Decidable Problems for Counter Systems

(revised version)

ESSLLI 2010, COPENHAGEN

Stéphane Demri
demri@lsv.ens-cachan.fr

Laboratoire Spécification et Vérification
Ecole Normale Supérieure de Cachan, France

March 24, 2011

This document contains a revised version of lecture notes for the advanced course "Decidable problems for counter systems" that has been delivered during ESSLLI'2010, Copenhagen, August 2010. A similar course (level M2) has been given at Département d'Informatique de l'Institut Galilée (Université Paris-Nord) for the master "Modélisation Informatique des Connaissances et du Raisonnement" (MICR), 2009/2010 and for the master MPRI, course 2.9, 2010/2011. I would like to thank Alain Finkel, Valentin Goranko, Arnaud Sangnier and Marc Zeitoun for their suggestions and remarks about a preliminary version of this document. I wish also to thank the attendees of the ESSLLI course and the master students for their questions, remarks and suggestions that helped me to improve the slides and the current version of the notes.

Contents

Preface

Model-checking is a well-known approach to verifying behavioral properties of computing systems that has been very successful in the verification of finite-state systems, see e.g. [McM93, CGP00, BBF+01]. The situation is different for infinite-state systems. Despite that numerous symbolic representations have been proposed to deal with such systems (see e.g. timed automata [AD94]), their formal verification remains a difficult problem. Many general formalisms referring to infinite-state systems have an undecidable model-checking problem. Sometimes, decidability can be regained by considering subproblems of the general problem. The class of counter systems is an example of such a formalism. Counter systems have many applications in formal verification. Their ubiquity stems from their use as operational models of numerous infinite-state systems, including for instance broadcast protocols [FL02], programs with pointer variables (see [BFLS06, BBH+06]) and logics for data words [BMS+06]. Even the case of a single counter has found some applications in the verification of cryptographic protocols [LLT05] and the validation of XML streams [CR04]. However, numerous model-checking problems for counter systems, such as reachability, are known to be undecidable. This does not end the story since many subclasses of counter systems admit a decidable reachability problem such as reversal-bounded counter automata [Iba78, ISD+00] and flat counter automata [Boi98, CJ98, FL02]. These two classes of systems admit reachability sets effectively definable in Presburger arithmetic (assuming some additional conditions, unspecified herein).

This course is dedicated to the presentation of decidable problems for counter systems. We develop techniques for various classes of counter systems (vector addition systems, reversal-bounded counter systems, counter systems with errors, etc.) and for various problems including reachability problems, and model-checking with linear-time temporal logics. Mainly, we focus on decision procedures based on Presburger arithmetic, on the direct analysis of runs and on the automata-based approach when it is relevant. Because of lack of space and time, the course does not deal with: proof techniques based on theory of well-structured transition systems [FS01] and decidability proof for the reachability problem on Petri nets [Reu90] (plus many other topics).

Chapter 1

Introduction to Counter Systems

In this chapter, we present the class of counter systems as well as several standard subclasses, some of them being further studied in the rest of the document. For instance, this includes Minsky machines, relational counter systems and other classes obtained by restriction (on the control graph for example). Several decision problems are also defined. Moreover, this chapter presents some basic material about Presburger arithmetic, the first-order theory of the set of natural numbers with addition. Indeed, this is a fundamental theory that is not only instrumental to define counter systems but although it is central to show the decidability of several problems on subclasses of counter systems.

1.1 Introductory Example: Phone Controller

We start by presenting a simple computer system, namely a phone controller [CJ98], in order to illustrate the goal we pursue by introducing and studying counter systems. Figure 1.1 presents the phone controller from [CJ98, CC00].

- ⋆ Control state q_1 is the initial and final contro state.
- ⋆ x_1 is the number of coins which have been inserted.
- ⋆ x_2 measures the total communication time. Herein, we assume that each coin allows a communication for exactly one time unit. Total communication time is therefore the number of time units spent for communication.
- ⋆ x_1' [resp. x_2'] is the next value of x_1 [resp. x_2].
- ⋆ The controller interacts with the environment including the phone box. It can receive or send messages. Messages followed by a question mark are received by the controller and messages followed by an exclamation mark are sent by the controller. The message 'coin?' means that the controller receives the information that a coin has been inserted. Similarly, 'signal?' means that a communication time unit has been used. In this example, the messages are considered in order to clarify the role of the different transitions. Nevertheless, a complete treatment would deserve to introduce channels or similar objects.

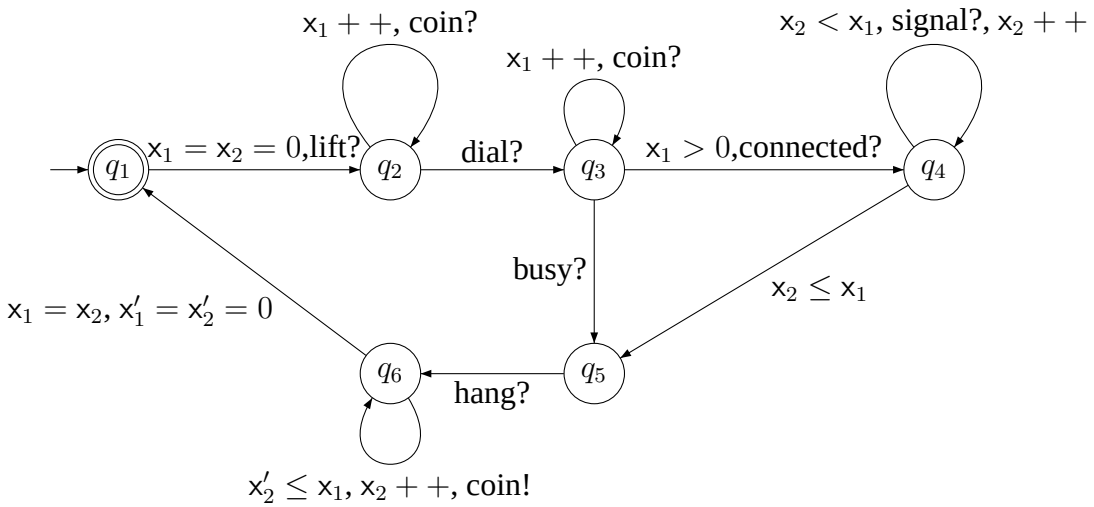


Figure 1.1: Phone controller

- ⋆ A configuration of the controller is a triple (q, n_1, n_2) where q is a control state among $\{q_1, \ldots, q_6\}$ and n_1 [resp. n_2] is the value of x_1 [resp. x_2]. It entirely describes the state of the controller. The control state q_1 is both an initial state and a final state.

