In *Proc. 14th Int. Workshop Computer Science Logic (CSL'2000), Fischbachau, Germany, Aug. 2000, volume 1862 of Lecture Notes in Computer Science, pages 262−276. Springer, 2000.*

Flatness is not a Weakness

Hubert Comon and Véronique Cortier

LSV, Ecole Normale Supérieure de Cachan
61 Avenue du président Wilson, 94235 Cachan cedex, France.
E-mail: {comon,cortier}@lsv.ens-cachan.fr
Tel: +33 1 47 40 24 30
Fax: +33 1 47 40 24 64

Abstract. We propose an extension, called $\mathcal{L}_p^+$, of the temporal logic LTL, which enables talking about finitely many register values: the models are infinite words over tuples of integers (resp. real numbers). The formulas of $\mathcal{L}_p^+$ are *flat*: on the left of an until, only atomic formulas or LTL formulas are allowed. We prove, in the spirit of the correspondence between automata and temporal logics, that the models of a $\mathcal{L}_p^+$ formula are recognized by a piecewise *flat* counter machine; for each state q, at most one loop of the machine on q may modify the register values.
Emptiness of (piecewise) flat counter machines is decidable (this follows from a result in [9]). It follows that satisfiability and model-checking the negation of a formula are decidable for $\mathcal{L}_p^+$. On the other hand, we show that inclusion is undecidable for such languages. This shows that validity and model-checking positive formulas are undecidable.

Keywords: Counter automata, temporal logics, model-checking, verification, logic in computer science.

1 Introduction

Temporal logics play a central role in the specification and verification of reactive systems (see e.g. [16]). Temporal logics come in two varieties: linear time and branching time [13]. We consider here the linear version PLTL. This (propositional) temporal logic is decidable (actually PSPACE-complete [18]). Model checking is also PSPACE-complete (linear w.r.t. the model). The set of words which satisfy a PLTL formula is recognized by a finite Büchi automaton, which shows the relatively weak expressive power of the logic; here we are interested in specifying and verifying infinite state systems.

One more general (hence more realistic) class of models would be machines with finitely many registers (or counters) taking their values in integers or real numbers and a finite control, of which the simplest example is Minsky machines. Unfortunately, even the most simple temporal property, reachability, is undecidable for 2-counters machines [17]. Several restrictions of this model have been studied. For instance Petri nets basically consist in removing the ability to test a counter for zero. Temporal properties of Petri nets have been studied in, e.g.,[15].

Another approach consists in adding hypotheses on the control instead of hypotheses on the basic operations only. That is the approach of [9]; a counter machine is called *flat* if there is at most one loop on each state. For such machines, the binary reachability relation between two control states is expressible in Presburger arithmetic [9], hence decidable. Flat automata are still a significant subclass of counter automata since, for instance, Alur and Dill's *timed automata* [2] can be encoded in this model [11].

The notion of flatness appears in several places. As we have seen, it appears to be a crucial hypothesis for counter machines. In [7] the authors study the set of reachable configurations for an automaton communicating through fifo channels. They show how to describe such a set of configurations using a Presburger formula, *provided that the control is flat*. Similarly, in [1] the authors study automata communicating through lossy fifo channels and introduce the so-called SRE which assume a flatness hypothesis on the control. This is not by chance that a similar hypothesis appears in several places: roughly, if only increments are allowed, using one loop one may compute addition and using two nested loops one can compute multiplication; from one loop to two nested loops we move from decidable to undecidable theories.

More interestingly, flatness appears naturally in PLTL itself: following the automata approach, the models of a PLTL formula are recognized by a *weak alternating automaton* (see e.g. [19]). Weakness means that there is an ordering on the states such that any state occurring in the image of q by the transition function is smaller (or equal to) q. Hence "weak" is a synonym of "flat" in the context of alternating automata, though the Büchi automaton accepting the same language as a weak alternating automaton may contain several loops on the same state, hence is not itself flat.

This raises the following question: assume that we design a temporal logic which includes as atomic formulas expressions involving finitely many counters and that we are able to construct for each formula ϕ an automaton which recognizes the models of ϕ, would the automaton be flat ? If this were the case, we could design decision procedures for such a logic, because we do have decision procedures for flat automata.

That is the purpose of the present paper: we define a *flat temporal logic* $\mathcal{L}_p$ whose atomic formulas include expressions such as $x \geq y - 1$ for instance where x, y are integer variables. "flatness" is a restriction in which only atomic formulas may occur on the left of an "until". If we drop such a restriction, we show that we immediately cross the boarder: the logic becomes undecidable. In [4, 6, 5] there are similar hypotheses: they design a logic in which it is possible to consider as a first class object the number of times a given propositional formula is satisfied. This logic is in general undecidable, but becomes decidable when on the left (resp. on the right) of an until only propositional formulas are allowed. Strictly speaking, the results of [4] for instance are incomparable with ours since neither the logic nor the models we consider are the same. (Roughly, they consider models which are described by a process algebra, i.e. in which there is no explicit counter. On the other hand, the integer variables in the logic only

count number of occurrences of a given event). Let us emphasize however that counting the number of steps which satisfy some proposition is possible in our logic: it suffices to add one counter and increase it each time the proposition is satisfied. In this respect, we get some "parametric quantitative reasoning" ([14]) for free; the number of times some transition is fired can be a free variable in our logic.

We prove that recognizability by a flat automaton is equivalent to definability in $\mathcal{L}_p$. Note that this result goes both ways: unlike PLTL for which only star-free languages are definable, here, any flat language is definable (and conversely any definable language is flat). We prove that satisfiability of $\mathcal{L}_p$ formulas, as well as model-checking the negation of formulas in $\mathcal{L}_p$ (against a model described by a flat automaton) are decidable: this is a consequence of the relationship between flat formulas and flat automata on one hand and decidability results for flat automata on the other hand.

