On the Verification Problem of Nonregular Properties for Nonregular Processes

Ahmed Bouajjani*‡ Rachid Echahed†‡ Peter Habermehl*‡

Abstract

We investigate the verification problem of infinite-state processes w.r.t. nonregular properties, i.e., nondefinable by finite-state ω-automata. We consider processes in the algebra PA which provides sequential and parallel (merge) composition, nondeterministic choice, and recursion. The algebra PA integrates and strictly subsumes the algebras BPA (context-free processes) and BPP (Basic Parallel Processes). On the other hand, we consider properties definable in a new temporal logic called CLTL which is an extension of the linear-time temporal logic LTL with two kinds of constraints on traces: constraints on the numbers of occurrences of states expressed using Presburger formulas (occurrence constraints), and constraints on the order of appearance of states expressed using finite-state automata (pattern constraints). Pattern constraints allow to capture all the ω-regular properties whereas occurrence constraints allow to define nonregular properties. Then, we present (un)decidability results concerning the verification problem for the different classes of processes mentioned above and different fragments of CLTL.

1 Introduction

Automatic verification of processes has been extensively studied in the last decade, and important progress has been made in the case of finite-state processes. For such processes, various verification methods have been developed and successfully applied, as behavioural equivalence or preorder testing [Mil80, Par81, KS83], and model-checking w.r.t. logic-based specifications [QS82, CES83, VW86, EL86].

Recently, intensive investigations have been consecrated to the extension of these verification methods to the case of infinite-state processes [Hüt91, AJ93, Chr93, Esp93]. One of the most important directions of these investigations concerns the processes expressible in the algebras BPA (basic process algebra) [BBK87] and BPP (basic parallel processes) [Chr93]. The terms in BPA are constructed using nondeterministic choice, recursion and sequential composition whereas in BPP, the sequential composition is replaced by the prefixing operation and asynchronous parallel composition (merge operation). While BPA processes generate context-free languages (sets of finite traces) [BBK87], BPP processes generate a subclass of context-sensitive languages which is not comparable with the class of context-free languages [Chr93]. The two algebras BPA and BPP are fragments of the algebra PA (process algebra) [BK88] where both sequential and parallel composition are allowed.

Important results have been obtained concerning the comparison between either BPA processes or BPP's w.r.t. behavioural equivalences and preorders. Mainly, it has been shown that (strong) bisimulation is decidable for processes in each of these two algebras, whereas trace equivalence is undecidable for both of them [CH93].

Concerning the verification w.r.t. logic-based specifications, in general, the existing results consider as specification languages either the branching-time or the linear-time propositional μ-calculus [Koz83, Var88]. For BPA processes, it can be deduced from [MS85] that the verification problem for BPA's w.r.t. the branching-time μ-calculus is decidable, and a model-checking algorithm is proposed in [BS92] for its alternation-free fragment. Moreover, it can be deduced from [CG77] that the verification problem for BPA's w.r.t. the linear-time μ-calculus is decidable. As for BPP processes, the verification problem w.r.t. the branching-time μ-calculus has been shown undecidable whereas it is decidable in the case of the linear-time μ-calculus [Bra92, Esp93].

We can observe that all these works consider the verification problem of *nonregular* processes in BPA or BPP w.r.t. *regular* properties since propositional μ-calculi can express only properties definable by *finite-state* automata on infinite sequences or trees [Tho90].

In this paper, we address the more general problem

*VERIMAG, Miniparc-Zirst, Rue Lavoisier, 38330 Montbonnot St-Martin, France.

†LGI-IMAG, BP53, 38041 Grenoble cedex, France.

‡Ahmed.Bouajjani@imag.fr, Rachid.Echahed@imag.fr, Peter.Habermehl@imag.fr

of verifying nonregular (BPA, BPP, or PA) processes w.r.t. *nonregular* properties. Indeed, a wide class of the relevant properties of infinite-state processes are nonregular. For instance, in the specification of a communication protocol, we may require that

> ***during every session, the number of acknowledgements never exceeds the number of requests, and furthermore, every session terminates and has exactly the same numbers of requests and acknowledgements.***

Actually, significant properties of infinite-state processes are essentially temporal properties involving constraints on *numbers of occurrences* of events (or numbers of states satisfying some state properties). Clearly, these properties can be expressed neither in the usual specification logics like propositional temporal logics and μ-calculi nor by finite-state ω-automata. In [BER94], we have proposed a temporal logic, called PCTL (Presburger CTL), allowing to express such properties. This logic is an extension of the branching-time temporal logic CTL [CES83] with the ability of constraining the numbers of occurrences of state properties using Presburger formulas. We have identified a fragment of PCTL for which the verification problem is decidable for all PA processes [BER94, BEH95].

In this paper, we alternatively consider the case of linear-time logics. For that, we define a new logic called Constrained LTL (CLTL) whose regular part is expressively maximal, i.e., equivalent to ω-regular automata, and which also allows to express constraints on numbers of occurrences of state formulas, as in PCTL.

The logic CLTL is obtained by extending the propositional temporal logic LTL [Pnu77] in two different manners. First of all, it is well known that LTL defines exactly the class of ω-star-free languages [Tho79, Wol83, LPZ85] and thus its expressive power is limited by comparison with logics like ETL [Wol83] and the linear-time μ-calculus [Var88]. Then, to achieve the full expressiveness of ω-automata, we add to LTL the ability of constraining finite computations by requiring their membership to languages of finite-state automata on finite sequences. This is done in a very flexible way. We dispose of *position variables* to refer to different points in a computation, which allows to require that the sequence since some designated point must be accepted by some given automaton. For example, the formula $\downarrow u.\ \Box A^u$ expresses the safety property saying that from the current state (corresponding to the position u), every finite computation is accepted by the automaton A.

Moreover, to express nonregular properties, we use *occurrence variables* that can be associated with state formulas and then used to express constraints on the numbers of occurrences of states satisfying these state formulas. The constraints are expressed in the language of Presburger arithmetics. For instance, in the formula $[x : \pi].\varphi$, we associate the state formula π with the variable x. Then, x counts the number of occurrences of π from the current state. Using this notation, the property informally described above can be expressed in CLTL by the formula:

$$\Box\,(\text{BEGIN} \Rightarrow [x, y : \text{REQ}, \text{ACK}].\\ ((x \geq y)\,\mathcal{U}\,\text{END}) \wedge \Box\,(\text{END} \Rightarrow (x = y))) \qquad (1)$$

It can be seen that the extension of LTL by constraints on numbers of occurrences of state properties allows to characterize a large class of nonregular properties. These properties can be context-free as in the example above, but also non context-free (context-sensitive). Moreover, the combination of automata with such arithmetical constraints allows to define new classes of nonregular properties as for instance *constrained safety properties* (see Section 4.3).

We prove that the validity problem of CLTL is highly undecidable (Π^1_1-complete). This holds also for the sublogic PLTL (Presburger LTL) corresponding to CLTL formulas without automata. A direct consequence of this result is that the verification problems of PLTL and CLTL are undecidable even for regular processes.

Therefore, we consider two positive fragments of PLTL and CLTL called respectively PLTL$^+$ and CLTL$^+$, and investigate their verification problems for processes in BPA, BPP, and PA.

First of all, we prove that the verification problem of PLTL$^+$ is decidable for all PA processes. This result is analogous to the one we have established in [BEH95]. Our proof is based on a reduction of the verification problem to the validity problem in Presburger arithmetics.