- ⋆ An execution is a (possibly infinite) sequence of configurations, constrained by transitions of the controller. Observe that different executions are possible, depending for instance on the received messages (signals). Here is one execution:

$$(q_1, 0, 0), (q_2, 0, 0), (q_2, 1, 0), (q_2, 2, 0), (q_2, 3, 0), \ldots$$

  For the sake of simplicity, the described system has no bound on the number of inserted coins.

- ⋆ The system presented in Figure 1.1 is a finite and concise representation of an infinite labeled transition system (its interpretation). Moreover, this is obviously an abstraction of a more complex system and refinements are still possible, for instance the system could be completed in order to take into account situations when a break of communication occurs.

Here are examples of properties that one may wish to specify about the system, assuming that the initial configuration is $(q_1, 0, 0)$. For each property, we provide a specification in natural language and in some temporal logic (dialect close to CTL*).

- ⋆ Total communication time is never greater than the number of inserted coins (true):

$$\mathtt{A}\,\mathtt{G}\,\neg(x_2 > x_1).$$

- ⋆ For all the executions, the number of coins is infinitely often equal to zero (false):

$$\mathtt{A}\,\mathtt{G}\,\mathtt{F}\,(x_1 = 0).$$

- ⋆ There is an execution of the controller such that the total communication time is always equal to zero (true):
$$\mathtt{E}\,\mathtt{G}\,(\mathsf{x}_2 = 0).$$
From q_1, it is sufficient to reach q_2 (in one step) and then to loop on q_2.

- ⋆ Whenever the communication is over, eventually the controller can reach the initial configuration (true):
$$\mathtt{A}\,\mathtt{G}\,(q_5 \Rightarrow \mathtt{F} q_1).$$

- ⋆ Whenever the control state q_1 is reached, $\mathsf{x}_1 = \mathsf{x}_2 = 0$ and conversely (false):
$$\mathtt{A}\,\mathtt{G}(q_1 \Leftrightarrow (\mathsf{x}_1 = 0 \wedge \mathsf{x}_2 = 0)).$$

The systems introduced in the sequel can be viewed as finite-state automata augmented with counters (variables interpreted as natural numbers). Transitions are labelled by arithmetical constraints on counters, and possibly by letters from a finite alphabet. So, the phone controller (without messages) shall be clearly an instance of *counter system*. Before defining the class of counter systems, and fragments allowing us to get decidable verification tasks, we present the Minsky machines that use elementary operations and guards on transitions but still they are Turing-complete.

1.2 Minsky Machines

A *Minsky machine* [Min67] can be viewed as a finite-state automaton with two counters. Each counter stores a nonnegative integer. The operations on counters are the following

- ⋆ Check whether the counter is zero (*zero-test*).
- ⋆ Increment the counter by one (*increment*).
- ⋆ Decrement the counter by one if nonzero (*decrement*).

A Minsky machine is defined as a set of n intructions on two counters C_1 and C_2. The lth instruction has one of the form below ($i \in \{1, 2\}$, $l' \in \{1, \ldots, n\}$):

l: $\mathsf{C}_i := \mathsf{C}_i + 1$; goto l'

l: if $\mathsf{C}_i = 0$ then goto l' else $\mathsf{C}_i := \mathsf{C}_i - 1$; goto l''.

Configurations are elements of $\{1, \ldots, n\} \times \mathbb{N} \times \mathbb{N}$ and the initial configuration is $(1, 0, 0)$. A *computation* is a sequence (finite or infinite) of configurations starting from the initial configuration and such that two successive configurations respect the instructions. Consider the Minsky machine described by the two instructions below:

1: $\mathsf{C}_1 := \mathsf{C}_1 + 1$; goto 2

2: $\mathsf{C}_2 := \mathsf{C}_2 + 1$; goto 1

Here is the unique computation:

$$(1,0,0) \to (2,1,0) \to (1,1,1) \to (2,2,1) \to (1,2,2) \to (2,3,2) \dots$$

We present below a classical decision problem for Minsky machines.
HALTING PROBLEM

Input: a Minsky machine M;

Question: is there a finite computation that reaches the instruction n?

An alternative way to define the halting problem is to assume that instruction n halts the machine (and therefore it has a special instruction).

Theorem 1.2.1. [Min67, pp. 255–258] For every Turing machine, there is a Minsky machine that simulates it.

Here are the different steps of the simulation (see also more details at `http://en.wikipedia.org/wiki/Counter_machine` or in [Min67]).

1. A Turing machine can be simulated by two stacks: the infinite tape is cut in half. For instance, moving the head left or right is equivalent to popping a bit from one stack and pushing it onto the other.
2. A stack over a binary alphabet can be simulated by two counters. One counter contains the binary representation of the bits on the stack. For instance, pushing a 1 is equivalent to doubling and adding 1, assuming that in the binary representation the least significant bit is on the top. To do so, the second counter is auxiliary. Similarly, popping a zero is equivalent to dividing by two.
3. Four counters can be simulated by two counters. The counter values $(a,b,c,d) \in \mathbb{N}^4$ are encoded by the counter value $2^a 3^b 5^c 7^d$. For instance, checking the third counter is zero is equivalent to dividing by 5 and see what the remainder is. The second counter is again auxiliary.

As a consequence, we get the following undecidability result based on the undecidability of halting problem for Turing machines [Tur36].

Theorem 1.2.2. [Min67] The halting problem is undecidable.

It is also possible to design nondeterministic Minsky machines by allowing nondeterministic choice after incrementation and decrementation. The instructions are of the forms below:

l: $\mathsf{C}_i := \mathsf{C}_i + 1$; goto l' or goto l''

l: if $\mathsf{C}_i = 0$ then goto l' else $\mathsf{C}_i := \mathsf{C}_i - 1$; goto l''_0 or goto l''_1.

Another classical decision problem for nondeterministic Minsky machines is the following.
RECURRENCE PROBLEM:

Input: a nondeterministic Minsky machine M;

Question: is there an infinite computation with instruction 1 occurring infinitely often?

Theorem 1.2.3. [AH94] The recurrence problem is Σ^1_1-complete.

Σ^1_1-hard problems in the analytical hierarchy are understood as highly undecidable [Rog67].