$\mathcal{L}_p$ has however several weaknesses. First, it is not closed by negation. This cannot be avoided as we show that validity of $\phi \in \mathcal{L}_p$ as well as model-checking ϕ are undecidable. Phrasing these results in term of automata, though emptiness is decidable for flat automata, the universality is undecidable.

$\mathcal{L}_p$ does not contain LTL. However, we can design a logic $\mathcal{L}_p^+$ which embeds both LTL and $\mathcal{L}_p$, while keeping the nice decidability properties. Now, instead of flat automata, each formula of $\mathcal{L}_p^+$ can be associated with a *piecewise flat automaton* which accepts the models of the formula. Emptiness remains decidable for such automata, which implies again that satisfiability and model-checking the negation of a formula are decidable (this includes reachability for instance).

We start in section 2 by definitions and examples of (flat) counter automata. In section 3 we establish (un)decidability results for flat automata. The flat logic $\mathcal{L}_p$ is introduced in section 4 where we also prove the correspondence with flat automata. Then we consider in section 5 the decision problems for this logic. Finally, in section 6 we consider the extension $\mathcal{L}_p^+$ which also embeds LTL.

2 Flat counter automata

Our constraints relate the current values (unprimed variables) and the next values (primed variables) of the counters, in a declarative way.

Definition 1 (Constraint). *An* atomic constraint *is one of the expressions:* $x\#y + c$, $x\#c$, $c\#x$ *where* $\# \in \{\leq, <\}$ *and* $c \in \mathbb{Z}$ *(resp.* $c \in \mathbb{Q}$*). A* constraint c *is either the constant* **true**, *the constant* **false** *or a conjunction of atomic constraints. The set of constraints with free variables* $x_1, \ldots, x_k, x'_1, \ldots, x'_k$ *is written* $\mathcal{C}(x_1, \ldots, x_k)$.

A constraint c in $\mathcal{C}(x_1, \ldots, x_k)$ defines a binary relation R_c on D^k where $D \in \{\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \ldots\}$: the relational symbols $\leq, <$ are interpreted as the usual ordering, as well as constant addition. $(v, v') \in R_c$ iff the valuation in which the ith component of v is assigned to x_i and the ith component of v' is assigned to x'_i satisfies c.

Definition 2 (Counter automaton: syntax). *An* automaton with k counters *$\mathcal{A}$ is a tuple $(\Sigma, Q, q_0, F, \delta)$ where Q is a finite set of* states*, Σ is a finite alphabet, $q_0 \in Q$ is the* initial state*, $F \subseteq Q$ is the set of* final states *and $\delta \subseteq Q \times \mathcal{C}(x_1, \ldots, x_k) \times \Sigma \times Q$ is a* transition relation*. We write sometimes $q \xrightarrow[\mathcal{A}]{c,a} q'$ instead of $(q, c, a, q') \in \delta$.*

A configuration *of the automaton is a pair (q, v) with $q \in Q$ and $v \in \mathbb{N}^k$ (resp. $v \in \mathbb{Q}_+^k$). The automaton may* move *from a configuration (q, v) to a configuration (q', v') iff there is a transition $(q, \phi, a, q') \in \delta$ such that $v, v' \models \phi$: the free variables $x_1, \ldots, x_k$ are interpreted by $v_1, \ldots, v_k$ and the free variables $x'_1, \ldots, x'_k$ are interpreted by $v'_1, \ldots, v'_k$. We write $(q, v) \xrightarrow[\mathcal{A}]{a} (q', v')$ when the automaton $\mathcal{A}$ may move from a configuration (q, v) to a configuration (q', v') while reading a. a may be dropped if it is not relevant.*

Definition 3 (Counter automaton: semantics). *Let w be finite (resp. infinite) word of length $|w|$: $w \in (\Sigma \times \mathbb{N}^k)^*$ (resp. $w \in (\Sigma \times \mathbb{N}^k)^\omega$). A* run *of $\mathcal{A}$ on w is a finite (resp. infinite) word $\rho \in Q^*$ of length $|w|$ (resp. $\rho \in Q^\omega$) such that $\rho(1) = q_0$ and, for every $1 \leq i \leq |w|-1$ (resp. $i \geq 1$), $(\rho(i), v_i) \xrightarrow[\mathcal{A}]{} (\rho(i+1), v_{i+1})$ if $w(i) = (a_i, v_i)$.*

A run ρ is successful *if its last letter belongs to F (resp. if it contains infinitely many elements of F). A word w is* accepted *by $\mathcal{A}$ if there is a successful run of $\mathcal{A}$ on w.*

We write $L(\mathcal{A})$ the set of finite words accepted by $\mathcal{A}$ and $L_\omega(\mathcal{A})$ the language of infinite words accepted by $\mathcal{A}$.

Example 1. On figure 1 we have depicted a controller for a pay phone. There are two counters: x is the number of quarters which have been inserted and y measures the total communication time. We use the classical abbreviations: $x{+}{+}$ stands for $x' = x+1$ and $x{-}{-}$ stands for $x' = x-1$. Also, by convention, when x' (resp. y') is not present in a transition, the constraint $x' = x$ (resp. $y' = y$) is assumed.

Such an automaton is expected to interact with its environment; messages are followed either by a question mark, when they are received by the controller, or by an exclamation mark, when they are sent by the controller. These aspects are however irrelevant here.

The initial state (which is also the only final state) is q_1. A possible sequence of consecutive moves of the automaton is:

$$q_1, \begin{pmatrix}0\\0\end{pmatrix} \xrightarrow{lift?} q_2, \begin{pmatrix}0\\0\end{pmatrix} \xrightarrow{quarter?} q_2, \begin{pmatrix}1\\0\end{pmatrix} \xrightarrow{dial?} q_3, \begin{pmatrix}1\\0\end{pmatrix}$$
$$\xrightarrow{quarter?} q_3, \begin{pmatrix}2\\0\end{pmatrix} \xrightarrow{quarter?} q_3, \begin{pmatrix}3\\0\end{pmatrix} \xrightarrow{connected?} q_4, \begin{pmatrix}3\\0\end{pmatrix} \ldots$$

Note that, by choice of the final state, it is not possible to insert quarters forever.

