted by $A$ and a model $\sigma$ (a realization of $v$) such that $\sigma, 0 \models \phi$ and for every $i \geq 0$, $\sigma, i \models \phi_i$?

![Fig. 1. An IPC\textsuperscript{*}-automaton](image)
The satisfiability problem and the model-checking problem are reducible to each other in logspace following techniques from [SC85], by possibly introducing a new variable. In the following sections, we prove results for the satisfiability problem but they also extend to the model-checking problem.

The equivalence problem for Extended Single-String automata [LM01] can be encoded as a model-checking problem for CLTL(IPC*) [Dem04]. Furthermore, the model-checking problem for integral relational automata restricted to the LTL fragment of CCTL* introduced in [Čer94] is a subproblem of the model-checking problem for CLTL(IPC*) (see details in [DG05]). The model-checking problem for CLTL(IPC++)(resp. CLTL(Z)) is shown to be pSPACE-complete in [Dem04] (resp. in [DD03]). However, the proof for IPC++ uses an ω-regular property of the set of models that does not hold when we introduce constraints of the form $x < y$. The problem for CLTL(Z*) is shown to be in EXPSPACE in [DD03] by a translation into CLTL(Z) that increases exponentially the size of formulae (with a binary encoding of the natural numbers).

A restricted IPC*-automaton is defined as an IPC*-automaton such that the labels on transitions are Boolean combinations of atomic formulae with terms of the form $x$ and $Xx$ (see Fig. 1). The logic CCTL*(IPC*) (constraint CTL* over IPC* constraints) is defined as the extension of CLTL(IPC*) with the path quantifiers $\exists$ and $\forall$ but restricted to atomic formulae with no variables in $V$ preceded by $X$. The models of CCTL*(IPC*) are the configuration graphs of restricted IPC*-automata. The satisfaction relation $A, (q, \pi) \models \phi$ is defined in the usual way. The model-checking problem for CCTL*(IPC*) takes as inputs a restricted IPC*-automaton $A$, an initial configuration $(q, 0)$ ($q$ is a control state and $0$ is the initial valuation with null values for the variables) and a CCTL*(IPC*) formula $\phi$ and checks whether $A, (q, 0) \models \phi$. Full CCTL*(IPC*) model-checking can be shown to be undecidable by using developments in [DG05] and [Čer94] (even its CTL-like fragment) and one can show that its LTL fragment is decidable in polynomial space, a new result not captured by [Čer94].

### 3 Properties of the constraint language

In this section, we establish results about the constraint language underlying the logic CLTL(IPC*). In order to define automata that recognize symbolic representations of CLTL(IPC*)-models, the valuations $v$ of the form $V \rightarrow Z$ are represented by symbolic valuations. 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 lcm of $k_1, \ldots, k_n$ where periodicity constraints with relations $\equiv k_1, \ldots, \equiv k_n$ occur in $X$. Observe that $|K|$ is in $O(|k_1| + \cdots + |k_n|)$.
- $C$ is the set of constants $d$ occurring in constraints of the form $x \sim d$.
- $m$ is the minimal element of $C$ and $M$ is its maximal element.
- $C'$ denotes the set of constants $\{m, m-1, \ldots, M\}$. The cardinality of $C'$ is in $O(2^{m+1+M})$ and each element of $C'$ can be binary encoded in binary representation with $O(|m| + |M|)$ bits.
V is the finite set of variables occurring in X.

In the remaining, we assume that the above objects are always defined (possibly by adding dummy valid constraints in order to make the sets non-empty).

A maximally consistent set Y of Zc constraints with respect to V and C is a set of Zc constraints using only the variables from V and the constants from C such that there is a valuation v : V → Z verifying v ≳ Y and for any proper extension Z of Y, there is no valuation v′ : V → Z verifying v′ ≳ Z.

A valuation is abstracted by three disjoint finite sets of IPC* constraints like regions for timed automata.