Toward counter systems. Even though Minsky machines have a strong computational power, it is unlikely that one may wish to solve decision problems by programming Minsky machines. Moreover, undecidability of the halting problem prevents us from encoding decidable problems by Minsky machines without designing subclasses of Minsky machines with desirable computational properties. That is why, we shall introduce the class of counter systems (subsuming the class of Minsky machines) that is of more practical use, in particular by allowing more flexibility and by admitting a richer set of instructions (in the same way, it makes sense to design convenient programming languages). Nevertheless, we shall impose restrictions on such counter systems in order to design classes with decidable problems.

1.3 A Fundamental Decidable Theory: Presburger Arithmetic

Roughly speaking, Presburger arithmetic is the first-order theory of the structure $(\mathbb{N}, +)$ shown decidable in [Pre29] (which contrasts with Peano arithmetic). This logical formalism is used to define sets of tuples of natural numbers. Moreover, in this course, it will serve two purposes. Firstly, in the definition of counter systems, Presburger arithmetic is used as a language to define guards and actions (updates on counter values) on transitions. Secondly, each formula from Presburger arithmetic defines a set of tuples (related to the set of valuations that make true the formula) and Presburger arithmetic is therefore a means to represent and manipulate symbolically infinite sets of tuples of natural numbers. This section is dedicated to the basics on Presburger arithmetic and to the main properties we shall use in the sequel.

1.3.1 Basics on tuples of natural numbers

We write $\mathbb{N}$ [resp. $\mathbb{Z}$] for the set of natural numbers [resp. integers] and $[m, m']$ with $m, m' \in \mathbb{Z}$ to denote the set $\{j \in \mathbb{Z} : m \leq j \leq m'\}$. Given a dimension $n \geq 1$ and $a \in \mathbb{Z}$, we write $\vec{a}$ to denote the vector with all values equal to a. For $\vec{x} \in \mathbb{Z}^n$, we write $\vec{x}(1), \ldots, \vec{x}(n)$ for the entries of $\vec{x}$. For $\vec{x}, \vec{y} \in \mathbb{Z}^n$, $\vec{x} \preceq \vec{y} \stackrel{\text{def}}{\Leftrightarrow}$ for $i \in [1, n]$, we have $\vec{x}(i) \leq \vec{y}(i)$. We also write $\vec{x} \prec \vec{y}$ when $\vec{x} \preceq \vec{y}$ and $\vec{x} \neq \vec{y}$.

In the sequel, we shall regularly use Dickson's Lemma [Dic13] that states that for any ω-sequence $\vec{x_0}, \vec{x_1}, \ldots$ of tuples in $\mathbb{N}^n$, there are $i < j$ such that $\vec{x_i} \preceq \vec{x_j}$.

1.3.2 Definition

Let $\mathrm{VAR} = \{\mathsf{x}, \mathsf{y}, \mathsf{z}, \ldots\}$ be a countably infinite of *variables*. *Terms* are defined by the grammar below:

$$t ::= 0 \mid 1 \mid \mathsf{x} \mid t + t$$

where $\mathsf{x} \in \mathrm{VAR}$ and 0 and 1 are distinguished constants (interpreted by zero and one respectively). For $k \geq 1$, we write $k\mathsf{x}$ instead of $\mathsf{x} + \cdots + \mathsf{x}$ (k times). *Presburger formulae* are defined by the grammar below:

$$\varphi ::= t \equiv_k t \mid t < t \mid \neg\varphi \mid \varphi \wedge \varphi \mid \exists \mathsf{x}\, \varphi \mid \forall \mathsf{x}\, \varphi$$

where $k \geq 2$. As usual, an occurrence of the variable x in the formula φ is *free* if it does not occur in the scope of either $\exists\mathsf{x}$ or $\forall\mathsf{x}$. Otherwise, the occurrence is *bound*. For instance, in $\mathsf{x}_1 < \mathsf{x}_2$, all

the occurrences of the variables are free. In $(\exists\, \mathsf{x}_1\, \mathsf{x}_2\, \mathsf{x}_1 < \mathsf{x}_2) \wedge \mathsf{x}_1 < \mathsf{x}_2$, each variable has a free occurrence and a bound occurrence.

A *valuation* $\mathbf{val}$ is a map $\mathrm{VAR} \to \mathbb{N}$ and it can be extended to the set of all terms as follows: $\mathbf{val}(0) = 0$, $\mathbf{val}(1) = 1$ and $\mathbf{val}(t+t') = \mathbf{val}(t) + \mathbf{val}(t')$. The satisfaction relation for Presburger arithmetic is equipped with a valuation witnessing that Presburger formulae are interpreted over the structure $(\mathbb{N}, +)$.

- $\star$ $\mathbf{val} \models t \equiv_k t' \stackrel{\text{def}}{\Leftrightarrow}$ there is $n \in \mathbb{Z}$ such that $kn + \mathbf{val}(t) = \mathbf{val}(t')$,
- $\star$ $\mathbf{val} \models t < t' \stackrel{\text{def}}{\Leftrightarrow} \mathbf{val}(t) < \mathbf{val}(t')$,
- $\star$ $\mathbf{val} \models \neg\varphi \stackrel{\text{def}}{\Leftrightarrow} \mathbf{val} \not\models \varphi$,
- $\star$ $\mathbf{val} \models \varphi \wedge \varphi' \stackrel{\text{def}}{\Leftrightarrow} \mathbf{val} \models \varphi$ and $\mathbf{val} \models \varphi'$,
- $\star$ $\mathbf{val} \models \exists \mathsf{x}\, \varphi \stackrel{\text{def}}{\Leftrightarrow}$ there is $n \in \mathbb{N}$ such that $\mathbf{val}[\mathsf{x} \mapsto n] \models \varphi$ where $\mathbf{val}[\mathsf{x} \mapsto n]$ is equal to $\mathbf{val}$ except that x is mapped to n,
- $\star$ $\mathbf{val} \models \forall \mathsf{x}\, \varphi \stackrel{\text{def}}{\Leftrightarrow}$ for every $n \in \mathbb{N}$, we have $\mathbf{val}[\mathsf{x} \mapsto n] \models \varphi$.

Equality between two terms, written $t = t'$, can be expressed by $\neg(t < t' \vee t' < t)$. Observe also that $t \equiv_k t'$ is equivalent to the formula below (x is a variable that does not occur in t and t'):

$$\exists\, \mathsf{x}\, (t = k\mathsf{x} + t' \vee t' = k\mathsf{x} + t)$$

As an exercise, we invite the reader to check that 0, 1 and $<$ can be removed from the above definitions without changing the expressive power of the formulae.

In the sequel, we assume that the variables in VAR are linearly ordered by their indices. So, any valuation restricted to $n \geq 1$ variables can be viewed as a tuple in $\mathbb{N}^n$.