Definition 4. *A counter automaton over a single letter alphabet ($|\Sigma| = 1$) is* flat *if there is an ordering on the states such that there is a possible move from some (q, v) to some (q', v') only if $q \geq q'$. Moreover, there is at most one transition from a state to itself.*

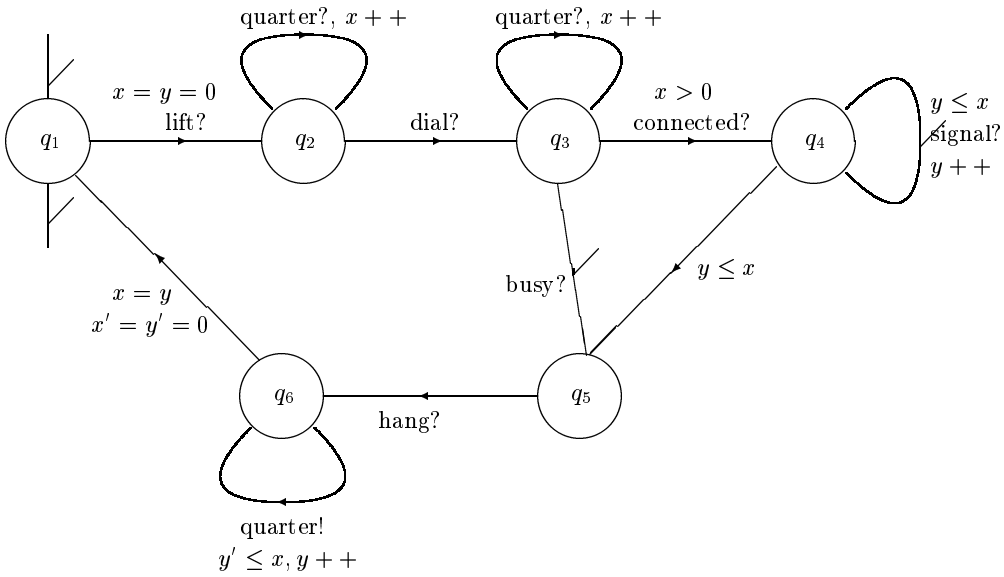

Fig. 1. A pay phone

Example 2. Consider the pay phone of figure 1 in which we forget the messages. The resulting automaton is not flat as there are several loops on a single state (e.g. q_2). It is however possible to replace each loop on a single state with a single transition, without changing the reachability relation. For instance the iteration of a loop labeled with $x++$ can be replaced with a single transition $x' > x$. Then the one step loops on q_2, q_3, q_4 and q_6 can be replaced with single transitions and the automaton becomes flat.

Also, if we remove the transition between q_6 and q_1, the automaton becomes flat.

3 (Un)decidability results for flat counter automata

We first recall here the decision results which can be derived from [9]. Then we prove new undecidability results.

Theorem 1 ([9]). *Given two states q_1, q_2 of a flat counter automaton A, there is an effectively computable formula of Presburger arithmetic $\phi_{q_1,q_2}(\boldsymbol{x}, n, \boldsymbol{x'})$ with $2k+1$ free variables such that $(q_1, v) \xrightarrow[A]{m} (q_2, v)$ iff $v, m, v' \models \phi_{q_1,q_2}$.*

where $\xrightarrow[A]{m} = \underbrace{\xrightarrow[A]{} \cdots \xrightarrow[A]{}}_{m}$.

Corollary 1 ([10]). *The emptiness of $L(A)$ (resp. $L_\omega(A)$) is decidable for flat automata A.*

Decidability of the emptiness of $L(A)$ follows directly from theorem 1: it suffices to decide $\exists m.q_0 \xrightarrow[A]{m} q_f$ for every final state q_f. Concerning $L_\omega(A)$, we need to decide the infinite iterability of a loop, which is also a consequence of the particular expression of the reachability relation, with some additional work [10].

Proposition 1. *The class of languages recognized by flat counter automata is effectively closed by union and intersection (both in the finite and in the infinite words cases).*

Proof sketch: The closure by union is straightforward. The closure by intersection is a consequence of the closure of $\mathcal{C}(x_1, \ldots, x_k)$ by conjunction. □

Unfortunately the class of languages recognized by flat automata is not closed under complement. Actually, we are going to show that the question of whether a flat automaton accepts all words in $(\mathbb{N}^k)^*$ is undecidable, which gives the non-closure results thanks to corollary 1.

First, consider the set CA_1 of counter automata over a one letter alphabet such that there is exactly one transition starting from a final state, which is labeled with **true**. The reachability of a final state in a Minsky machine reduces to the emptiness of the language recognized by such a counter automaton. Hence we have the undecidability result:

Lemma 1. *The emptiness problem for $L(A)$ (resp. $L_\omega(A)$) is undecidable for $A \in CA_1$.*

We may further restrict the class of counter machines, encoding the states into a counter. Let CA_2 be the class of automata in CA_1 which only contain two states q_1, q_2, such that q_2 is final and there is no transition from q_2 to q_1. (See figure 2.)

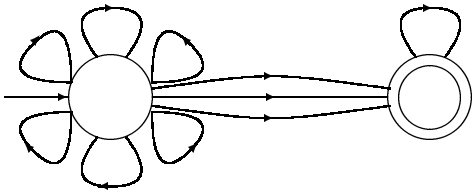

Fig. 2. An automaton in CA_2

If A is an automaton with $k+1$ counters and x is one particular counter then the *projection* $\pi_x(L(A))$ (resp. $\pi_x(L_\omega(A))$) is the subset of $(\mathbb{N}^k)^*$ (resp. $(N^k)^\omega$) of words in $L(A)$ in which the x component has been erased.