**Definition 1.** Given a finite set X of IPC* constraints, a symbolic valuation sv is a triple ⟨Y1, Y2, Y3⟩ such that

- Y1 is a maximally consistent set of Zc constraints wrt V and C.
- Y2 is a set of constraints of the form x = d with x ∈ V and d ∈ C′ \ C. Each x ∈ V occurs at most in one constraint of the form x = d in Y2. Moreover, for every x ∈ V, (x = d) ∈ Y2 for some unique d ∈ C′ \ C iff for every d′ ∈ C, (x = d′) /∈ Y1 and \{m < x, x < M\} ⊆ Y1.
- Y3 is a set of constraints of the form x ≡K c with x ∈ V and 0 ≤ c ≤ K − 1. Each x ∈ V occurs exactly in one constraint of the form x ≡K c in Y3.

A consequence of Definition 1 is that in a symbolic valuation sv = ⟨Y1, Y2, Y3⟩, no constraint occurs in more than one set. That is why, given an IPC* constraint p, we write p ∈ sv instead of p ∈ Y1 ∪ Y2 ∪ Y3. A symbolic valuation is satisfiable iff there is a valuation v : V → Z such that v ≳ Y1 ∪ Y2 ∪ Y3.

**Lemma 2.** Let X be a finite set of IPC* constraints and sv = ⟨Y1, Y2, Y3⟩ be a triple composed of IPC* constraints such that Y1 is a set of Zc constraints built over V and C, Y2 is a set of Zc constraints of cardinality at most |V| built over V and C′ \ C, Y3 is a set of constraints of the form x ≡K c of cardinality |V|.

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.

Maximal consistency of Y1 can be checked in polynomial-time by using developments from [Cer94, Lemma 5.5]. Indeed, given a set Y of Zc constraints built over V and C, a graph GY can be built such that Y is maximally consistent wrt V and C iff GY satisfies the conditions below. GY is a structure ⟨V ∪ C, ≃, ≈⟩ such that n ≃ n′ ≔ n ~ n′ belongs to Y. Following [Cer94, Lemma 5.5], Y is maximally consistent iff GY satisfies the conditions below:

- (MC1) For all n, n′, either n ≃ n′ or n′ ≃ n for some ~ ∈ {<, =}.
- (MC2) ≃ is a congruence relation compatible with ≈.
- (MC3) There is no path n0 ≃ n1 ≃ ... ≃α−1 nα with n0 = nα and < occurs in \{<0, <1, ..., <α−1\}.
- (MC4) For all d1, d2 ∈ C, d1 ∼ d2 implies d1 ≃ d2.
- (MC5) For all d1, d2 with d1 ≤ d2, there is no path n0 ≃ n1 ≃ ... ≃α−1 nα with n0 = d1 and nα = d2 such that the cardinality of \{i : i <, 1 ≤ i ≤ α − 1\} is strictly more than d2 − d1.
The symbolic representations of valuations contain the relevant information to evaluate constraints.

**Lemma 3.** Let $X$ be a finite set of IPC* constraints. (I) For every valuation $v : V \rightarrow \mathbb{Z}$ there is a unique symbolic valuation $sv(v) = \langle Y_1, Y_2, Y_3 \rangle$ such that $v \models Y_1 \cup Y_2 \cup Y_3$. (II) For all valuations $v, v'$ such that $sv(v) = sv(v')$ and for every $p \in X$, $v \models p$ iff $v' \models p$.

The proof of (I) is by an easy verification whereas (II) is shown by structural induction on $p$ similarly to the proof of [Dem04, Lemma 1]. By Lemma 3, a symbolic valuation is an equivalence class of valuations.

Given a symbolic valuation $sv$ and $p$ a constraint, we write $sv \models_{sym} p$ for every valuation $v$ such that $sv(v) = sv$, $v \models p$.

**Lemma 4.** The problem of checking whether $sv \models_{sym} p$ is PSPACE-complete (given that the syntactic resources used in $p$ are included in those used for the symbolic valuation $sv$).