Given a Presburger formula φ, we write $free(\varphi)$ to denote the free variables occurring in φ. Any formula with $n \geq 1$ free variables $\mathsf{x}_1, \ldots, \mathsf{x}_n$ defines a set of n-tuples as follows:

$$\mathrm{REL}(\varphi) \stackrel{\text{def}}{=} \{(\mathbf{val}(\mathsf{x}_1), \ldots, \mathbf{val}(\mathsf{x}_n)) \in \mathbb{N}^n : \mathbf{val} \models \varphi\}.$$

For instance, $\mathrm{REL}(\mathsf{x}_1 < \mathsf{x}_2) = \{(n, n') \in \mathbb{N}^2 : n < n'\}$. Similarly, the set of odd natural numbers can be defined by the formula below:

$$\exists \mathsf{y}\, \mathsf{x} = \mathsf{y} + \mathsf{y} + 1$$

The set $\{0\}$ can be defined by the formula $\mathsf{x} = \mathsf{x} + \mathsf{x}$.

A formula φ is *satisfiable* (in Presburger arithmetic) whenever there is a valuation $\mathbf{val}$ such that $\mathbf{val} \models \varphi$. Similarly, a formula φ is *valid* (in Presburger arithmetic) when for all valuations $\mathbf{val}$, we have $\mathbf{val} \models \varphi$. When φ has no free variables, satisfiability and validity are equivalent notions. Moreover, assuming that φ has at least one free variable, satisfiability is equivalent to the nonemptiness of $\mathrm{REL}(\varphi)$. Furthermore, a formula φ with n free variables $\mathsf{x}_1, \ldots, \mathsf{x}_n$ is valid iff $\forall\, \mathsf{x}_1\, \cdots \forall\, \mathsf{x}_n \varphi$ is valid/satisfiable.

Two formulae are *equivalent* (in Presburger arithmetic) whenever they define the same set of tuples.

Theorem 1.3.1. [Pre29] (I) The satisfiability problem for Presburger arithmetic is decidable. (II) Every Presburger formula is equivalent to a Presburger formula without first-order quantification.

Theorem 1.3.1(II) takes advantage of atomic formulae of the form $t \equiv_k t'$ that contain an implicit quantification. Removing atomic formulae of the form $t \equiv_k t'$ does not change the expressive power but the equivalence in Theorem 1.3.1(II) would not hold in that case. Moreover, in (II) above, the equivalent formula can be effectively built witnessing the quantifier elimination property. Here are a few known tools dealing with satisfiability based on automata: MONA [BKR96], LASH [BJW01] and TAPAS [LP09] to quote a few of them. In subsequent developments, we mainly consider quantifier-free Presburger formulae, which does not restrict the expressive power but may modify complexity issues. It is worth noting that the first-order theory of $(\mathbb{N}, \times)$ is decidable too (known as *Skolem arithmetic*) whereas the first-order theory of $(\mathbb{N}, \times, +)$ is undecidable (see e.g. [Tar53]). Observe also that $(\mathbb{Z}, <, +)$ is decidable [Pre29].

Satisfiability problem for Presburger arithmetic can be solved in triple exponential time [Opp78] by analyzing the quantifier elimination procedure described in [Coo72]. Besides, satisfiability problem for Presburger arithmetic is shown 2EXPTIME-hard in [FR74] and in 2EXPSPACE in [FR79]. An exact complexity charaterization is provided in [Ber80] (double exponential time on alternating Turing machines with linear amounts of alternations). Due to the wide range of applications for Presburger arithmetic, computational complexity of numerous fragments has been also characterized, see e.g.,[Grä88]. Moreover, its restriction to quantifier-free formulae is NP-complete [Pap81] (see also [BT76]).

As mentioned earlier, Presburger arithmetic will be shown to be an essential logical formalism to define the class of counter systems, providing symbolic constraints between counter values. Furthermore, it is used in many occasions, for instance to verify infinite-state systems (see e.g., [Ler03, Sch07]), to express constraints on the number of event occurrences [BEH95], on XML documents [ZL03, SSM07], to define linear-time temporal logic LTL with counters [CC00, DG08] (see Chapter 2) or graded modal logics and description logics (see e.g. [Fin72, HB91, DL10]). This list is certainly not exhaustive.

1.3.3 Semilinear sets

A *linear set* X (of dimention $k \geq 1$) is defined as a subset of $\mathbb{N}^k$ for which there exists a *basis* $\vec{b} \in \mathbb{N}^k$ and a finite set of *periods* $P = \{\vec{p}_1, \ldots, \vec{p}_m\} \subseteq \mathbb{N}^k$ such that

$$X = \{\vec{b} + \sum_{i=1}^{i=m} n_i \vec{p}_i : n_1, \ldots, n_m \in \mathbb{N}\}$$

For example, the set of even numbers $\{0 + i \times 2 : i \in \mathbb{N}\}$ is a linear subset of $\mathbb{N}$ (dimension 1). Similarly, $\{1 + i \times 2 : i \in \mathbb{N}\}$ is a linear set with $\vec{b} = 1$ and with unique period 2. A *semilinear set* is defined as a finite union of linear sets. Each semilinear set can be represented by a finite set of pairs of the form $(\vec{b}, P)$. Here is a linear set of dimension 2:

$$\left\{ \begin{pmatrix} 3 \\ 4 \end{pmatrix} + i \times \begin{pmatrix} 2 \\ 5 \end{pmatrix} + j \times \begin{pmatrix} 4 \\ 7 \end{pmatrix} : i, j \in \mathbb{N} \right\}$$

By contrast, one can show that the sets below are not semilinear:

$$\{2^i : i \in \mathbb{N}\} \quad \{i^2 : i \in \mathbb{N}\}$$

By way of example, let us show that $X = \{2^i : i \in \mathbb{N}\}$ is not semilinear. The proof is *ad absurdum*. Suppose that X is semilinear. Since X is an infinite set, there exist a basis $b \in \mathbb{N}$ and period(s) $p_1, \ldots, p_m \in (\mathbb{N} \setminus \{0\})$ ($m \geq 1$) such that $Y = \{b + \sum_{i=1}^{i=m} n_i p_i : n_1, \ldots, n_m \in \mathbb{N}\} \subseteq X$. There exists $2^\alpha \in Y$ such that $p_1 < 2^\alpha$. By definition of Y, we have $2^\alpha + p_1 \in Y$ but $2^\alpha < 2^\alpha + p_1 < 2^{\alpha+1}$, which leads to a contradiction.

Theorem 1.3.2. [GS66] The class of semilinear sets are effectively closed under union, intersection and complementation.