Lemma 2. *For every automaton $A \in CA_1$ with k counters $x_1, \ldots, x_k$, there is an automaton $A' \in CA_2$ with $k+1$ counters $c, x_1, \ldots, x_k$ such that $L(A) = \pi_c(L(A'))$ and there is a flat automaton A'' such that $L(A'') = (\mathbb{N}^k)^* - L(A')$ (resp. $L_\omega(A'') = (\mathbb{N}^k)^\omega - L_\omega(A')$)*

Proof sketch: First add a counter c which records the state number; without loss of generality, we may assume that numbering the states is such that $Q = \{q_1, \ldots, q_n\}$ and $Q_f = \{q_f, \ldots, q_n\}$ (i.e. states whose number is larger than f are final). The automaton A' contains two states: Q and Q_f. A transition from state i to state j with a constraint ϕ becomes, when i is not final (for instance), a constraint $\phi \wedge c = i \wedge c' = j$ from the initial state to itself, or to the final state if $q_j \in Q_f$.

Let $\phi_1, \ldots, \phi_n$ be the constraints of the transitions on the initial state and $\psi_1, \ldots, \psi_m$ be the constraints of the transitions from the initial to the final state in A'. Note that, by construction, for every i, $\psi_i \models c' \geq f$.

Let $g_1 \vee \ldots \vee g_r$ be a disjunction of constraints which is logically equivalent to

$$\neg((\bigvee_{i=1}^{n} \phi_i) \vee (\bigvee_{i=1}^{m} \psi_i))$$

Such a disjunction of constraints always exist since the negation of an atomic constraint can always be written as a disjunction of atomic constraints.

Our flat automaton is built as depicted on figure 3. A word which is not

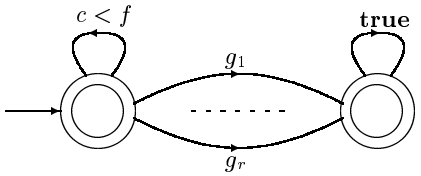

Fig. 3. The flat automaton in the proof of lemma 2.

accepted either never reaches a final state, i.e. c remains strictly smaller than f, or else it is not compatible with the transition relation at some point, before reaching a final state.

□

Lemma 2 is a little bit confusing; one may get the impression that the complement of any counter language (over a one letter alphabet) is a recognized by a flat automaton. This is not true, however; the projection plays an important role here. On the other hand, we know that the complement of a flat automaton, cannot be always recognized by a flat automaton: universality would then be decidable, hence the emptiness for any counter automaton.

From the two previous lemmas we can derive the following:

Theorem 2. *The universality is undecidable for flat automata (both in the case of finite and in the case of infinite words).*

4 The flat counter logic $\mathcal{L}_p$

We introduce first a logic with counters CLTL, which, unfortunately, is too expressive. However, the notion of flat automaton which we introduced in the last section can be easily characterized at the logical level using a restriction of CLTL, which is similar to the so-called "flat fragment" in [12] for instance.

4.1 A logic with counters

Basically, we consider a temporal logic whose modalities are the same as in PLTL. The only difference is that, instead of propositional atomic formulas, we allow arbitrary constraints in $\mathcal{C}(x_1, \ldots, x_k)$.

More precisely, given a natural number k and a finite set of propositional variables $\mathcal{P}$, CLTL is the smallest set of formulas such that P belongs to CLTL for every $P \in \mathcal{P}$, $\mathcal{C}(x_1, \ldots, x_k)$ is included in CLTL and if ϕ_1 and ϕ_2 are formulas of CLTL, then $\phi_1 \wedge \phi_2, \phi_1 \vee \phi_2, \neg\phi_1, \mathbf{X}\phi_1, \phi_1 \,\mathcal{U}\, \phi_2$ are formulas of CLTL.

We may also use the classical derived operators $\Box$ ("henceforth") and $\Diamond$ ("eventually").

Temporal formulas are interpreted over *computations* which are now infinite words in $2^{\mathcal{P}} \times \mathbb{N}^k$. Given an infinite path $\pi \in (2^{\mathcal{P}} \times \mathbb{N}^k)^\omega$, we write $\pi(i)$ for the ith letter of π and we let $\widetilde{\pi}$ be the infinite word in $(2^{\mathcal{P}} \times \mathbb{N}^k \times \mathbb{N}^k)^\omega$ defined by:

$$(\pi(i) = (a, \boldsymbol{v}) \text{ and } \pi(i+1) = (b, \boldsymbol{w})) \text{ implies } \widetilde{\pi}(i) = (a, v, w)$$

This little technicality is necessary because the constraints may express relations between two successive values of the counters and not only constraints on a given value of the counters.

Now, a path π satisfies ϕ iff $\widetilde{\pi}, 0 \models \phi$ and:

- $\widetilde{\pi}, i \models \mathbf{true}$ and $\widetilde{\pi}, i \not\models \mathbf{false}$
- $\widetilde{\pi}, i \models P$ where $P \in \mathcal{P}$ if and only if $\pi(i) = (a, \boldsymbol{v})$ and $P \in a$
- $\widetilde{\pi}, i \models \phi(x_1, \ldots, x_k, x'_1, \ldots, x'_k)$ where $\phi \in \mathcal{C}(x_1, \ldots, x_k)$ iff $\widetilde{\pi}(i) = (a, \boldsymbol{v}, \boldsymbol{w})$ and $\boldsymbol{v}, \boldsymbol{w} \models \phi$ (with the usual definition of satisfaction in Presburger arithmetic).
- $\widetilde{\pi}, i \models \mathbf{X}\phi$ iff $\widetilde{\pi}, i+1 \models \phi$,
- $\widetilde{\pi}, i \models \phi_1 \wedge \phi_2$ iff $\widetilde{\pi}, i \models \phi_1$ and $\widetilde{\pi}, i \models \phi_2$, ...
- $\widetilde{\pi}, i \models \phi_1 \,\mathcal{U}\, \phi_2$ iff there is an index $j \geq i$ such that $\widetilde{\pi}, j \models \phi_2$ and for all $k \in [i, j[$, $\widetilde{\pi}, k \models \phi_1$.