### 4 Satisfiable $\omega$-sequences of symbolic valuations

Given a CLTL(IPC*) formula $\phi$, we write IPC*$\phi$ to denote the set of IPC* constraints $p$ such that some atomic formula of the form $p[x_1 \leftarrow X^1 x_j, \ldots, x_r \leftarrow X^r x_j]$ occurs in $\phi$. To IPC*$\phi$ we associate the objects relative to any finite set of IPC* constraints. The set $V$ denotes the set of variables occurring in $\phi$. We write $|\phi|_X$ to denote the maximal natural number $i$ such that $X^i x$ occurs in $\phi$ for some variable $x$. $|\phi|_X$ is called the $X$-length of $\phi$. Without any loss of generality, we can assume that $|\phi|_X \geq 1$. In the following, we assume that $V = \{x_1, \ldots, x_s\}$ and $|\phi|_X = l$. We write Terms$(\phi)$ to denote the set of terms of the form $X^\beta x_\alpha$ with $\beta \in \{0, \ldots, l\}$ and $\alpha \in \{1, \ldots, s\}$.

Let $V'$ be a set of variables of cardinality $|\text{Terms}(\phi)|$ and $f : \text{Terms}(\phi) \rightarrow V'$ be an unspecified bijection such that $f$ and $f^{-1}$ can be computed in polynomial time. By extension, for every atomic subformula $p$ of $\phi$, $f(p)$ is obtained from $p$ by replacing each occurrence of $X^\beta x_\alpha$ by $f(X^\beta x_\alpha)$. The map $f^{-1}$ is used in a similar fashion. A symbolic valuation wrt $\phi$ is a symbolic valuation built over the set of variables $V'$, $C$ and $K$.

We say that a pair $(\langle Y_1, Y_2, Y_3 \rangle, \langle Y'_1, Y'_2, Y'_3 \rangle)$ of symbolic valuations wrt $\phi$ is one-step consistent $\iff$

1. $f(X^i x_i) \sim f(X^{i'} x_{i'}) \in Y_1$ and $j, j' \geq 1$ imply $f(X^{j-1} x_i) \sim f(X^{j'-1} x_{i'}) \in Y'_1$.
2. $f(X^i x_i) \sim d \in Y_1 \cup Y_2$ and $j \geq 1$ imply $f(X^{j-1} x_i) \sim d \in Y'_1 \cup Y'_2$.
3. $f(X^i x_i) \equiv_K c \in Y_3$ and $j \geq 1$ imply $f(X^{j-1} x_i) \equiv_K c \in Y'_3$.

An $\omega$-sequence $\rho$ of satisfiable symbolic valuations wrt $\phi$ is one-step consistent $\iff$ for every $j \in \mathbb{N}$, $(\rho(j), \rho(j+1))$ is one-step consistent. A model for $\rho$ is defined as a CLTL(IPC*)-model $\sigma$ such that for all $j \in \mathbb{N}$ and $p \in \rho(j)$, $\sigma, j \models f^{-1}(p)$. In order to simplify the future developments, we write $\rho_f$ to denote the $\omega$-sequence obtained from $\rho$ by substituting each occurrence of some variable $x$ by $f^{-1}(x)$. 
One-step consistent ω-sequences of symbolic valuations wrt φ define abstractions of models for φ. We represent a one-step consistent sequence ρ as an infinite labeled structure \( G_\rho = \langle (V \cup C') \times \mathbb{N}, \to, \preceq, mod \rangle \) where \( mod : (V \cup C') \times \mathbb{N} \to \{0, \ldots, K - 1\} \):

\[
\langle x, i \rangle \sim \langle y, j \rangle \quad \text{iff either } i \leq j \text{ and } x \sim X^{j-i} y \in \rho_f(i) \quad \text{or } i > j \text{ and } X^{j-i} x \sim y \in \rho_f(j),
\]

\[
\langle x, i \rangle \equiv \langle d, j \rangle \quad \text{iff } x = d \in \rho_f(i),
\]

\[
\langle x, i \rangle \preceq \langle x, j \rangle \quad \text{iff } x = d \in \rho_f(j),
\]

\[
\langle x, i \rangle \preceq \langle d, j \rangle \quad \text{iff there is } d' \sim d \text{ such that } x \sim' d' \in \rho_f(i) \text{ and } \in \{\sim, \sim'\},
\]

\[
\langle d, i \rangle \preceq \langle x, j \rangle \quad \text{iff there is } d \sim d' \text{ such that } d' \sim' x \in \rho_f(j) \text{ and } \in \{\sim, \sim'\},
\]

\[
\langle d_1, i \rangle \preceq \langle d_2, j \rangle \quad \text{iff } d_1 \sim d_2, \quad \text{mod}(\langle x, i \rangle) = c \quad \text{iff } x \equiv_K c \in \rho_f(i) \text{ and } \text{mod}(\langle d, i \rangle) = c \text{ iff } d \equiv_K c.
\]