Then, we investigate the effect of adding automata and consider the fragment CLTL$^+$. We prove that the verification problem for BPA processes w.r.t. CLTL$^+$ formulas is decidable. To establish this new decidability result, we adopt the same approach as previously, and show that in this case also, the verification problem is reducible (although in a different manner) to the validity problem in Presburger arithmetics. On the other hand, we show that for every extension of BPA using the merge operator, for instance the merge of two BPA's, the verification problem of CLTL$^+$ becomes undecidable. Thus, the verification problem of CLTL$^+$ is undecidable for PA processes whereas it is decidable for PLTL$^+$. Moreover, we show that for PA processes, it suffices to consider the class of ω-star-free safety properties (i.e., safety properties that are expressible in LTL) to make the verification problem undecidable. Concerning BPP, we prove that the verification problem of CLTL$^+$ for this class of processes cannot be reduced to the validity problem in Presburger arithmetics. This result shows that the approach we used to establish our decidability results is not applicable in this case. However, this fact does not mean that the verification problem of CLTL$^+$ for BPP's is undecidable. Indeed, we show that there are at least some nonregular properties, corresponding to a simple form of constrained safety properties, for which the verification problem for BPP's is decidable although not reducible to the validity problem in Presburger arithmetics. A summary table is given in Section 6.

The remainder of the paper is organized as follows: In Section 2, we recall some basic definitions and introduce some notations. In Section 3 we define the syntax and the semantics of PA processes. In Section 4, we introduce the logic CLTL and its fragments, and discuss their expressiveness and their satisfiability problem. In Section 5, we expose our results concerning the verification problem of CLTL for PA processes. Concluding remarks are given in Section 6.

For lack of space, the details of the proofs are omitted and left to the full version of the paper.

2 Preliminaries

2.1 Presburger arithmetics

Presburger arithmetics is the first order logic of natural numbers with addition, substraction and the usual ordering. We recall hereafter the definition of this logic and its relation with semilinear sets.

Let $\mathcal{V}$ be a set of variables, and let $x, y, \ldots$ range over $\mathcal{V}$. Consider the set of terms defined by

$$t ::= 0 \mid 1 \mid x \mid t - t \mid t + t$$

Integer constants ($k \in \mathbb{Z}$) and multiplication by constants (kt) can be introduced as abbreviations. Then, the set of Presburger formulas is defined by

$$f ::= t \leq t \mid \neg f \mid f \vee f \mid \exists x.\, f$$

Classical abbreviations can be used like boolean connectives as conjunction ($\wedge$), implication ($\Rightarrow$) and equivalence ($\Leftrightarrow$) as well as universal quantification ($\forall$). The semantics of these formulas is defined in the standard way. Given a formula f with free variables $x_1, \ldots, x_n$, and a valuation $E : \mathcal{V} \to \mathbb{N}$, we say that E satisfies f, and write $E \vdash f$, if the evaluation of f under E is true. We say that a formula f is valid if every valuation satisfies f. It is well known that the satisfiability problem for Presburger formulas is decidable, and hence, so is for the validity problem.

A *linear* set is a subset of $\mathbb{N}^n$ of the form $\{\vec{v}+k_1\vec{u}_1+\cdots+k_m\vec{u}_m \ :\ k_1,\cdots,k_m \in \mathbb{N}\}$ where $n > 0$ and $\vec{v}, \vec{u}_1, \cdots, \vec{u}_m \in \mathbb{N}^n$. A *semilinear* set is a finite union of linear sets. It can be shown that any Presburger formula with n free variables defines a semilinear set, and vice versa.

2.2 Sequences

Let Σ be a finite alphabet. We denote by Σ^* (resp. Σ^ω) the set of finite (resp. infinite) sequences over Σ. Let $\Sigma^\infty = \Sigma^* \cup \Sigma^\omega$.

Given a sequence $\sigma \in \Sigma^\infty$, $|\sigma| \in \{0, 1, \ldots, \omega\}$ denotes the length of σ. Let ϵ denote the empty sequence, i.e., the sequence of length 0. Let $\Sigma^+ = \Sigma^* - \{\epsilon\}$.

For every $a \in \Sigma$, $|\sigma|_a$ is the number of occurrences of a in σ. We write $a \in \sigma$ to denote the fact that a appears in the sequence σ. Given $\Sigma' \subseteq \Sigma$, we denote by $\sigma|_{\Sigma'}$ the projection of σ on the subalphabet Σ'. For every $i \in \mathbb{N}$ such that $1 \leq i \leq |\sigma|$, $\sigma(i)$ denotes the i^{th} element of σ, and for every $j \in \mathbb{N}$ such that $i \leq j \leq |\sigma|$, $\sigma(i,j)$ denotes the subsequence $\sigma(i)\cdots\sigma(j)$ (with $\sigma(i,i) = \sigma(i)$). We denote by $Pref(\sigma) = \{\sigma(1,i) \ :\ 1 \leq i \leq |\sigma|\}$. For a set of sequences $L \subseteq \Sigma^\infty$, we define $Pref(L) = \bigcup\{Pref(\sigma) \ :\ \sigma \in L\}$.

Now, let $\Sigma = \{a_1, \cdots, a_n\}$. Given a finite sequence $\sigma \in \Sigma^*$, we define the vector $[\sigma] = (|\sigma|_{a_1}, \cdots, |\sigma|_{a_n}) \in \mathbb{N}^n$. Then, given a set of sequences $L \subseteq \Sigma^*$, we denote by $[L]$ the set $\{[\sigma] \ :\ \sigma \in L\}$.

2.3 Kripke structures

A Kripke structure over the alphabet Σ (KS for short) is a tuple $K = (\Sigma, S, \Pi, R)$ where S is a countable set of states, $\Pi : S \to \Sigma$ is a labelling function and $R \subseteq S \times S$ is a transition relation.

We write $s \to_R s'$ to denote the fact that $(s, s') \in R$. We write $s \not\to_R$ when there is no state s' such that $(s, s') \in R$. We say that K is *finite-branching* if for each state $s \in S$, the set $\{s' \ :\ s \to_R s'\}$ is finite.

An *infinite computation sequence* of K from a state s is a sequence $s_1 s_2 \cdots$ such that $s = s_1$ and $\forall i \geq 1.\ s_i \to_R s_{i+1}$. A *finite computation sequence* of K starting from s is a sequence $s_1 \cdots s_n$ such that $s = s_1$, $\forall i.\ 1 \leq i < n.\ s_i \to_R s_{i+1}$, and $s_n \not\to_R$. We denote by $\mathcal{C}(K, s)$ the set of finite and infinite computation sequences of K starting from s.

Given a computation sequence $\rho \in S^\infty$, the *trace* of ρ is the sequence in Σ^∞ defined by $\Pi(\rho) = \Pi(\rho(1)) \cdot \Pi(\rho(2)) \cdots$. Given a state s, we denote by $T(K, s)$ the set of traces of the computation sequences starting from s. We say that two states s and s' are *trace equivalent* iff $T(K, s) = T(K, s')$.

3 Process Algebra

We present in this section the class of processes for which we consider the verification problem. These processes are described in the process algebra PA [BK88] which subsumes the algebras BPA [BBK87] and BPP [Chr93]. We give the syntax of PA processes and define their operational semantics. Then, we introduce the notion of PA processes in *normal form* which is important for our decision technique, and we discuss briefly the expressiveness of PA.