Example 3. CLTL allows to express properties such as: "x is never greater than 100" or "each time x is larger than 100, an alarm is raised" or "ultimately, the register x remains stable" :

$$\Box(x \leq 100), \qquad \Box(x \leq 100) \vee (x \leq 100 \,\mathcal{U}\, alarm \geq 1) \qquad \Diamond\Box(x' = x)$$

Unfortunately, CLTL is too expressive:

Theorem 3. *Satisfiability is undecidable for CLTL. Model checking (of a flat automaton) is also undecidable in this logic.*

Proof sketch: We reduce the halting problem of a counter machine. Roughly, we use an auxiliary variable c ranging over the states of the machine and encode the computations of the machine by the formula:

$$c = q_0 \wedge \left(\left[\bigwedge_i (c = q_i \Rightarrow \bigvee_{q_i \xrightarrow{G} q_j} [G_{q_i,q_j}(\boldsymbol{x}, \boldsymbol{x'}) \wedge c' = q_j]) \right] \mathcal{U} \left[\bigvee_{q_f \in F} c = q_f \right] \right)$$

□

4.2 The flat fragment of the logic

$\mathcal{L}_p$ is defined by a syntactic restriction of the formulas, which, roughly, restricts the left members of "until" to be conjunctions of atomic formulas, thus preventing the construction of theorem 3. For simplicity, we assume here that $\mathcal{P} = \emptyset$; propositional variables will be re-introduced in section 6 and, anyway, they can be encoded by integer variables.

Definition 5. *An* elementary formula *is a Boolean combination of constraints in $\mathcal{C}(x_1, \ldots, x_k)$.*

The set $\mathcal{L}_p$ of flat formulas, is the smallest subset of CLTL such that:

- *elementary formulas are flat*
- *if ϕ_1, ϕ_2 are flat, then $\phi_1 \wedge \phi_2, \phi_1 \vee \phi_2$, $\mathbf{X}\phi_1$ are flat.*
- *if ϕ_1 is a constraint in $\mathcal{C}(x_1, \ldots, x_k)$ and ϕ_2 is flat, then $\phi_1 \,\mathcal{U} \phi_2$ is flat*
- *if ϕ is a constraint in $\mathcal{C}(x_1, \ldots, x_k)$, then $\neg(\mathbf{true}\,\mathcal{U} \neg\phi)$ (i.e. $\Box\phi$) is flat*

The last condition is ad-hoc: it corresponds to the encoding of final states, as we will see.

Let us emphasize that $\mathcal{L}_p$ is *not* closed by negation. This is unavoidable as we will see in the next section. On the other hand, we could add the weak until, as both $\Box\phi$ and $\phi \,\mathcal{U} \psi$ are in $\mathcal{L}_p$ when ϕ is a constraint.

Example 4. The formulas given in example 3 are all flat.

One of the main interest of $\mathcal{L}_p$ is the correspondence with flat automata:

Theorem 4. *For every formula ϕ of $\mathcal{L}_p$, there is a flat automaton which accepts the models of ϕ.*

Conversely, for every flat automaton $\mathcal{A}$, there is a formula ϕ of $\mathcal{L}_p$ whose models are the words accepted by $\mathcal{A}$.

Proof sketch: From logic to automata we use the closure properties of flat automata by union and intersection (theorem 1) and the standard constructions for $\mathcal{U}$, $\mathbf{X}$ and $\Box$. For instance consider $\phi\ \mathcal{U}\psi$. By hypothesis, ϕ belongs to $\mathcal{C}(x_1, \ldots, x_k)$. We construct the automaton for $\phi\,\mathcal{U}\psi$ by adding in front of the automaton for ψ a state on which there is a loop guarded by ϕ

From the automata to the logic, we proceed by induction on the ordering on states. From minimal states q there is at most one departing transition, say labeled with ϕ, and whose target is q itself. Then, if q is final, the corresponding formula will be $\Box\phi$ (**false** otherwise). For the induction step, if $q_1, \ldots, q_n$ are the successors of q and ϕ is the constraint of the loop on q, we get roughly the formula $\phi\,\mathcal{U}((\phi_1 \wedge \mathbf{X}\phi_{q_1}) \vee \ldots (\phi_n \wedge \mathbf{X}\phi_{q_n}))$. $\Box$

5 Satisfiability and model-checking in $\mathcal{L}_p$

Thanks to theorem 4 we can decide satisfiability and model checking of the negation of a formula of $\mathcal{L}_p$:

Theorem 5. *Given a formula $\phi \in \mathcal{L}_p$ and a flat automaton $\mathcal{A}$, the following questions are decidable:*

- *Is ϕ satisfiable ?*
- *Does $\mathcal{A}$ satisfy $\neg\phi$? (In other words, is there a word accepted by $\mathcal{A}$ which is a model of ϕ ?)*

Proof: Thanks to theorem 4, for every formula $\phi \in \mathcal{L}_p$, there is an automaton $\mathcal{A}_\phi$ which accepts the models of ϕ. Then satisfiability reduces to the emptiness of $L(\mathcal{A}_\phi)$ and $\mathcal{A} \models \neg\phi$ reduces to $L(\mathcal{A}) \cap L(\mathcal{A}_\phi) = \emptyset$. Now, thanks to theorems 1 and 1, both questions are decidable. $\Box$

Example 5. Negation of formulas in $\mathcal{L}_p$ include for instance reachability formulas $\Diamond q$ (adding here a new counter whose value is 0, except when reaching q) or safety formulas $\Box\neg\phi$ where ϕ is a constraint. Actually, considering the formulas in example 3, the negations of the first two formulas also belong to $\mathcal{L}_p$ because the negation of constraints $s \geq t$ are atomic constraints and the negation of $c\,\mathcal{U}c'$ is in $\mathcal{L}_p$ when c, c' are both of the form $s \geq t$. Only the negation of $\Diamond\Box x' = x$ is not a $\mathcal{L}_p$ formula.