For all \( x, y \in V, d_1, d_2 \in C \) and \( i, j \in \mathbb{N} \) such that \( |i - j| \leq l \). By construction of \( G_\rho \), the variables and constants are treated in a similar fashion. It is worth observing that \( G_\rho \) is well-defined because \( \rho \) is one-step consistent. The construction ensures that the “local” representation of every \( \rho(i) \) verifies the conditions (MC1) to (MC5) of Sect. 3.

In the following, we say that a vertex represents the constant \( d \) if it is of the form \( \langle d, i \rangle \) for some \( i \). The level of a node \( n = \langle a, t \rangle \) in \( G_\rho \) is \( t \), and is denoted by \( \text{lev}(n) \). There is some redundancy in \( G_\rho \) for the nodes of the form \( \langle d, i \rangle \). However, this is useful to establish strict relationships between \( \rho \) and \( G_\rho \).

**Example 1.** Assuming that \( C = \{2, 4\} \), \( K = 2 \), \( V' = \{x, x'\} \) \( (f(x) = x \) and \( f(Xx) = x') \) and \( l = 1 \), let us define the sequence \( \rho = sv^0 \cdot (sv^1 \cdot sv^2)^\omega \) where \( sv^0 = (Y^0_1, Y^0_2, Y^0_3) \) such that \( Y^0_1 = \{x = x, x' = x', x < x', 2 < x, x < 4, x' = 4, x' > 2\} \), \( Y^0_2 = \{x = 3\} \) and \( Y^0_3 = \{x \equiv_2 1, x' \equiv_2 0\} \). The symbolic valuation \( sv^1 = (Y^1_1, Y^1_2, Y^1_3) \) satisfies: \( Y^1_1 = (Y^0_1 \setminus \{2 < x, x < 4, x' = 4, x' > 2\}) \cup \{4 < x, 4 < x'\} \), \( Y^1_2 = \emptyset \) and \( Y^1_3 = \{x \equiv_2 0, x' \equiv_2 1\} \). The symbolic valuation \( sv^2 = (Y^2_1, Y^2_2, Y^2_3) \) verifies \( Y^2_1 = Y^1_1, Y^2_2 = Y^1_2 \) and \( Y^2_3 = Y^1_3 \). The graph \( G_\rho \) is presented in Fig. 2. In order to simplify the representation, closure by transitivity for \( \preceq \) and the fact that \( \equiv \) is a congruence are omitted. The function \( \text{mod} \) is directly encoded in the node label.

A path in \( G_\rho \) is a sequence (possible infinite) of the form \( n_0 \sim n_1 \sim n_2 \sim \ldots \). For any finite path \( w = n_0 \sim n_1 \sim n_2 \sim \ldots \sim n_{\alpha-1} \sim n_\alpha \), its strict length \( \text{slen}(w) \) is the cardinality of \( \{i : 0 \leq i \leq \alpha - 1, \sim_i \text{ equals } <\} \). When \( w \) has a strict length greater than 1, we say that \( w \) is strict. A finite path \( w \) such that \( n_0 = n_\alpha \) is called a cycle. The strict length between two nodes \( n_1 \) and \( n_2 \), written \( \text{slen}(n_1, n_2) \), is the least upper bound of the strict lengths of finite paths between \( n_1 \) and \( n_2 \). By convention, if there is no path between \( n_1 \) and \( n_2 \), \( \text{slen}(n_1, n_2) \) takes the value \( -\infty \). In Fig. 2, \( \text{slen}(\langle 2, 2 \rangle, \langle x, 3 \rangle) = 4 \).

In Lemma 5 below, the one-step consistency of \( \rho \) 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 5. Let $\rho$ be a one-step consistent sequence.

(I) $G_{\rho}$ 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 $\langle d, j \rangle \preceq n$, otherwise $\langle d, j \rangle \equiv n$.

(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 $n \preceq \langle d, j \rangle$, otherwise $n \equiv \langle d, j \rangle$.