First of all, let us give the syntax and the semantics of the PA processes. Our definition of the operational semantics of PA processes is done in a slightly different style from that usually adopted. Indeed, we consider that an atomic process is characterized by a set of atomic propositions instead of an atomic action label and we define an operational semantics of PA's by means of Kripke structures (state-labelled graphs) instead of labelled transition systems (edge-labelled graphs). We choose this semantics since our aim is to consider PA's as models for the temporal logic CLTL introduced in the next section which is interpreted on traces of KS's (sequences of state labels).

Let *Prop* be a finite set of *atomic propositions* and $\Sigma = 2^{Prop}$. We call the elements of Σ *state labels*. We consider a set of *process variables Var*. We use $a, b, \ldots$ to range over elements of Σ, and capital letters $X, Y, Z, \ldots$ (possibly with subscripts) to range over elements of *Var*.

Consider the set of terms $\mathcal{T}$ defined by the following grammar:

$$t ::= \mathbf{0} \mid a \mid X \mid t + t \mid t \cdot t \mid t \| t$$

Intuitively, $\mathbf{0}$ represents the idle process, a is an atomic process, the operator "+" stands for nondeterministic choice, the operator "·" is the sequential composition, "$\|$" is the parallel composition (merge operator). In the sequel, we identify the terms $\mathbf{0} \cdot t$, $t \cdot \mathbf{0}$, $\mathbf{0} + t$, $t + \mathbf{0}$, $\mathbf{0} \| t$, and $t \| \mathbf{0}$ with t, for any term t.

A PA process is defined by a finite family of recursive equations $\Delta = \{X_i \mathrel{\hat{=}} t_i \ : \ 1 \leq i \leq n\}$ where all the X_i's are distinct and all the variables occurring in the terms t_i's are in the (finite) set $Var_\Delta = \{X_1, \ldots, X_n\}$.

The algebras BPA and BPP are fragments of PA. The BPA's are defined by considering only terms of the form:

$$t ::= \mathbf{0} \mid a \mid X \mid t + t \mid t \cdot t$$

whereas the BPP's are obtained by considering terms of the form:

$$t ::= \mathbf{0} \mid X \mid a \cdot t \mid t + t \mid t \| t$$

A term $t \in \mathcal{T}$ is *guarded* iff every variable occurrence in t is within the scope of an atomic action $a \in \Sigma$. A process in PA $\Delta = \{X_i \mathrel{\hat{=}} t_i \ : \ 1 \leq i \leq n\}$ is guarded iff every term t_i is guarded. From now on, we consider only guarded processes.

We define the operational semantics of a process in PA by associating with each family Δ a Kripke structure $\mathcal{K}_\Delta$ representing its computation graph. This transition system is given by $\mathcal{K}_\Delta = (\Sigma, S_\Delta, \Pi_\Delta, R_\Delta)$ where

- $S_\Delta = \Sigma \times \mathcal{T}$,
- $\forall \langle a, t \rangle \in S_\Delta . \, \Pi_\Delta(\langle a, t \rangle) = a$,
- $R_\Delta \subseteq S_\Delta^2$ is the smallest relation such that
  1. $\langle a, a' \rangle \rightarrow_{R_\Delta} \langle a', \mathbf{0} \rangle$,
  2. "$X \mathrel{\hat{=}} t$" $\in \Delta$ and $\langle a, t \rangle \rightarrow_{R_\Delta} \langle a', t' \rangle$ implies $\langle a, X \rangle \rightarrow_{R_\Delta} \langle a', t' \rangle$,
  3. $\langle a, t_1 \rangle \rightarrow_{R_\Delta} \langle a', t'_1 \rangle$ implies $\langle a, t_1 + t_2 \rangle \rightarrow_{R_\Delta} \langle a', t'_1 \rangle$ and $\langle a, t_2 + t_1 \rangle \rightarrow_{R_\Delta} \langle a', t'_1 \rangle$,
  4. $\langle a, t_1 \rangle \rightarrow_{R_\Delta} \langle a', t'_1 \rangle$ implies $\langle a, t_1 \cdot t_2 \rangle \rightarrow_{R_\Delta} \langle a', t'_1 \cdot t_2 \rangle$,
  5. $\langle a, t_1 \rangle \rightarrow_{R_\Delta} \langle a', t'_1 \rangle$ implies $\langle a, t_1 \| t_2 \rangle \rightarrow_{R_\Delta} \langle a', t'_1 \| t_2 \rangle$ and $\langle a, t_2 \| t_1 \rangle \rightarrow_{R_\Delta} \langle a', t_2 \| t'_1 \rangle$.

It is easy to see that every guarded process in PA generates a finite-branching kripke structure.

Now, we introduce a special syntactical form of PA processes which is a generalization of the well-known Greibach normal form used in formal language theory for context-free languages (see [Har78]). For this, let us consider the set $\mathcal{T}(Var)$ of terms that are either $\mathbf{0}$ or constructed using only process variables, and the three operators "·", and "$\|$", i.e., terms of the form:

$$\tau ::= \mathbf{0} \mid X \mid \tau \cdot \tau \mid \tau \| \tau$$

We use Greek letters $\alpha, \beta, \tau, \ldots$ to range over elements of $\mathcal{T}(Var)$. Then, a PA process $\Delta = \{X_i \mathrel{\hat{=}} t_i \ : \ 1 \leq i \leq n\}$ is in *normal form*, if every term t_i is either $\mathbf{0}$ or of the form $a_1^i \cdot \tau_1^i + \cdots + a_{m_i}^i \cdot \tau_{m_i}^i$ such that no variable X_k occurring in some τ_j^i is defined to be an idle process, i.e., $X_k \mathrel{\hat{=}} \mathbf{0} \in \Delta$. Notice that for a BPA (resp. BPP) process, the τ_j^i's are constructed by means of the operator "·" (resp. "$\|$") only.

It can be shown that guarded BPA's, BPP's, and PA's can be transformed into their respective normal forms preserving trace equivalence. This is due to the fact that "·" and "$\|$" are left and right distributive w.r.t. "+" for this equivalence.

Now, let us discuss the expressive power of the classes BPA, BPP and PA by considering the classes of languages they generate. First of all, it is clear that both BPA and BPP subsume the class of *regular processes* (RP for short), i.e., finite-state processes. Moreover, it is well-known that the BPA processes generate context-free languages [BBK87], and it has been shown that the BPP's are not comparable with the BPA's [Chr93]. For instance, the language $\{a^n b^n \ : \ n > 0\}$, which is definable by the BPA:

$$X \hat{=} a \cdot X \cdot b + a \cdot b$$

cannot be generated by any BPP whereas the language $\{\sigma \in \Sigma^+ \ : \ |\sigma|_a = |\sigma|_b = |\sigma|_c\}$, which is obviously non context-free, can be generated by the BPP:

$$X \hat{=} a \cdot (b||c||X + b||c) + b \cdot (a||c||X + a||c) + c \cdot (a||b||X + a||b)$$

Then, PA is strictly more expressive than both BPA and BPP. Furthermore, PA is strictly more expressive than the union of BPA and BPP (BPA $\uplus$ BPP), i.e., PA processes can generate languages that can be generated neither by BPA's nor by BPP's. To see this, consider the following language, which can be defined in PA as a parallel composition of two BPA's: $\{\sigma \in \Sigma^+ \ : \ \sigma|_{\{a,b\}} \in \{a^n b^n \ : \ n > 0\}$ and $\sigma|_{\{c,d\}} \in \{c^n d^n \ : \ n > 0\}\}$. Actually, it is easy to see (from the last example for instance) that the class BPA is not closed under the "$||$" operator, and it has been also proved (see [Chr93]) that BPP is not closed under the "$\cdot$" operator. We denote by $||^k$-BPA (resp. $\odot^k$-BPP) the class of processes obtained as parallel (resp. sequential) compositions of k BPA's (resp. BPP's); let $||$-BPA$= \bigcup_k ||^k$-BPA and $\odot$-BPP$= \bigcup_k \odot^k$-BPP.