Closure by union is immediate from the definition of semilinear sets.

The class of semilinear sets happen to be much more interesting since it contains exactly the sets of tuples defined by Presburger formulae.

Theorem 1.3.3. [GS66] Semilinear sets coincide with sets definable by Presburger formulae, i.e.,

1. for every Presburger formula φ with $n \geq 1$ free variables, $\mathrm{REL}(\varphi)$ is a semilinear subset of $\mathbb{N}^n$.
2. for every semilinear set $X \subseteq \mathbb{N}^n$, there is a Presburger formula φ such that $X = \mathrm{REL}(\varphi)$.

Observe that if X_1 and X_2 are semilinear subsets of $\mathbb{N}^n$ such that $X_1 = \mathrm{REL}(\varphi_1)$ and $X_2 = \mathrm{REL}(\varphi_2)$, then $X_1 \cap X_2 = \mathrm{REL}(\varphi_1 \wedge \psi_2)$ and $\mathbb{N}^n \setminus X_1 = \mathrm{REL}(\neg\varphi_1)$. An alternative proof can be found in [Kra02].

For example, a Presburger formula for the semilinear set

$$\left\{ \begin{pmatrix} 3 \\ 4 \end{pmatrix} + i \times \begin{pmatrix} 2 \\ 5 \end{pmatrix} + j \times \begin{pmatrix} 4 \\ 7 \end{pmatrix} : i, j \in \mathbb{N} \right\}$$

is $\exists\, \mathsf{y}, \mathsf{y}'\ (\mathsf{x}_1 = 3 + 2\mathsf{y} + 4\mathsf{y}' \wedge \mathsf{x}_2 = 4 + 5\mathsf{y} + 7\mathsf{y}')$.

We conclude this section about Presburger arithmetic and semilinear sets by a result relating commutative images of context-free languages and semilinear sets. Let $\Sigma = \{a_1, \ldots, a_k\}$ equipped with an arbitrary linear ordering of the letters, say $a_1 < \cdots < a_k$. Given a word $u \in \Sigma^*$, its *Parikh image* is defined as a tuple $\Pi(u) \in \mathbb{N}^k$ such that for $i \in [1, k]$, $\Pi(u)(i)$ is the number of occurrences of the letter a_i in the word u. For instance, the Parikh of the word $abaab$ under the ordering $a < b$ is the tuple $\begin{pmatrix} 3 \\ 2 \end{pmatrix}$. Naturally, the *Parikh image* of the language $\mathrm{L} \subseteq \Sigma^*$ is the set $\{\Pi(u) \in \mathbb{N}^k : u \in \mathrm{L}\}$. Parikh's remarkable result states that the Parikh image of any context-free language is semilinear [Par66] and that its representation is effectively computable from pushdown automata.

This result can be refined by imposing constraints on the size of the representation in terms of basis and periods. Let $\mathcal{A}$ be a finite-state automaton with set of control states Q and finite alphabet Σ. The Parikh image of $\mathrm{L}(\mathcal{A})$, a subset of $\mathbb{N}^{\mathrm{card}(\Sigma)}$, is a finite union $X_1 \cup \cdots \cup X_m$ of linear sets of the form $X_i = \{\vec{b} + \sum_{j=1}^{h} y_j \vec{p}_j : y_j \geq 0\}$ where $\vec{b}$ and each $\vec{p}_j$ is in $\{0, \ldots, \mathrm{card}(Q)\}^{\mathrm{card}(\Sigma)}$ by [SSMH04, Theorem 1]. Consequently, h is bounded by $(\mathrm{card}(Q) + 1)^{\mathrm{card}(\Sigma)}$.

1.3.4 Fragments

We define below several fragments of Presburger arithmetic that allow us to define fragments of $\mathrm{CLTL}(\mathrm{PrA})$ in Chapter 2. Formulae of the *difference logic* DL are the following:

$$\varphi ::= \mathsf{x} \sim \mathsf{y} + d \mid \mathsf{x} \sim d \mid \varphi \wedge \varphi \mid \neg\varphi$$

with $d \in \mathbb{Z}$, $\sim \in \{<, >, =\}$. Formulae of the logic DL^+ are obtained from those for DL by adding periodicity constraints $\mathsf{x} \equiv_k c$ and $\mathsf{x} \equiv_k \mathsf{y} + c$ with $c \in \mathbb{N}$ and $k \geq 1$. Finally, the formulae of *quantifier-free Presburger arithmetic* QFP are defined as follows:

$$\varphi ::= \sum_{i \in I} a_i \mathsf{x}_i \sim d \mid \sum_{i \in I} a_i \mathsf{x}_i \equiv_k c \mid \varphi \wedge \varphi \mid \neg\varphi$$

with the a_i's in $\mathbb{Z}$.

Let us conclude this section by introducing the fragment IPC* made of qualitative constraints only, for which formulae are defined as follows:

$$\varphi ::= \mathsf{x} \sim d \mid \mathsf{x} \sim \mathsf{y} \mid \mathsf{x} \equiv_k k' \mid \neg\varphi \mid \varphi \wedge \varphi$$

with $d \in \mathbb{Z}$, $k' < k \in \mathbb{N}$, $\sim \in \{<, >, =, \leq, \geq\}$. Such constraints can be found in formalisms dealing with calendars or in DATALOG with integer periodicity constraints.

1.4 Classes of Counter Systems

1.4.1 Counter systems

A counter system $\mathcal{S}$ is defined below as a finite-state automaton equipped with counters, i.e. variables interpreted over $\mathbb{N}$. In full generality, the counters are governed by constraints that can be expressed by Presburger formulae. Minsky machines form a special class of counter systems and therefore most interesting problems on counter systems happen to be undecidable. However, we shall study important subclasses of counter systems for which decidability can be regained for various decision problems.