Corollary 1. Let $\rho$ be a one-step consistent sequence and $G_{\rho}$ its graph representation. Then, for all nodes $\langle d_1, i \rangle$ and $\langle d_2, j \rangle$ in $G_{\rho}$ representing constants such that $d_1 \leq d_2$, $\text{slen}(\langle d_1, i \rangle, \langle d_2, j \rangle) = d_2 - d_1$.

So far, we have stated properties about the graph $G_{\rho}$. Below, we establish simple conditions on $G_{\rho}$ equivalent to the existence of a model for $\rho$. An edge-respecting labeling for $G_{\rho}$ is a map $\text{lab} : (V \cup C') \times \mathbb{N} \to \mathbb{Z}$ such that for all nodes $n_1, n_2$, $n_1 \sim n_2$ implies $\text{lab}(n_1) \sim \text{lab}(n_2)$ and for every node $n$, $\text{lab}(n) \equiv K \pmod{n}$. Additionally, $\text{lab}$ is said to be strict if for every $\langle d, i \rangle$ in $G_{\rho}$, $\text{lab}(\langle d, i \rangle) = d$.

Lemma 6. A one-step consistent sequence $\rho$ has a model iff $G_{\rho}$ has a strict edge-respecting labeling.

The proof is quite direct by unfolding the definitions. A refinement is possible.

Lemma 7. A one-step consistent sequence $\rho$ has a model iff $G_{\rho}$ has an edge-respecting labeling (not necessarily strict).

 Lemmas 6 and 7 state correspondences between $\rho$ and its graphical representation $G_{\rho}$. However, we need a more abstract characterization of the one-step consistent sequences admitting a model (see Lemma 8 below).

Lemma 8. Let $\rho$ be a one-step consistent sequence. The graph $G_{\rho}$ has an edge-respecting labeling iff for all nodes $n_1, n_2$ in $G_{\rho}$, $\text{slen}(n_1, n_2) < \omega$. 
By construction of \( G_\rho \), for all nodes \( \langle d_1, i \rangle \) and \( \langle d_2, j \rangle \) representing constants such that \( d_1 \leq d_2 \), \( \text{slen}(\langle d_1, i \rangle, \langle d_2, j \rangle) = d_2 - d_1 \) (see Corollary 1). That is why, in Lemma 8, there is no additional constraint for nodes of the graph representing constants.

Lemma 8 characterizes the set of sequences having a model but what we really need is to recognize them with automata. The main difficulty rests on the fact that the set of satisfiable one-step consistent \( \omega \)-sequences of satisfiable symbolic valuations is not \( \omega \)-regular, a consequence of [DD03] for the fragment \text{CLTL}(Z). In order to approximate this class of sequences, we define below a condition \((C)\) shown to be \( \omega \)-regular such that for every one-step consistent \( \omega \)-sequence \( \rho \) of satisfiable symbolic valuations that is ultimately periodic, \( \rho \) has a model iff \( G_\rho \) satisfies \((C)\).

An infinite forward (resp. backward) path in \( G_\rho \) is defined as a sequence \( w : \mathbb{N} \rightarrow (V \cup C') \times \mathbb{N} \) such that: for every \( i \in \mathbb{N} \), there is an edge \( w(i) \xrightarrow{\omega} w(i+1) \) (resp. \( w(i+1) \xrightarrow{\omega} w(i) \)) in \( G_\rho \) and if \( \text{lev}(w(i)) = j \), then \( \text{lev}(w(i+1)) \geq j + 1 \).

The path \( w \) is infinitely often strict \( \forall \; i \geq 0 \) for every \( i \geq 0 \), there is \( j \geq i \) such that \( w(j) \xrightarrow{\omega} w(j+1) \) (resp. \( w(j+1) \xrightarrow{\omega} w(j) \)). The condition \((C)\) on the graph \( G_\rho \) is: there do not exist vertices \( n_1 \) and \( n_2 \) in \( G_\rho \) with \( |\text{lev}(n_1) - \text{lev}(n_2)| \leq l \) satisfying

\begin{enumerate}
\item[(AP1)] there is an infinite forward path \( w_{\text{for}} \) from \( n_1 \),
\item[(AP2)] there is an infinite backward path \( w_{\text{back}} \) from \( n_2 \),
\item[(AP3)] either \( w_{\text{for}} \) or \( w_{\text{back}} \) is infinitely often strict, and
\item[(AP4)] for all \( i, j \in \mathbb{N} \), whenever \( |\text{lev}(w_{\text{for}}(i)) - \text{lev}(w_{\text{back}}(j))| \leq l \), \( w_{\text{for}}(i) \xrightarrow{\omega} w_{\text{back}}(j) \) in \( G_\rho \).
\end{enumerate}