We can prove that every PA process can be simulated by some linearly bounded automaton, hence the PA processes generate context-sensitive languages (CSL's). On the other hand, we can prove that the context-sensitive language $\{a^n b^n c^n \ : \ n > 0\}$ cannot be defined by any PA process.

Finally, we observe that BPP's can be defined in terms of Petri Nets (PN's) whereas BPA is not comparable with PN [Chr93], and hence, so is for the class of PA processes. Also, since PN is closed under sequential composition, $\odot$-BPP is included in PN.

Then, we obtain the following picture where edges represent strict inclusions between the classes of languages defined by the considered classes of processes:

$$\begin{tikzcd}
 & \text{BPA} \arrow[rr] \arrow[dr] & & \|\text{-BPA} \arrow[dr] & & \\
\text{RP} \arrow[ur] \arrow[dr] & & \text{BPA} \uplus \text{BPP} \arrow[rr] & & \text{PA} \arrow[r] & \text{CSL} \\
 & \text{BPP} \arrow[rr] \arrow[ur] & & \odot\text{-BPP} \arrow[ur] \arrow[r] & \text{PN} \arrow[ur] &
\end{tikzcd}$$

4 Constrained LTL

We introduce in this section the logic Constrained LTL (CLTL) and discuss its expressiveness and its satisfiability problem.

4.1 Definition

CLTL is an extension of the linear-time propositional temporal logic LTL [Pnu77] with the ability of constraining the *pattern* of computations, i.e., the order of appearance of states, or constraining the *numbers of occurrences* of states in computations. Constraints on patterns are expressed using finite-state automata, whereas constraints on numbers of occurrences are expressed using Presburger arithmetics.

Before giving the syntax of CLTL, let us recall and introduce some notations. Recall that *Prop* is a finite set of atomic proposition and that $\Sigma = 2^{Prop}$. Recall also that $\mathcal{V}$ is a set of integer valued variables. We use letters $P, Q, \ldots$ to range over elements of *Prop*, letters $x, y, \ldots$ to range over variables in $\mathcal{V}$ and $f, g, \ldots$ to range over Presburger formulas. Let us introduce a set $\mathcal{W}$ of *position variables.* We use letters $u, v, \ldots$ to range over $\mathcal{W}$. Finally, we use letters $A, B, \ldots$ to range over deterministic finite-state automata over Σ. Given an automaton A, we denote by $L(A)$ the set of sequences accepted by A (the language of A), and by $\overline{A}$ an automaton accepting the complement of $L(A)$ in Σ^*. Now, consider the set of *state formulas* given by:

$$\pi ::= P \mid \neg\pi \mid \pi \vee \pi$$

The set of formulas of CLTL is defined by:

$$\begin{array}{rcl} \varphi & ::= & \tilde{\exists} x.\varphi \mid [x : \pi].\varphi \mid \downarrow\!\! u.\varphi \mid f \mid A^u \mid \\ & & P \mid \neg\varphi \mid \varphi \vee \varphi \mid \bigcirc\varphi \mid \varphi\mathcal{U}\varphi \end{array}$$

We consider as abbreviations the usual boolean connectives as conjunction ($\wedge$), implication ($\Rightarrow$) and equivalence ($\Leftrightarrow$). In addition, we use the universal quantification $\tilde{\forall} x.\varphi = \neg\tilde{\exists} x.\neg\varphi$ and the following abbreviations: $\Diamond\varphi = \mathsf{true}\mathcal{U}\varphi$, and $\Box\varphi = \neg\Diamond\neg\varphi$. We write $[x_1, \ldots, x_n : \pi_1, \ldots, \pi_n].\varphi$ or $[x_i : \pi_i]_{i=1}^n.\varphi$ for $[x_1 : \pi_1].\cdots[x_n : \pi_n].\varphi$.

CLTL formulas are interpreted on sequences of state labels (sequences in Σ^∞).

The operators $\bigcirc, \mathcal{U}, \Diamond$, and $\Box$, are the usual *next*, *until*, *eventually*, and *always* operators of LTL.

The operator $\tilde{\exists}$ corresponds to the usual existential quantification over integers. We distinguish (even syntactically) between the CLTL operator $\tilde{\exists}$ and the Presburger existential quantifier $\exists$ that may be used locally in some formula f.

The Presburger formulas f are used to express constraints on the numbers of occurrences of states satisfy-

$$
\begin{array}{lll}
\langle\sigma,i\rangle \models_{(E,\theta,\gamma)} P & \text{iff} & P \in \sigma(i) \\
\langle\sigma,i\rangle \models_{(E,\theta,\gamma)} \neg\varphi & \text{iff} & \langle\sigma,i\rangle \not\models_{(E,\theta,\gamma)} \varphi \\
\langle\sigma,i\rangle \models_{(E,\theta,\gamma)} \varphi_1 \vee \varphi_2 & \text{iff} & \langle\sigma,i\rangle \models_{(E,\theta,\gamma)} \varphi_1 \text{ or } \langle\sigma,i\rangle \models_{(E,\theta,\gamma)} \varphi_2 \\
\langle\sigma,i\rangle \models_{(E,\theta,\gamma)} \widetilde{\exists} x.\varphi & \text{iff} & \exists k \in I\!N.\ \langle\sigma,i\rangle \models_{(E',\theta,\gamma)} \varphi \text{ where } E' = E[x \leftarrow k] \\
\langle\sigma,i\rangle \models_{(E,\theta,\gamma)} [x:\pi].\varphi & \text{iff} & \langle\sigma,i\rangle \models_{(E',\theta,\gamma')} \varphi \ \text{ where } E' = E[x \leftarrow 0] \text{ and } \gamma' = \gamma[x \leftarrow \pi] \\
\langle\sigma,i\rangle \models_{(E,\theta,\gamma)} f & \text{iff} & E' \vdash f \ \text{ where } E' = E[x \leftarrow E(x) + (\text{if } \langle\sigma,i\rangle \models \gamma(x) \text{ then } 1 \text{ else } 0)]_{x\in\mathcal{D}(\gamma)} \\
\langle\sigma,i\rangle \models_{(E,\theta,\gamma)} \downarrow\!u.\varphi & \text{iff} & \langle\sigma,i\rangle \models_{(E,\theta',\gamma)} \varphi \ \text{ where } \ \theta' = \theta[u \leftarrow i] \\
\langle\sigma,i\rangle \models_{(E,\theta,\gamma)} A^u & \text{iff} & \sigma(\theta(u),i) \in L(A) \\
\langle\sigma,i\rangle \models_{(E,\theta,\gamma)} \bigcirc\varphi & \text{iff} & i < |\sigma| \text{ and } \langle\sigma,i+1\rangle \models_{(E',\theta,\gamma)} \varphi, \ \text{ where} \\
 & & E' = E[x \leftarrow E(x) + (\text{if } \langle\sigma,i\rangle \models \gamma(x) \text{ then } 1 \text{ else } 0)]_{x\in\mathcal{D}(\gamma)} \\
\langle\sigma,i\rangle \models_{(E,\theta,\gamma)} \varphi_1 \mathcal{U} \varphi_2 & \text{iff} & \exists j \in I\!N.\ i \le j \le |\sigma|.\ \langle\sigma,j\rangle \models_{(E'(j),\theta,\gamma)} \varphi_2 \text{ and} \\
 & & \forall k \in I\!N.\ i \le k < j.\ \langle\sigma,k\rangle \models_{(E'(k),\theta,\gamma)} \varphi_1, \ \text{ where} \\
 & & E'(m) = E[x \leftarrow E(x) + |\{\ell \in \{i,\ldots,m\} \ : \ \langle\sigma,\ell\rangle \models \gamma(x)\}|]_{x\in\mathcal{D}(\gamma)}.
\end{array}
$$