It is also possible to reduce in polynomial time Presburger arithmetic satisfiability to $\mathcal{L}_p$ satisfiability, hence, in principle, $\mathcal{L}_p$ is at least as hard as Presburger arithmetic (between 2-DEXPTIME and 3-DEXPTIME).

Now, deciding $\mathcal{A} \models \phi$ for $\phi \in \mathcal{L}_p$ is equivalent to the decision of inclusion of flat automata, which is undecidable:

Theorem 6. *The validity problem and the model checking on a flat automaton are undecidable for a formula $\phi \in \mathcal{L}_p$.*

Sketch of the proof: This follows from theorems 4 and 2. $\Box$

6 $\mathcal{L}_p^+$: a decidable extension of $\mathcal{L}_p$ and *LTL*

The logic $\mathcal{L}_p$ is not fully satisfactory in many respects. In particular, the restrictions on the left member of an $\mathcal{U}$ disallow arbitrary LTL formulas. On the other hand, theorem 3 shows that we cannot simply drop the restriction. At least, we have to consider positive Boolean combinations of PLTL formulas and $\mathcal{L}_p$ formulas. We can still go a little further, as we will see.

Informally, $\mathcal{L}_p^+$ extends $\mathcal{L}_p$ by allowing any conjunction of a PLTL formula and a constraint where only constraints were allowed.

Definition 6 (Syntax of $\mathcal{L}_p^+$). *We assume given a finite set of propositional variables $\mathcal{P}$ and a positive integer k.*

Given a constraint ϕ, $PLTL_\phi$ is the smallest set of temporal formulas containing $\phi \wedge P_1 \wedge \ldots \wedge P_n \wedge \neg Q_1 \wedge \ldots \wedge \neg Q_m$ for every propositional variables $P_1, \ldots, P_n, Q_1, \ldots, Q_m$ and which is closed by $\wedge, \vee$, $\mathcal{U}, \mathbf{X}, \Box$. A basic formula *is a formula $\psi \in PLTL_\phi$ for some $\phi \in \mathcal{C}(x_1, \ldots, x_k)$.*

$\mathcal{L}_p^+$ is the smallest set of formulas such that:

- *every basic formula is in $\mathcal{L}_p^+$,*
- *if ϕ_1, ϕ_2 are in $\mathcal{L}_p^+$, then $\phi_1 \wedge \phi_2$, $\phi_1 \vee \phi_2$, $\mathbf{X}\phi_1$ are in $\mathcal{L}_p^+$*
- *if ϕ_1 is a basic formula and $\phi_2 \in \mathcal{L}_p^+$, then $\phi_1 \,\mathcal{U} \phi_2 \in \mathcal{L}_p^+$*
- *if ϕ_1 is a basic formula, then $\Box\phi_1 \in \mathcal{L}_p^+$.*

Note that, in PLTL, negation can be pushed to the propositional variables level if we include $\Box$ in the syntax. That is why PLTL formulas are basic formulas in the above definition: it is sufficient to choose $\phi = \mathbf{true}$. Constraints are also basic formulas, hence $\mathcal{L}_p^+$ is an extension of both $\mathcal{L}_p$ and PLTL.

On the other hand $\mathcal{L}_p^+$ is a fragment of the logic CLTL which was defined in section 4.1, from which we borrow the the semantics.

Example 6. We may record the elapsed time in a LTL formula using an auxilliary counter; for instance:

$$x = 0 \wedge ((p \wedge (x' = x + 1))\,\mathcal{U}(Q \wedge x' = x + 1))\,\mathcal{U}(R \vee x > \alpha)$$

is an $\mathcal{L}_p^+$formula, x recording the elapsed time. We could consider e.g. a second phase in R in which the time spent for each action is larger (or smaller), or even record something different, as, e.g., distance or available resources... However, it is not allowed to replace one of the two occurrences of $x + 1$ with $x + 2$: on the left of an until the constraint has to be the same everywhere.

Here, we have to extend the notion of a flat automaton, corresponding to the extension of the syntax of formulas.

Definition 7. *A* piecewise flat automaton *is a counter automaton on an alphabet $\Sigma = 2^{\mathcal{P}}$ such that there is a partition $Q_1 \uplus \ldots \uplus Q_m$ of the set of states Q and an ordering on $\{Q_1, \ldots, Q_n\}$ such that:*

- *for every i, there is a constraint $\phi_i \in \mathcal{C}(x_1, \ldots, x_k)$*

– *for every transition* $q \xrightarrow{c,a} q'$ *of the automaton, if* $q \in Q$ *and* $q' \in Q'$*, then* $Q \geq Q'$
– *for every transition* $q \xrightarrow{c,a} q'$ *such that* $q, q' \in Q_i$*, there is a conjunction* ψ *of proprositional variables and negations of propositional variables such that* $c = \phi_i \wedge \psi$

Example 7. Consider the pay phone example of figure 1. With each event, we associate a propositional variable. Then the behavior between two lift events (i.e. a "session") is described by a piecewise flat automaton. Actually, more complex actions could be described within the same class of models, for instance using more coins types, calling services...