We say an infinite word is ultimately periodic if it is of the form \( \tau \cdot \delta^\omega \) for some finite words \( \tau \) and \( \delta \).

**Lemma 9.** Let \( \rho \) be one-step consistent \( \omega \)-sequence of satisfiable symbolic valuations that is ultimately periodic. Then \( \rho \) admits a model iff \( G_\rho \) satisfies \((C)\).

Thanks to the way \( G_\rho \) is built from \( \rho \), \((C)\) does not explicitly mention the constants in \( C' \) and the constraints of the form \( x \equiv_K c \). Hence, Lemma 9 can be proved as [DD03, Lemma 6.2]: the map \text{mod} in \( G_\rho \) is ignored and a uniform treatment for all nodes in \((V \cup C') \times \mathbb{N} \) is provided. In [DD03, Lemma 6.2], there are no nodes of the form \( C' \times \mathbb{N} \) but we take into account their specificity in our construction of \( G_\rho \). If \( \rho \) admits a model then by Lemma 8 it satisfies the condition \((C)\). Conversely, let \( \rho = \tau \cdot \delta^\omega \) be an ultimately periodic one-step consistent \( \omega \)-sequence. We can show that if \( \rho \) has no model then it does not satisfy \((C)\). By Lemma 8, if \( \rho \) has no model, then there exist two vertices \( n_1 \) and \( n_2 \) such that \( \text{slen}(n_1, n_2) = \omega \). One can construct a finite path \( w \) between \( n_1 \) and \( n_2 \) long enough so that paths \( w_{\text{for}} \) and \( w_{\text{back}} \) satisfying the conditions \((AP1)-(AP4)\) can be constructed, witnessing that \( G_\rho \) does not satisfy \((C)\). The construction of \( w_{\text{for}} \) and \( w_{\text{back}} \) from \( w \) uses the fact that \( \rho \) is ultimately periodic by repeating infinitely finite subpaths. The construction of such infinite paths can be done smoothly by using the properties established in this section (see e.g. Lemma 5). As the proof is not essentially different from [DD03, Lemma 6.2] modulo slight changes mentioned above, we omit it here.
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 Aφ such that φ is CLTL(IPC⋆) satisfiable iff L(Aφ) is non-empty. Moreover, we establish that emptiness of L(Aφ) can be checked in polynomial space in |φ|.

From the technical viewpoint, the construction of Aφ as the intersection of three Büchi automata can be done quite smoothly thanks to the previous results. In the following, V, V′, C and C′ are the sets of variables and constants associated to φ as defined in Sect. 4. Moreover, K, m and M are constants with their usual meaning and we use the map f: Terms(φ) → V′ as previously.

Unlike LTL, the language recognized by the Büchi automaton Aφ is not a set of models but rather a set of symbolic models. We write Σ to denote the set of satisfiable symbolic valuations wrt φ. A symbolic model for φ is an ω-sequence ρ: N → Σ. We write ρ| =′ φ where the symbolic satisfaction relation | =′ is defined as = except at the atomic level: ρ, i | =′ p if and only if ρ(i) | =symb f(p) where | =symb is the satisfaction relation between symbolic valuations and constraints.

By Lemma 4 and by using standard techniques for LTL [VW94], checking whether there is a symbolic model ρ satisfying ρ| =′ φ can be done in PSPACE (see more details below). Since every model for φ generates a unique symbolic model for φ, 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 ρ| =′ φ and ρ has a model.