Definition 1.4.1. A *counter system* $\mathcal{S} = (Q, n, \delta)$ (of dimension n) is a structure such that

- Q is a nonempty finite set of *control states* (a.k.a. *locations*),
- $n \geq 1$ is the *dimension* of the system, i.e. the number of counters; we assume that the counters are represented by the variables $\mathsf{x}_1, \ldots, \mathsf{x}_n$,
- δ is the *transition relation* defined as a finite set of triples of the form

  $$(q, \varphi, q')$$

  where q, q' are control states and φ is a Presburger formula whose free variables are among $\mathsf{x}_1, \ldots, \mathsf{x}_n, \mathsf{x}'_1, \ldots, \mathsf{x}'_n$

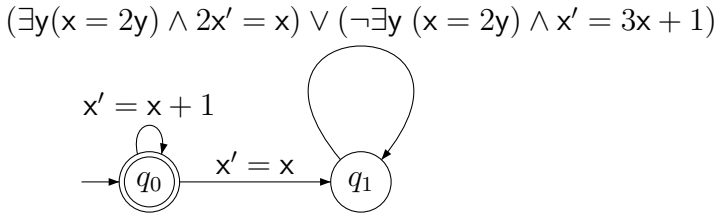


Figure 1.2: An example of counter system

∇

Elements $t = (q, \varphi, q')$ are called *transitions* and are often represented by $q \xrightarrow{\varphi} q'$. As usual, by convention, prime variables are intended to be interpreted as the next values of the unprimed variables. Moreover, observe that a counter system has no initial control state and no final control state but in the sequel we shall introduce such control states on demand. It is certainly possibly to propose an alternative definition without control states and to encode them by a new counter, for instance. However, when infinite-state transition systems arise in the modeling of computational processes, there is often a natural factoring of each system state into a control component and a memory component, where the set of control states (locations) is typically finite.

Figure 1.4.1 contains a counter system (augmented with an initial control state and a final control state). It is related to the famous Collatz problem (see e.g. `http://mathworld.wolfram.com/CollatzProblem.html`). The role of control state q_0 is to compute an arbitrary counter value before reaching the control state q_1. At the control-state q_1, if the counter value is even, then divide by two the counter value. Otherwise, multiply by 3 and add 1. It is open whether whenever the system enters in the control-state q_1, eventually it reaches the counter value 1.

A *configuration* of the counter system $\mathcal{S} = (Q, n, \delta)$ is defined as a pair $(q, \vec{x}) \in Q \times \mathbb{N}^n$. Given two configurations $(q, \vec{x})$, $(q', \vec{x'})$ and a transition $t = q \xrightarrow{\varphi} q'$, we write $(q, \vec{x}) \xrightarrow{t} (q', \vec{x'})$ whenever $\mathbf{val}_{\vec{x},\vec{x'}} \models \varphi$ and for $i \in [1, n]$, $\mathbf{val}_{\vec{x},\vec{x'}}(\mathsf{x}_i) \stackrel{\text{def}}{=} \vec{x}(i)$ and $\mathbf{val}_{\vec{x},\vec{x'}}(\mathsf{x}'_i) \stackrel{\text{def}}{=} \vec{x'}(i)$. The operational semantics of counter systems updates configurations, and runs of such systems are essentially sequences of configurations.

Every counter system $\mathcal{S} = (Q, n, \delta)$ induces a (possibly infinite) graph made of configurations. Indeed, all the interesting problems on counter systems can be formulated on its *transition system*.

Definition 1.4.2. Given a counter system $\mathcal{S} = (Q, n, \delta)$, its *transition system* $T(\mathcal{S}) = (S, \rightarrow)$ is a graph such that $S = Q \times \mathbb{N}^n$ and $\rightarrow \subseteq S \times S$ such that $((q, \vec{x}), (q', \vec{x'})) \in \rightarrow \stackrel{\text{def}}{\Leftrightarrow}$ there exists a transition $t \in \delta$ such that $(q, \vec{x}) \xrightarrow{t} (q', \vec{x'})$. ∇

As usual, $\xrightarrow{*}$ denotes the reflexive and transitive closure of the binary relation $\rightarrow$.

The class of counter systems is quite general and very often it makes sense to label the transitions by Presburger formulae that can be decomposed by a *guard* (constraints on the current

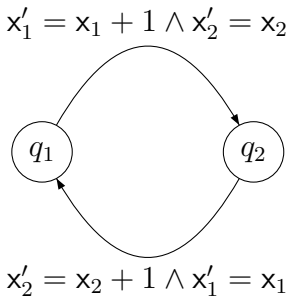


Figure 1.3: A Minsky machine

counter values) and an *update function* (constraints on the way the new counter values are computed from the previous ones).

Given a counter system $\mathcal{S}$, a *run* ρ is a nonempty (possibly infinite) sequence

$$\rho = (q_0, \vec{x_0}), \ldots, (q_k, \vec{x_k}), \ldots$$

of configurations such that two consecutive configurations are in the relation $\rightarrow$ from $T(\mathcal{S})$. $(q_0, \vec{x_0})$ is called the *initial* configuration of ρ. A run can be alternatively represented by an initial configuration and a sequence of transitions, assuming that firability holds true for the intermediate configurations and Presburger formulae labelling the transitions define deterministic relations (as in VASS, see Section 1.4.3).

Extensions. Sometimes, we may slightly extend the model of counter systems, for instance by labelling the transitions by a letter from a finite alphabet or by allowing counter values in $\mathbb{Z}$ or $\mathbb{R}$. In those cases, we need to interpret the formulae on the adequate structures. Similarly, the transition system can be also defined as a labelled transition system by labelling transition by letters or by transitions. In the sequel, we shall make it clear when we need these slight extensions.

In Figure 1.4.1, we present a graphical representation of the counter system corresponding to the Minsky machine made of the two instructions below:

1: $\mathsf{C}_1 := \mathsf{C}_1 + 1$; goto 2

2: $\mathsf{C}_2 := \mathsf{C}_2 + 1$; goto 1

1.4.2 Decision problems

In this section, we enumerate a list of standard decision problems about counter systems. They are mainly related to reachability questions. The list is certainly not exhaustive (model-checking problems can be found in Chapter 2).

REACHABILITY PROBLEM:

Input: a counter system $\mathcal{S}$ and two configurations $(q, \vec{x})$ and $(q', \vec{x'})$.

Question: is there a finite run with initial configuration $(q, \vec{x})$ and final configuration $(q', \vec{x'})$?

CONTROL STATE REACHABILITY PROBLEM:

Input: a counter system $\mathcal{S}$, a configuration $(q, \vec{x})$ and a control state q_f.

Question: is there a finite run with initial configuration $(q, \vec{x})$ and whose final configuration has control state q_f?

CONTROL STATE REPEATED REACHABILITY PROBLEM:

Input: a counter system $\mathcal{S}$, a configuration $(q, \vec{x})$ and a control state q_f.

Question: is there an infinite run with initial configuration $(q, \vec{x})$ such that the control state q_f is repeated infinitely often?

COVERING PROBLEM:

Input: a counter system $\mathcal{S}$ and two configurations $(q, \vec{x})$ and $(q', \vec{x'})$.

Question: is there a finite run with initial configuration $(q, \vec{x})$ and whose final configuration is $(q', \vec{x''})$ with $\vec{x'} \preceq \vec{x''}$?