Proposition 2. *The class of languages accepted by piecewise flat automata is closed under union and intersection.*

Sketch of the proof: It is almost the same as the closure of flat languages. We use the closure of $\mathcal{C}(x_1, \ldots, x_n)$ by conjunction and, for intersection, a product construction which is similar to the Büchi automata intersection construction. □

Theorem 7. *The models of an* $\mathcal{L}_p^+$ *formula are recognized by a piecewise flat automaton.*

Sketch of the proof: As before, we proceed by induction on the formula. Thanks to proposition 2, we only have to show the construction for $\mathbf{X}$ and $\mathcal{U}$. The construction for $\mathcal{U}$ is actually complicated. An example is depicted on figure 4. Let $\mathcal{A}_{\phi_1}$ be the automaton accepting the models of ϕ_1, Q_1 its set of states, and

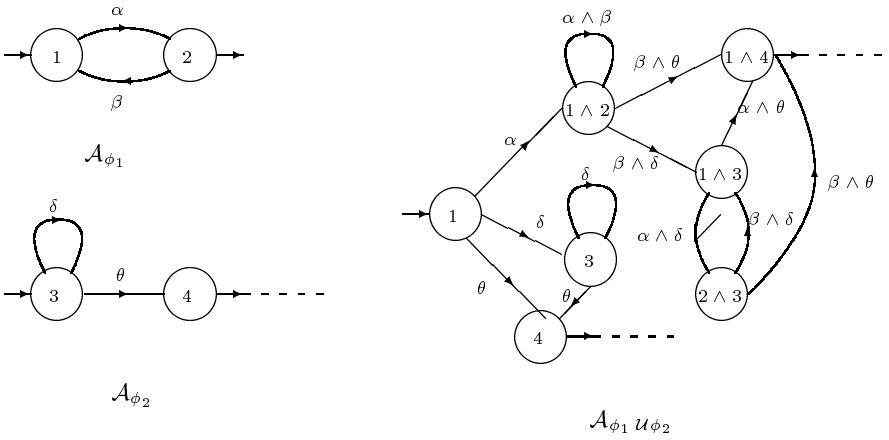

Fig. 4. The piecewise flat automaton for $\phi_1 \mathcal{U} \phi_2$

$\mathcal{A}_{\phi_2}$ be the automaton accepting the models of ϕ_2 and Q_2 its set of states.

The idea is the following: while we do not reach a point where ϕ_2 is satisfied, at each move, the automaton launches a copy of $\mathcal{A}_{\phi_1}$ on the rest of the word. This is shown on an example on figure 4. Hence the set states of the automaton $\mathcal{A}_{\phi_1 \mathcal{U} \phi_2}$ is the union of $\{S \subseteq 2^{Q_1}, \mid q_0 \in S\}$ and $2^{Q_1} \times Q_2$, if q_0 is the initial state of $\mathcal{A}_{\phi_1}$.

The initial state of $\mathcal{A}_{\phi_1 \mathcal{U} \phi_2}$ is the singleton $\{q_0\}$, the final states are the pairs (S, q) where $S \subseteq Q_f^1$ and $q \in Q_f^2$, respectively the final states of $\mathcal{A}_{\phi_1}$ and $\mathcal{A}_{\phi_2}$. Transitions are computed as follows: a state $S \subseteq Q_1$ is considered as the conjunction for all states in S; if $S, S' \subseteq Q_1$, $q_0 \in S'$, f is a mapping from S to S', then there is a transition from S to S' which is labeled $\bigwedge_{q \in S} c_{q,f(q)}$ where $c_{q,f(q)}$ is the constraint of one of the transitions from q to $f(q)$ in $\mathcal{A}_{\phi_1}$. This corresponds to the case where we did not hit yet a position at which ϕ_2 is satisfied. We may also move from a state S to a state (S', q') if $q' \in Q_2$ and $S' \subseteq Q$, under the same conditions as above, except that we do not require $q_0 \in S'$ and move instead from the initial state of $\mathcal{A}_{\phi_2}$ to q' (see figure 4): this corresponds to the guess that we are going to satisfy ϕ_2 at the current position. Finally, we also have transitions from (S, q) to (S', q') which corresponds basically to the intersection of copies of $\mathcal{A}_{\phi_1}$ and one copy of $\mathcal{A}_{\phi_2}$.

The construction would be similar if we defined an alternating version of the automata and then transform it into a non-deterministic one: the exponential blow-up is unavoidable for the states of the formula ϕ_1.

One important remark is that we still get a piecewise flat automaton here, which would not be the case if we allowed arbitrary $\mathcal{L}_p^+$ formulas on the left of an until. Indeed, the powerset construction for $\mathcal{A}_{\phi_1}$ introduces transitions which are labeled with arbitrary conjunctions of constraints occurring in ϕ_1. It remains piecewise flat only because all these constraints are identical. □.

Theorem 8. *Emptiness is decidable for piecewise flat automata.*

Sketch of the proof: We only have to check the reachability for the projection automaton, where we forget the letters of Σ. Then all states in the same Q_i collapse into a single state and we are back to corollary 1. □

Theorem 9. *Satisfiability and model checking of $\neg\phi$ on A are decidable for $\phi \in \mathcal{L}_p^+$ an A a piecewise flat automaton.*

Sketch of the proof: This follows from theorems 8 and 7 and proposition 2. □

Finally, let us remark that we can also consider the conjunction of $\mathcal{L}_p^+$ formulas with arbitrary constraints in the additive theory of our domain D ($\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}_+$, $\mathbb{R}_+$). It is not difficult to see directly how satisfiability and model-checking can be decided, but there is one elegant way to do it:

Proposition 3. *For every formula ϕ in Presburger arithmetic, whose free variables are $x_1, \ldots, x_k$, there is a flat automaton $\mathcal{A}_\phi$ with $k+m$ counters such that,*

if E is the set of last letters of finite words accepted by $\mathcal{A}_\phi$, then

$$\{v \in \mathbb{N}^k, v \models \phi\} = \{v \in \mathbb{N}^k, \exists w \in \mathbb{N}^m, (v, w) \in E\}$$

Then we can build a piecewise flat counter automaton which accepts the models of both the $\mathcal{L}_p^+$ formula and the first-order constraint.