Table 1: Definition of the satisfaction relation

ing some given state formulas. We call these formulas *occurrence constraints.*

In the formula $[x:\pi].\varphi$, the variable x is associated with the state formula π, and then, starting from the current state, x represents the number of occurrences of states satisfying π. The variable x can be used in the occurrence constraints appearing in φ. For instance, the formula $\phi_1 = [x,y:\pi_1,\pi_2].\Box(P \Rightarrow (x \le y))$ expresses the fact that from the current state until every state where P holds, the number of states satisfying π_2 is greater than the number of state satisfying π_1. From now on, we refer to the variables x that are associated with state formulas as *occurrence variables.*

In the formula $\downarrow\!u.\varphi$, the current state is associated in some manner with the position variable u. Then, in φ, subformulas of the form A^u are used to express the fact that the trace of the computation since the state at position u is accepted by the automaton A. For instance, the formula $\phi_2 = \downarrow\!u.[x,y:\pi_1,\pi_2].\Box(A^u \Rightarrow (x \le y))$ expresses the fact that in every computation starting from the current state and which is accepted by A, the number of states satisfying π_2 is greater or equal than the number of states satisfying π_1. We call formulas of the form A^u *pattern constraints.*

In the formula ϕ_2, the construction "$[x:\pi_1]$" (resp. "$\downarrow\!u$") binds the variable x (resp. u) in the subformula $\Box(A^u \Rightarrow (x \le y))$. So, a variable $x \in \mathcal{V}$ may be bound by either the quantifier $\exists$, or by the quantifier $\widetilde{\exists}$, or by the construction "$[x:\pi]$", and a position variable $u \in \mathcal{W}$ can be bound by the construction "$\downarrow\!u$". We suppose without loss of generality that each variable is bound at most once in a formula. Then, every variable appearing in some formula is either *bound* or *free.* A formula φ is *closed* if all the variables occurring in it are bound, otherwise φ is *open.*

The formal semantics of CLTL is defined using a satisfaction relation $\models$ between sequences in Σ^∞, integers (indexes on the sequences), and formulas. Since formulas may be open, the satisfaction relation between a sequence σ, an index i, and a formula φ, is defined w.r.t. a *valuation* E of the variables in $\mathcal{V}$, and a *position association* θ that associates with each position variable u the index in σ (some integer less than i) where u has been introduced. The valuation E changes according to the satisfaction at the visited states of the state formulas associated with occurrence variables. A *state formula association* γ is used to record for each occurrence variable the state formula which is associated with it. To define the updating of E, θ, and γ, let us introduce some notations: Let F stands for E, θ, or γ. Then, we denote by $\mathcal{D}(F)$ (resp. $\mathcal{I}(F)$) the domain (resp. image) of F. We denote simply by $\emptyset$ the function F such that $\mathcal{D}(F) = \mathcal{I}(F) = \emptyset$. Finally, we denote by $F[z \leftarrow e]$ the function F' which associates the value e with the variable z and coincides with F on all the other variables.

Now, let $\sigma \in \Sigma^\infty$. Then, for every index i such that $1 \le i \le |\sigma|$, state formula association γ, position association θ (such that for every x in the domain of θ, $1 \le \theta(x) \le i$), valuation E, and CLTL formula φ, we define the meaning of $\langle\sigma,i\rangle \models_{(E,\theta,\gamma)} \varphi$ inductively on the structure of φ; the definition is given in Table 1. Notice that when φ is a closed formula, we have $\langle\sigma,i\rangle \models_{(E,\theta,\gamma)} \varphi$ iff $\langle\sigma,i\rangle \models_{(\emptyset,\emptyset,\emptyset)} \varphi$, and in this case we write simply $\langle\sigma,i\rangle \models \varphi$.

We define a satisfaction relation between states of a Kripke structure and closed formulas in CLTL. Let $K = (\Sigma, S, \Pi, R)$ be a KS and s a state in S. Then, we say that s *satisfies* a CLTL closed formula φ, and write $s \models \varphi$, iff $\forall \rho \in \mathcal{C}(K,s).\ \langle\Pi(\rho),1\rangle \models \varphi$.

4.2 Fragments

We introduce fragments of CLTL that are considered in the next section for the decidability issue of the verification problem.

The first fragment we define is a positive fragment called CLTL$^+$. It is obtained by the following definition:

$$\begin{array}{lll} \varphi & ::= & \widetilde{\forall} x.\varphi \mid \downarrow\!\!u.\varphi \mid [x:\pi].\varphi \mid f \mid \psi \mid \\ & & \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \bigcirc\varphi \mid \varphi \mathcal{U} \psi \mid \Box\varphi \\ \psi & ::= & P \mid A^u \mid \neg\psi \mid \psi \vee \psi \end{array}$$

We define also the logic PLTL (Presburger LTL) which corresponds to CLTL without pattern constraints (formulas of the form $\downarrow\!\!u.\ \varphi$ and A^u). Similarly, we define the positive fragment PLTL$^+$ as CLTL$^+$ without pattern constraints. We consider also the logic ALTL (Automata + LTL) which corresponds to CLTL without occurrence constraints (formulas of the form $\widetilde{\exists} x.\ \varphi$, $[x : \pi].\ \varphi$, and f), as well as its fragment ALTL$^+$ corresponding to CLTL$^+$ without occurrence constraints.

4.3 Expressiveness

Let us discuss the expressiveness of CLTL and its fragments. First of all, it can be observed that the logic ALTL is as expressive as ω-regular automata. Indeed, by the McNaughton's characterization of ω-regular languages [Tho90], we can define every ω-regular language using formulas of the form

$$\downarrow\!\!u.\ \bigvee_{i=1}^{n}(\Box\Diamond A_i^u \wedge \Diamond\Box B_i^u)$$

Thus, ALTL has the maximal expressiveness as a "*regular*" logic. However, formulas of the form $\Diamond\Box$ cannot be used in the fragment ALTL$^+$, which weakens its expressiveness. Nevertheless, we can define in ALTL$^+$ all the ω-regular languages definable by deterministic Büchi automata, using formulas of the form $\downarrow\!\!u.\ \Box\Diamond A^u$. Thus, according to the classification of properties introduced in [MP90], CLTL$^+$ can express all the ω-regular recurrence properties, including safety properties ($\downarrow\!\!u.\ \Box A^u$), guarantee properties ($\downarrow\!\!u.\ \Diamond A^u$), and obligation properties ($\downarrow\!\!u.\ \bigwedge_{i=1}^{n}(\Box A_i^u \Rightarrow \Diamond B_i^u)$).