BOUNDEDNESS PROBLEM:

Input: a counter system $\mathcal{S}$ and a configuration $(q, \vec{x})$.

Question: is the set $\{(q', \vec{x'}) \in Q \times \mathbb{N}^n : (q, \vec{x}) \xrightarrow{*} (q', \vec{x'})\}$ finite?

TERMINATION PROBLEM:

Input: a counter system $\mathcal{S}$ and a configuration $(q, \vec{x})$.

Question: is there an infinite run with initial configuration $(q, \vec{x})$?

Designing algorithms for counter systems can be helpful for instance to verify programs with pointers [BFN04, BBH$^+$06, FLS09], broadcast protocols [EFM99] or systems with energy constraints [BFL$^+$08], see also its use for discrete timed automata (with digital clocks) [DPK03].

1.4.3 Various classes

In this section, we introduce several subclasses of counter systems by restricting the general definition provided above. Additional requirements can be of distinct nature:

- $\star$ restriction on syntactic ressources (number of counters, Presburger formulae etc.)
- $\star$ restriction on the control graph (e.g. flatness),
- $\star$ semantical restrictions (reversal-boundedness, etc.)

Other subclasses will be considered in this course, but we shall consider them in due course. Our intention in this section is to provide typical examples and results, without being necessarily exhaustive. In Figure 1.4.3, we present (syntactic) inclusions between classes of counter systems (CS stands for 'counter systems' and CA for 'counter automata').

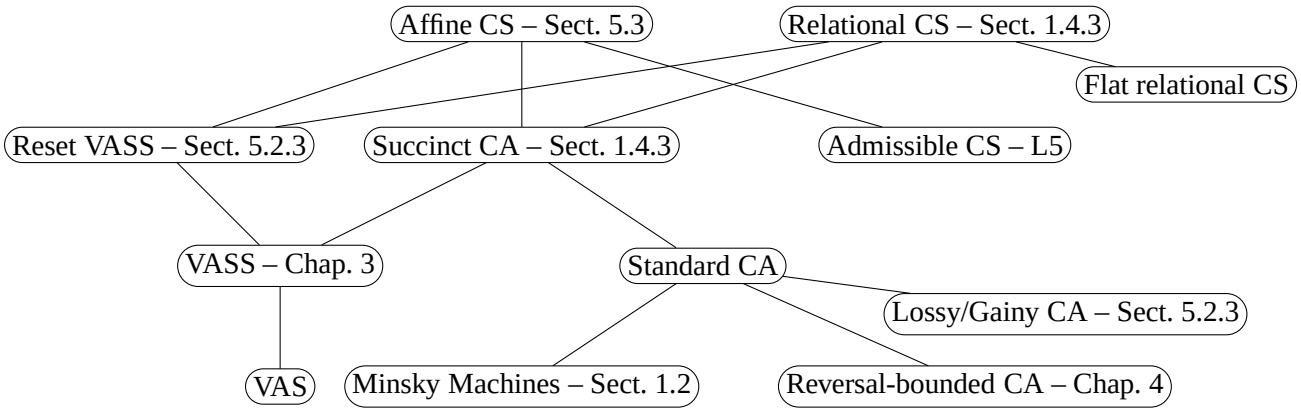


Figure 1.4: Classes of Counter Systems

Relational counter systems

A *relational counter system* $\mathcal{S} = (Q, n, \delta)$ is a counter system such that for each transition $q \xrightarrow{\varphi} q' \in \delta$, the Presburger formula φ is a conjunction of atomic formulae of the form

- $\star$ either $\mathsf{x} \sim \mathsf{y} + c$,
- $\star$ or $\mathsf{x} \sim c$,

where $\mathsf{x}, \mathsf{y} \in \{\mathsf{x}_1, \ldots, \mathsf{x}_n, \mathsf{x}'_1, \ldots, \mathsf{x}'_n\}$, $c \in \mathbb{Z}$ and $\sim \in \{\geq, \leq, =, >, <\}$. It is worth observing that other Presburger formulae can define relations between counter values (instead of functions). However, in the sequel, relational counter systems are understood with the above meaning (more general classes are considered in [BIK10]).

Here is an example of formula labelling a transition for $n = 2$: $\varphi = (\mathsf{x}_1 + 1 < \mathsf{x}'_1) \wedge (\mathsf{x}_2 - 3 = \mathsf{x}'_2)$. The phone controller in Figure 1.1 in which messages are removed can be viewed as a relational counter system (see Figure 1.4.3). In Figure 1.4.3, id preserves the counter values and each variable that does not occur in the expression labelling a transition implicitly preserves its value.

In [CJ98] such systems have been studied and the first result states that the set of Presburger formulae occurring in relational counter systems are closed under composition. For instance, $q \xrightarrow{\mathsf{x}'_1 = \mathsf{x}_1 + 1} q'$ followed by $q' \xrightarrow{\mathsf{x}'_1 > \mathsf{x}_1} q''$ is equivalent to

$$q \xrightarrow{\mathsf{x}'_1 \geq \mathsf{x}_1 + 2} q''$$

Similarly, $q \xrightarrow{\mathsf{x}'_1 = \mathsf{x}'_2 = \mathsf{x}_1} q'$ followed by $q' \xrightarrow{\mathsf{x}'_1 > \mathsf{x}_1 \wedge \mathsf{x}'_2 > \mathsf{x}_2} q''$ is equivalent to

$$q \xrightarrow{\mathsf{x}'_1 > \mathsf{x}_1 \wedge \mathsf{x}'_2 > \mathsf{x}_1} q''$$

which is generalized below.