In other words, the proposition says that we can encode Presburger arithmetic in $\mathcal{L}_p^+$, which shows that we can perform some general parametric quantitative reasoning.

7 Conclusion

The symbolic representation of states played a crucial role in increasing the efficiency of model-checkers [8]. It is even more crucial for infinite states systems. We believe that *constraints*, i.e. logical formulas interpreted in a given domain, are an adequate symbolic representation in this case. The main advantage w.r.t. other representations is its declarativeness and the easy combination with logical formalisms.

In this paper, we provided with an example of application: we can design a temporal logic which combines the representation of infinite sets of configurations using constraints and the usual temporal properties. We have also shown a device (automaton) accepting the set of models, hence allowing to decide e.g. the satisfiability.

This generalizes the results on LTL satisfiability and model-checking: it is now possible to consider counters in a restricted way. Unlike in the previous works, we put the restrictions on the control of the automaton (flatness), which has a logical counterpart.

There is still one important weakness of our results: we do not know anything about their possible usefulness in practice. In principle, the complexity of the algorithms are prohibitive. However, the main source of complexity is the number of counters, which can be low (2 or 3) in many examples.

As we noticed at the end of the previous section, it is possible to express some parametric quantitative properties, as defined in [3, 14] using additional counters and the logic $\mathcal{L}_p^+$. For instance, $\phi\ \mathcal{U}_{\leq x}\psi$ can be translated using an additional counter y into: $y = 0 \wedge ((\phi \wedge y' = y + 1)\,\mathcal{U}(y \leq x \wedge \psi))$. We want to investigate this application: which fragments of the PLTL logic of [3] are (easily) expressible in $\mathcal{L}_p^+$? For these fragments, we can check quantitative properties not only on finite automata, but also on piecewise flat automata with counters.

Another possible further investigation would be to consider the branching time temporal logic instead of PLTL.

References

1. P. A. Abdulla, A. Bouajjani, and B. Jonsson. On-the-fly analysis of systems with unbounded, lossy fifo channels. In *Proc. Computer Aided Verification*, volume 1427 of *Lecture Notes in Computer Science*, pages 305–318. Springer-Verlag, 1998.

2. R. Alur and D. Dill. Automata for modeling real-time systems. In *Proc. 17th Int. Coll. on Automata, Languages and Programming, Warwick, LNCS 443*, pages 322–335. Springer-Verlag, 1990.
3. R. Alur, K. Etessami, S. La Torre, and D. Peled. Parametric temporal logic for model measuring. In *Proc. Int. Conf. on Automata, Languages and Programming (ICALP'99)*, volume 1644 of *Lecture Notes in Computer Science*, pages 159–168, Prague, 1999. Springer-Verlag.
4. A. Bouajjani, R. Echahed, and P. Habermehl. On the verification problem of nonregular properties for nonregular processes. In *Tenth Annual IEEE Symposium on Logic in Computer Science*, pages 123–133, 1995.
5. A. Bouajjani, R. Echahed, and P. Habermehl. Verifying infinite state processes with sequential and parallel composition. In *Proc. POPL'95*, pages 95–106, San Francisco, 1995.
6. A. Bouajjani, R. Echahed, and R. Robbana. Verification of nonregular temporal properties of context free processes. In *Proc. CONCUR'94*, volume 836 of *Lecture Notes in Computer Science*, pages 81–97. Springer-Verlag, 1994.
7. A. Bouajjani and P. Habermehl. Symbolic reachability analysis of FIFO channel systems with non regular sets of configurations. In *Proc. 24th Int. Coll. on Automata, Languages and Programming (ICALP)*, volume 1256 of *Lecture Notes in Computer Science*, 1997.
8. J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic model checking: 10^{20} states and beyond. *Information and Computation*, 98(2):142–170, June 1992.
9. H. Comon and Y. Jurski. Multiple counters automata, safety analysis and presburger arithmetic. In A. Hu and M. Vardi, editors, *Proc. Computer Aided Verification*, volume 1427 of *LNCS*, pages 268–279, Vancouver, 1998. Springer-Verlag.
10. H. Comon and Y. Jurski. Counter automata, fixpoints and additive theories. Submitted to TCS. Available at http://www.lsv.ens-cachan.fr/~comon/ftp.articles/mca.ps.gz, 1999.
11. H. Comon and Y. Jurski. Timed automata and the theory of real numbers. In *Proc. Conf. on Concurrency Theory (CONCUR)*, number 1664 in Lecture Notes in Computer Science, pages 242–257. Springer-Verlag, 1999.
12. D. R. Dams. Flat fragments of ctl and ctl*. *Journal of the IGPL*, 7(1):55–78, 1999.
13. E. Emerson and J. Y. Halpern. Sometimes and not never revisited. *J. ACM*, 33, 1986.
14. E. Emerson and R. Trefler. Parametric quantitative temporal reasoning. In *Proc. IEEE Symp. on Logic in Computer Science*, pages 336–343, Trento, 1999. IEEE Comp. Soc. Press.
15. J. Esparza. Decidability of model checking for infinite-state concurrent systems. *Acta Informatica*, 34:85–107, 1997.
16. Z. Manna and A. Pnueli. *Temporal Verification of Reactive Systems. Safety.* Springer-Verlag, 1995.
17. M. Minsky. *Computation, Finite and Infinite Machines.* Prentice Hall, 1967.
18. A. Sistla and E. M. Clarke. The complexity of propositional linear temporal logic. *J. ACM*, 32:733–749, 1985.
19. M. Vardi. An automata-theoretic approach to linear time temporal logic. In *Logic for concurrency: structure versus automata*, volume 1043 of *Lecture Notes in Computer Science*. Springer Verlag, 1996.