Now, let us present some examples of nonregular properties expressible in PLTL$^+$ or CLTL$^+$. As a first example, consider the formula (1) given in the introduction. This formula is in PLTL$^+$, and it expresses the fact that every sequence between any pair of BEGIN and END is in the language of well-balanced parentheses (semi-Dyck set) with REQ (resp. ACK) as a left (resp. right) parenthesis. We can express the stronger property that every sequence between two consecutive BEGIN and END is in the language $\{\text{REQ}^n \cdot \text{ACK}^n\ :\ n \geq 1\}$. This is done by the conjunction of (1) with the following PLTL$^+$ formula:

$$\begin{array}{l} \widetilde{\forall} n.\ \Box(\text{BEGIN} \Rightarrow \\ \quad [x, y, z : \text{REQ}, \text{ACK}, \text{END}]. \\ \quad \Box((\text{ACK} \wedge (x = n) \wedge (y = 1)) \Rightarrow \\ \qquad \Box((\text{END} \wedge (z = 1)) \Rightarrow (x = n) \wedge (y = n)))) \quad (2) \end{array}$$

Then, we can characterize in PLTL a large class of nonregular languages. These languages can be context-free as in (1), and (2), but also context-sensitive using a conjunction of more than two occurrence constraints concerning different sets of occurrence variables. For instance, we can consider languages as $\{a_1^n \cdots a_k^n\ :\ n \geq 1\}$, or $\{a^n \cdot b^m \cdot a^n \cdot b^m\ :\ n, m \geq 1\}$.

It is interesting to see that PLTL$^+$ allows to express very easily some ω-regular properties that are not expressible in LTL. For example the property that P is true at each *even* rank of every computation, which is a non ω-star-free property [Wol83, Tho79], can be expressed in PLTL$^+$ by:

$$[x : \textsf{true}].\ \Box((\exists k.\ 2k = x) \Rightarrow P) \qquad (3)$$

The properties corresponding to the formulas (1), (2), and (3) can be expressed differently using a combination of pattern and occurrence constraints. For instance the formula (2) above is equivalent to the more elegant CLTL$^+$ formula:

$$\downarrow\!\!u.\ [x, y : \text{REQ}, \text{ACK}].\Box(A^u \Rightarrow (x = y)) \qquad (4)$$

where A is an automaton such that $L(A) = \Sigma^* \cdot \text{BEGIN} \cdot \text{REQ}^+ \cdot \text{ACK}^+ \cdot \text{END}$.

Then, the use of automata in combination with occurrence constraints provides a concise and natural way to express nonregular properties. In addition, automata allow to enrich significantly the expressive power of PLTL$^+$. In fact, we can deduce from our decidability and undecidability results given in Section 5 that CLTL$^+$ is strictly more expressive than PLTL$^+$, and moreover, that PLTL$^+$ cannot express all the ω-star-free safety properties (safety properties expressible in LTL) whereas, as we have seen above, CLTL$^+$ can express (in particular) all ω-regular safety properties (see the comments after Corollary 5.3, Corollary 5.4, and Theorem 5.5).

Furthermore, the combination of pattern constraints and occurrence constraints allows to define new classes of nonregular properties. For instance, we can describe in CLTL$^+$ *constrained safety properties* that correspond to formulas of the form

$$\downarrow u.\ [x_k : \pi_k]_{k=1}^{\ell} \cdot \bigvee_{i=1}^{n} \bigwedge_{j=1}^{m} \Box(A^u_{(i,j)} \Rightarrow f_{(i,j)}(x_1, \cdots, x_\ell)).$$

Following the classification of [MP90], we can define further classes of properties corresponding to CLTL formulas involving automata and Presburger formulas, like for instance *general constrained guarantee properties*, *recurrence properties*, and so on. However, such a general hierarchy of properties cannot be described fully in CLTL$^+$ since it is a positive fragment and it does not allow $\Diamond$ formulas with occurrence constraints.

4.4 The satisfiability problem

Finally, let us consider the satisfiability problem of CLTL and its fragments. We can prove that this problem is highly undecidable (Σ_1^1-complete) for CLTL as well as for CLTL$^+$, PLTL, and PLTL$^+$. Indeed, we show first that the satisfiability problem of CLTL can be expressed by a Σ_1^1 formula. Then, we prove that the satisfiability problem of PLTL$^+$ is Σ_1^1-hard. For that, we use the fact that the problem of deciding whether a given nondeterministic 2-counter machine has a recurring computation, i.e., a computation that visits infinitely often the starting location, has been shown in [HPS83] and then in [AH89] to be Σ_1^1-hard. We reduce this problem to the satisfiability problem of PLTL$^+$ formulas. For that, given a 2-counter machine M, we construct a formula of the form $([x_i, y_i : inc_i, dec_i]_{i=1}^2 \Box\varphi) \wedge \Box\Diamond P$ where φ encodes the transition relation of M (test to zero of each counter i is done by comparing the occurrence variables x_i and y_i), and P is a proposition corresponding to its starting location. Hence, we have the following undecidability results.

Theorem 4.1 *The satisfiability problems of CLTL, CLTL$^+$, PLTL, and PLTL$^+$, are Σ_1^1-complete. Consequently, the validity problems of the logics CLTL and PLTL are Π_1^1-complete.*

Notice that we cannot deduce from Theorem 4.1 any information concerning the validity problems of the fragments CLTL$^+$ et PLTL$^+$ since these fragments are positive (nonclosed under negation). Actually, we can prove that these validity problems are decidable (see Corollary 5.1 and Corollary 5.2).

5 The Verification Problem

We tackle in this section the decidability of the verification problems of CLTL and its fragments for processes in BPA, BPP, and PA.

Let us start by considering the verification problems of the full logics PLTL and CLTL. Since a closed formula φ is valid iff it is satisfied by the regular process generating Σ^∞, we obtain as a consequence of Theorem 4.1 the following undecidability result.

Theorem 5.1 *The verification problems for regular processes (in RP) w.r.t. PLTL and CLTL formulas are Π_1^1-complete.*

Thus, the verification problems of the logics PLTL and CLTL are highly undecidable for the classes BPA, BPP, and PA.

Recall that in the proof of Theorem 4.1, from which Theorem 5.1 is deduced, we have to consider the satisfiability of formulas of the form $\Box\varphi \wedge \Box\Diamond P$. That means that the verification problem for regular processes is highly undecidable when we consider formulas of the form $\Diamond\varphi \vee \Diamond\Box P$. Actually, we can prove that this problem is undecidable as soon as we consider formulas of the form $\Diamond f$. But in this case, the verification problem becomes semi-decidable (Σ_1^0-complete). Indeed, since the satisfaction of occurrence constraints depends only on the past, and since regular processes are finite-branching, the verification problem of $\Diamond f$ formulas for such processes is recursively enumerable (in Σ_1^0). On the other hand, we can reduce the halting problem of any 2-counter machine to the verification problem of a $\Diamond f$ formula, which shows the Σ_1^0-hardness.

Now, let us consider the fragment PLTL$^+$ (where formulas of the form $\Diamond f$ are forbidden). We prove that for PLTL$^+$, the verification problem is decidable for all guarded PA processes. For that, we use the fact that every guarded PA process can be transformed into normal form preserving trace equivalence, and that equivalent processes satisfy the same CLTL formulas, in order to concentrate on guarded PA's in normal form. Then, given a state s of some PA in normal form and a PLTL$^+$ closed formula φ, we construct a Presburger formula Ω which is valid if and only if $s \models \varphi$. This reduction is based on a similar technique to the one used in [BEH95] concerning a fragment of Presburger CTL. So, we have the following decidability result.