All the following automata are built over the alphabet Σ which is of exponential size in |φ|. The automaton Aφ is formally defined as the intersection A_LTL ∩ A_cons ∩ A_C of Büchi automata where L(A_LTL) is the set of symbolic models satisfying φ, L(A_cons) is the set of one-step consistent sequences of satisfiable symbolic valuations, and L(A_C) is the set of sequences of symbolic valuations verifying (C). We briefly explain below how these automata are built. The automaton A_LTL is obtained from [VW94] with a difference for atomic formulae. We define cl(φ) the closure of φ as usual, and an atom of φ is a maximally consistent subset of cl(φ). We define A_LTL = (Q, Q_0, →, F) as the generalized Büchi automaton below:

- Q is the set of atoms of φ and Q_0 = {X ∈ Q : φ ∈ X},
- X → Y iff (atomic constraints) for every atomic formula p in X, sv |=symb f(p), (one step) for every Xψ ∈ cl(φ), Xψ ∈ X iff ψ ∈ Y,
- let {ψ_1 U φ_1, ..., ψ_r U φ_r} be the set of until formulas in cl(φ). We pose F equal to {F_1, ..., F_r} with for every i ∈ {1, ..., r}, F_i = {X ∈ Q : ψ_i U φ_i ∉ X or φ_i ∈ X}.

By Lemma 4, the condition about atomic formulae can be checked in PSPACE. Hence, the transition relation can be computed in PSPACE.
We define $A_{1\text{cons}} = \langle Q, Q_0, \rightarrow, F \rangle$ as a Büchi automaton such that $Q = Q_0 = F = Q = \Sigma$ and the transition relation satisfies: $sv \xrightarrow{sv''} sv' \iff (sv, sv')$ is one-step consistent and $sv' = sv''$. Since checking whether a symbolic valuation is satisfiable can be done in $P$ (Lemma 2) and checking whether a pair of symbolic valuations is one-step consistent can be also done in $P$, the transition relation of $A_{1\text{cons}}$ can be computed in $P$.

It remains to define $A_C$ that recognizes $\omega$-sequences of symbolic valuations satisfying $(C)$. As done in [DD03], instead of building $A_C$, it is easier to construct the Büchi automaton $A_C^\neg$ that recognizes the complement language of $L(A_C)$. The automaton $A_C^\neg$ is essentially the automaton $B$ defined in [DD03, page 20] except that we work with a different type of alphabet. We need to consider vertices in the graph that represent constants in $C$ and equality between constants does not need to be explicitly present in the symbolic valuations (see details in [DG05]).

**Lemma 11.** A CLTL(IPC*) formula $\phi$ is satisfiable iff $L(A_\phi)$ is non-empty.

The proof of this lemma is similar to [DD03, Lemma 6.3]. The main trick is to observe that if $L(A_\phi)$ is non-empty then $A_\phi$ accepts an ultimately periodic $\omega$-sequence so that Lemma 9 can be applied. Since given a formula $\phi$ we can effectively construct $A_\phi$ and check whether $L(A_\phi)$ is empty, 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 subtle arguments from complexity theory and [Saf89].

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

All the temporal operators in CLTL(IPC*) are definable in monadic second order logic (MSO) and 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 $A_{LTL}$ needs to be updated.

**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 the paper, we have introduced the logic CLTL(IPC*) extending formalisms in [Čer94,LM01,DD03,Dem04] 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 emptiness problem for standard Büchi automata and on the approximation of non $\omega$-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 [Dem04] (which are quite useful to compare absolute dates) and comparison with constants unlike logics
shown in pSPACE in [DD03]. 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.

- CTL* for integral relational automata is undecidable [Čer94] whereas we have shown that its LTL fragment is pSPACE-complete. It is interesting to design other decidable fragments of CTL* strictly more expressive than Boolean combinations of LTL formulae.
- The decidability status of constraint LTL over the domain $(\{0,1\}^*, \subseteq)$ is open either with the subword relation or with the prefix relation. Constraint LTL over the domain $(\{0\}^*, \subseteq)$ is already equivalent to constraint LTL over $\langle \mathbb{N}, <, = \rangle$ that is a strict fragment of CLTL(IPC*).
- The decidability status of CLTL(IPC*) extended with constraints of the form $3x + 2Xy \equiv_5 3$ is open. They are considered in [MOS05] but not integrated in any LTL-like language.

References