Theorem 5.2 *The verification problem for guarded PA's w.r.t. PLTL$^+$ closed formulas is decidable.*

Recall that a closed formula φ is valid iff it is satisfied by the regular process generating Σ^∞. Then, we obtain as a consequence of Theorem 5.2 the following result.

Corollary 5.1 *The validity problem of PLTL$^+$ closed formulas is decidable.*

Now, let us see the effect of adding pattern constraints, i.e., taking CLTL$^+$ instead of PLTL$^+$, on the decidability results above.

Consider first the class of guarded BPA processes. We prove that for this class of processes, the verification problem of CLTL$^+$ is decidable. For that, we show that this problem also is reducible to the validity problem in Presburger arithmetics. The reduction we use in this case is however technically different from that used in the proof of Theorem 5.2. So, we obtain the following decidability result.

Theorem 5.3 *The verification problem for guarded BPA's w.r.t. CLTL$^+$ closed formulas is decidable.*

Corollary 5.2 *The validity problem of CLTL$^+$ closed formulas is decidable.*

Theorem 5.3 says that the addition of pattern constraints to PLTL$^+$ does not alterate the decidability of the verification problem for BPA processes. However, this is no longer the case as soon as we consider the merge of two (nonregular) BPA processes, and very simple formulas involving pattern constraints (and without occurrence constraints), namely the ω-regular safety formulas, i.e., formulas of the form $\downarrow u.\Box A^u$. In fact, we can reduce the halting problem of deterministic 2-counter machines to the nonsatisfaction problem of ω-regular safety formulas by $||^2$-BPA processes; that means that the verification problem of ω-regular safety formulas for $||^2$-BPA's is Π_1^0-hard. The idea behind this reduction is, given a 2-counter machine M, to encode the behaviours of the two counters of M (incrementations, decrementations, tests to zero) using two BPA's Δ_1 and Δ_2 with distinct vocabularies, and to construct from the finite control graph of M a finite state deterministic automaton C which accepts sequences reaching some designated halting control node. Then, M halts iff the $||^2$-BPA obtained as the merge of Δ_1 and Δ_2 does not satisfy the formula $\downarrow u.\,\Box\overline{C}^u$. On the other hand, it is easy to see that the complement of the verification problem for finite-branching processes (in particular guarded PA processes) w.r.t. ω-regular safety properties is semi-decidable (in Σ_1^0). Then, we obtain the following undecidability results.

Proposition 5.1
The verification problem for guarded $||^2$-BPA's w.r.t. ω-regular safety formulas is Π_1^0-complete.

Corollary 5.3 *The verification problem for guarded PA processes w.r.t. CLTL$^+$ formulas is undecidable.*

We can deduce from Corollary 5.3 and Theorem 5.2 that CLTL$^+$ is strictly more expressive than PLTL$^+$. We can also deduce from Proposition 5.1 that PLTL$^+$ cannot express all the ω-regular safety properties.

Now, concerning the class of $||$-BPA processes, we can actually prove a stronger undecidability result than Proposition 5.1.

Proposition 5.2
The verification problem for guarded $||^3$-BPA's w.r.t. ω-star-free safety formulas is Π_1^0-complete.

The proof of Proposition 5.2 is also done by a reduction of the halting problem of deterministic 2-counter machines. The idea here is that given a 2-counter machine M, we encode its counters as well as its control graph by three BPA's Δ_1, Δ_2, and Δ_3, and we force the synchronizations between these processes by considering the intersection with a finite-state deterministic automaton D which is counter-free, i.e., $L(D)$ as well as $L(\overline{D})$ are star-free languages. Then, M halts iff the merge of Δ_1, Δ_2, and Δ_3 does not satisfy the formula $\downarrow u.\,\Box\overline{D}^u$.

Since ω-star-free safety formulas are definable in LTL [Tho79, Wol83, LPZ85, MP90], we deduce from Proposition 5.2 the following undecidability results.

Theorem 5.4 *The verification problem of guarded $||$-BPA's w.r.t. LTL is undecidable.*

Corollary 5.4 *The verification problem of guarded PA processes w.r.t. LTL is undecidable.*

Thus, concerning PA processes, its suffices to consider LTL formulas, even LTL safety properties, to get an undecidable verification problem. By contrast, the verification problems of BPA's and BPP's w.r.t. ALTL (or equivalently the linear-time μ-calculus) are decidable. Indeed, for BPA processes, this decidability result can be deduced from the fact that the inclusion problem between ω-context-free grammars and ω-regular grammars is decidable [CG77]. As for BPP's, this result has been shown in [Esp93] (actually for all Petri nets).

We can deduce from Corollary 5.4 and Theorem 5.2 that PLTL$^+$ cannot express all ω-star-free safety properties (this improves what we have said after Proposition 5.1 and Corollary 5.3).

Now, let us consider the verification problem of BPP's w.r.t. CLTL$^+$. We show that this problem cannot be reduced to the validity problem in Presburger arithmetics. For that, it suffices again to consider ω-regular safety formulas. Indeed, let Δ be a BPP, $s \in \mathcal{K}_\Delta$, and $\varphi = \downarrow u.\Box A^u$. Then, since $s \models \varphi$ iff $Pref(T(\mathcal{K}_\Delta, s)) \cap L(\overline{A}) = \emptyset$, the problem $s \models \varphi$ is reducible to the (non)satisfiability problem in Presburger arithmetics if and only if the set

$\models$	LTL	ALTL	PLTL	CLTL	PLTL$^+$	CLTL$^+$
RP	yes	yes	no	no	yes	yes
BPA	yes	yes	no	no	yes	yes
$\|\|$-BPA	no	no	no	no	yes	no
BPP	yes	yes	no	no	yes	?
$\odot$-BPP	yes	yes	no	no	yes	?
PA	no	no	no	no	yes	no

Table 2: Decidability of the verification problem

$[Pref(T(\mathcal{K}_\Delta, s)) \cap L(\overline{A})]$ is semilinear (see the definition of $[\cdot]$ in Section 2.2). However, we can prove the following fact.

Theorem 5.5 *There exists a guarded BPP Δ with 3 variables, a deterministic finite-state automaton B with 2 states, and s a state in $\mathcal{S}_\Delta$ such that $[Pref(T(\mathcal{K}_\Delta, s)) \cap L(B)]$ is not semilinear.*

The proof of Theorem 5.5 is inspired by the one given in [HP79] to show that 3-dim vector addition systems with states (VASS's) can have nonsemilinear sets of reachable states.

Notice that, since the verification problem of BPP's w.r.t. PLTL$^+$ is reducible to the validity problem in Presburger arithmetics (Theorem 5.2), we can deduce from Theorem 5.5 that CLTL$^+$ with only two-state automata is strictly more expressive than PLTL$^+$. Actually, to gain in expressiveness using pattern constraints, the number of states of the used automata is not relevant by itself, but rather the number of states appearing in their loops. Indeed, we can show that CLTL$^+$ with automata such that all their loops are self-loops is equivalent to PLTL$^+$.

Moreover, the automaton B mentioned in Theorem 5.5 defines actually a star-free language. That means that the verification problem of BPP's w.r.t. LTL formulas is not reducible to the validity problem in Presburger arithmetics. However, Theorem 5.5 does not mean that the verification problem of BPP's w.r.t. LTL or ω-regular safety formulas is undecidable. Actually, this problem is decidable since the verification problem of PN's w.r.t. ω-regular properties is decidable [Esp93]. Using this fact together with Theorem 5.2, we deduce the following result concerning $\odot$-BPP's and simple constrained safety properties, i.e., formulas of the form $\downarrow u.\ [x_i : \pi_i]_i^n.\ \Box(A^u \wedge f(x_1, \cdots, x_n))$.

Theorem 5.6 *The verification problem for guarded $\odot$-BPP's w.r.t. simple constrained safety formula is decidable.*

6 Conclusion

We have addressed in this paper the verification problem of infinite-state processes w.r.t. nonregular properties, stated in a fairly general framework. We have considered the class of processes definable in the algebras BPA and BPP, as well as in the more general algebra PA which provides both sequential and parallel (merge) composition. On the other hand, we have considered properties definable in the new temporal logic CLTL which extends the logic LTL by the ability of expressing two kinds of constraints on traces: constraints on numbers of occurrences of state properties using Presburger formulas, and constraints on the ordering in which state properties appear using finite-state automata.

The results we present show for which kinds of processes and which kinds of properties in this framework the verification problem is decidable, and when this problem becomes undecidable. These results are summarized in Table 2. We prove our decidability results using reductions to the validity problem in Presburger arithmetics. We apply this approach for the verification problems concerning PA's w.r.t. PLTL$^+$ and BPA's w.r.t. CLTL$^+$. On the other hand, we show that the verification problem of BPP's w.r.t. CLTL$^+$ cannot be reduced to the validity problem in Presburger arithmetics, which indicates its hardness. Nevertheless, the question whether this problem is decidable remains open.

As far as we know, our work is the first one that provides decision procedures for the verification problem of infinite-state processes w.r.t. *nonregular* properties. Indeed, the existing results on the verification of infinite-state processes consider properties expressible in the traditional temporal logics or μ-calculi, or by finite ω-automata [BS92, Esp93]. On the other hand, the existing works on logics expressing nonregular properties [HPS83, HR90] consider only the satisfiability problem and do not address the verification problem as we do.

References

[AH89] R. Alur and T. Henzinger. A Really Temporal Logic. In *FOCS'89*. IEEE, 1989.

[AJ93] P. Abdulla and B. Jonsson. Verifying Programs with Unreliable Channels. In *LICS'93*. IEEE, 1993.

[BBK87] J.C.M. Baeten, J.A. Bergstra, and J.W. Klop. Decidability of Bisimulation Equivalence for Processes Generating Context-Free Languages. Tech. Rep. CS-R8632, 1987. CWI.

[BEH95] A. Bouajjani, R. Echahed, and P. Habermehl. Verifying Infinite State Processes with Sequential and Parallel Composition. In *POPL'95*. ACM, 1995.

[BER94] A. Bouajjani, R. Echahed, and R. Robbana. Verification of Nonregular Temporal Properties for Context-Free Processes. In *CONCUR'94*. LNCS 836, 1994.

[BK88] J.A. Bergstra and J.W. Klop. Process Theory based on Bisimulation Semantics. In *REX School/Workshop on Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, 1988. LNCS 354.

[Bra92] J.C. Bradfield. *Verifying Temporal Properties of Systems*. Birkhauser, 1992.

[BS92] O. Burkart and B. Steffen. Model Checking for Context-Free Processes. In *CONCUR'92*, 1992. LNCS 630.

[CES83] E.M. Clarke, E.A. Emerson, and E. Sistla. Automatic Verification of Finite State Concurrent Systems using Temporal Logic Specifications: A Practical Approach. In *POPL'83*. ACM, 1983.

[CG77] R.S. Cohen and A.Y. Gold. Theory of ω-Languages. I: Characterizations of ω-Context-Free Languages. *J.C.S.S.*, 15, 1977.

[CH93] S. Christensen and H. Hüttel. Decidability Issues for Infinite State Processes - A Survey. *Bull. of the EATCS*, 51, 1993.

[Chr93] S. Christensen. *Decidability and Decomposition in Process Algebra*. PhD thesis, Univerisity of Edinburgh, 1993.

[EL86] E.A. Emerson and C.L. Lei. Efficient Model-Checking in Fragments of the Propositional μ-Calculus. In *LICS'86*. IEEE, 1986.

[Esp93] Javier Esparza. On the Decidability of Model-Checking for Several Mu-calculi and Petri Nets. Report ECS-LFCS-93-274, Dep. of Comp. Sci., University of Edinburgh, 1993.

[Har78] M.A. Harrison. *Introduction to Formal Language Theory*. Addison-Wesley Pub. Comp., 1978.

[HP79] J. Hopcroft and J.-J. Pansiot. On The Reachability Problem for 5-Dimensional Vector Addition Systems. *T.C.S.*, 8, 1979.

[HPS83] D. Harel, A. Pnueli, and J. Stavi. Propositional Dynamic Logic of Nonregular Programs. *J.C.S.S.*, 26, 1983.

[HR90] D. Harel and D. Raz. Deciding Properties of Nonregular Programs. In *FOCS'90*. IEEE, 1990.

[Hüt91] H. Hüttel. *Decidability, Behavioural Equivalences and Infinite Transition Graphs*. PhD thesis, Univeristy of Edinburgh, 1991.

[Koz83] D. Kozen. Results on the Propositional μ-Calculus. *T.C.S.*, 27, 1983.

[KS83] P. Kanellakis and S.A. Smolka. CCS Expressions, Finite State Processes, and Three Problems of Equivalence. In *PODC'83*. ACM, 1983.

[LPZ85] O. Lichtenstein, A. Pnueli, and L. Zuck. The Glory of the Past. In *Conf. Logics of Programs*. LNCS 193, 1985.

[Mil80] R. Milner. A Calculus of Communication Systems. 1980. LNCS 92.

[MP90] Z. Manna and A. Pnueli. A Hierarchy of Temporal Properties. In *PODC'90*. ACM, 1990.

[MS85] D.E. Muller and P.E. Schupp. The Theory of Ends, Pushdown Automata, and Second-Order Logic. *T.C.S.*, 37, 1985.

[Par81] D. Park. Concurrency and Automata on Infinite Sequences. In *5th GI-Conference on Theoretical Computer Science*. 1981. LNCS 104.

[Pnu77] A. Pnueli. The Temporal Logic of Programs. In *FOCS'77*. IEEE, 1977.

[QS82] J-P. Queille and J. Sifakis. Specification and Verification of Concurrent Systems in CESAR. In *Intern. Symp. on Programming, LNCS 137*, 1982.

[Tho79] W. Thomas. Star-Free Regular Sets of ω-Sequences. *Information and Control*, 42, 1979.

[Tho90] W. Thomas. Automata on infinite objects. In *Handbook of Theo. Comp. Sci.* Elsevier Sci. Pub., 1990.

[Var88] M.Y. Vardi. A Temporal Fixpoint Calculus. In *POPL'88*. ACM, 1988.

[VW86] M.Y. Vardi and P. Wolper. An Automata-Theoretic Approach to Automatic Program Verification. In *LICS'86*. IEEE, 1986.

[Wol83] P. Wolper. Temporal Logic Can Be More Expressive. *Information and Control*, 56, 1983.