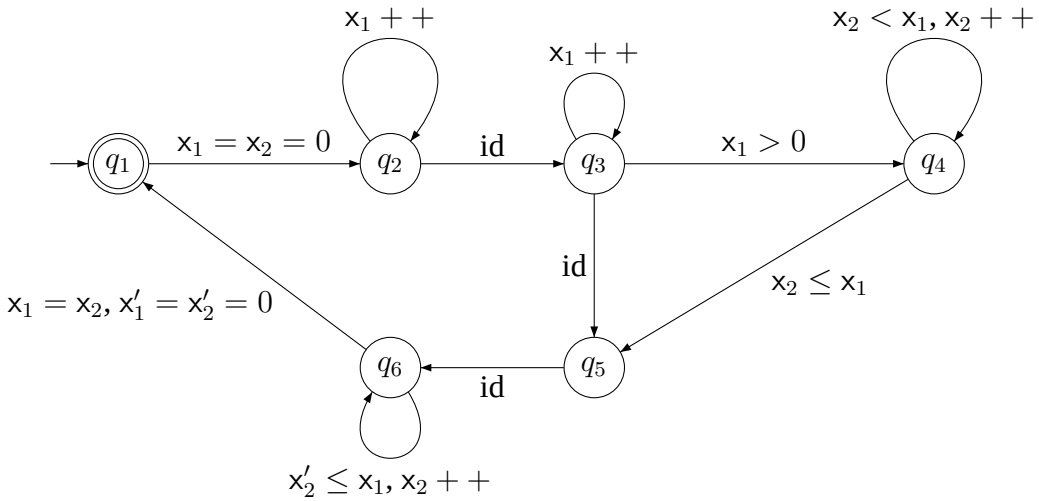


Figure 1.5: Phone controller (bis)

Lemma 1.4.1. [CJ98] Let $\mathcal{S}$ be a relational counter system. Given two transitions $t_1 = q \xrightarrow{\varphi_1} q'$ and $t_2 = q' \xrightarrow{\varphi_2} q''$, there exists a Presburger formula φ of the form described above for defining relational counter systems such that for all $\vec{x}$, $\vec{x'}$ and $\vec{x''}$ in $\mathbb{N}^n$, we have $(q, \vec{x}) \xrightarrow{t_1} (q', \vec{x'}) \xrightarrow{t_2} (q'', \vec{x''})$ iff $(q, \vec{x}) \xrightarrow{t} (q'', \vec{x''})$ with $t = q \xrightarrow{\varphi} q''$.

By way of example, the composition of $x'_1 \geq x_1 + 1 \wedge x'_2 \leq x_2$ and $x'_1 \leq x_2 \wedge x'_1 = x'_2$ leads to $x'_1 \leq x_2 \wedge x'_1 = x'_2$.

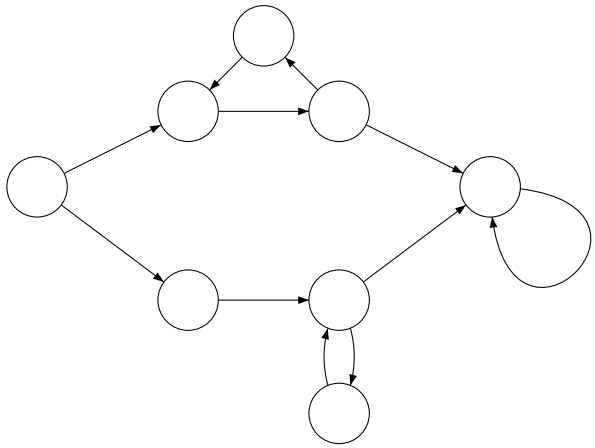
Flat relational counter systems

The key result in [CJ98] states that the transitive closure of transitions occurring in such systems is definable in Presburger arithmetic, see a precise statement in Theorem 1.4.2. The proof is quite difficult; an alternative proof can be found in [BIL09]. For instance, with unique transition $t = q \xrightarrow{x'_1 = x_1 + 1} q$, we have $(q, K) \xrightarrow{*} (q, K')$ iff $K' \geq K$. So, the finite iteration of the transition t corresponds to the transition $q \xrightarrow{x'_1 \geq x_1 + 1} q$. By contrast, with unique transition $t = q \xrightarrow{x'_1 = x_1 + 2} q$, we have $(q, K) \xrightarrow{*} (q, K')$ iff there is $k \in \mathbb{N}$ such that $K' = K + 2k$. Consequently, $(q, K) \xrightarrow{*} (q, K')$ iff $\mathbf{val}_{K,K'} \models \exists\, y\ x'_1 = x_1 + 2 \times y$. Theorem 1.4.2 below generalizes closure under iteration with Presburger arithmetic.

Theorem 1.4.2. [CJ98] Let $\mathcal{S}$ be a relational counter system made of a unique transition $q \xrightarrow{\varphi} q$. One can effectively compute a Presburger formula φ' with $free(\varphi') = \{x_1, \ldots, x_n, x'_1, \ldots, x'_n\}$ such that for all $\vec{x}$, $\vec{x'}$ in $\mathbb{N}^n$, $(q, \vec{x}) \xrightarrow{*} (q, \vec{x'})$ iff $\mathbf{val}_{\vec{x},\vec{x'}} \models \varphi'$ (where $\rightarrow$ is the transition relation from the transition system $T(\mathcal{S})$).

A direct application of the above theorem concerns relational counter systems with restriction on the control graph. A relational counter system is *flat* whenever, in the control graph, every control state belongs to at most one simple cycle. i.e. with no repeated vertex. Moreover, we

require that there is at most one transition between two control states. Here is an example of flat control graph:



The main result in [CJ98] is the following.

Theorem 1.4.3. [CJ98] Let $\mathcal{S}$ be a flat relational counter system and $q, q' \in Q$. One can effectively compute a Presburger formula φ such that for all $\vec{x}$, $\vec{x'}$ in $\mathbb{N}^n$, $(q, \vec{x}) \xrightarrow{*} (q', \vec{x'})$ iff $\mathbf{val}_{\vec{x},\vec{x'}} \models \varphi$ (with $free(\varphi) = \{\mathsf{x}_1, \ldots, \mathsf{x}_n, \mathsf{x}'_1, \ldots, \mathsf{x}'_n\}$).

The proof goes roughly as follows. Lemma 1.4.1 allows to compute the effects of each simple cycle and by flatness the number of simple cycles is bounded by $\mathrm{card}(Q)$. Then, Theorem 1.4.2 allows to compute the effects of passing a finite number of times on each simple cycle. Flatness guarantees that reaching a control state from another control state implies passing through the simple cycles in a regular manner which can be mimicked at the level of formulae.

Consequently,

Corollary 1.4.4. The reachability problem for flat relational counter systems is decidable.

The corollary can be obtained as follows. Consider the instance $\mathcal{S}$, $(q, \vec{y})$ and $(q', \vec{y'})$. We have seen that we can compute the Presburger formula φ that encodes the reachability relation in $\mathcal{S}$. It remains to check satisfiability of the formula below:

$$(\bigwedge_{i=1}^{i=n} (\mathsf{x}_i = \vec{y}(i) \wedge \mathsf{x}'_i = \vec{y'}(i))) \wedge \varphi$$

assuming free variables in φ are $\mathsf{x}_1, \ldots, \mathsf{x}_n, \mathsf{x}'_1, \ldots, \mathsf{x}'_n$. This can be done since the satisfiability problem for Presburger arithmetic is decidable. Other types of counter systems with semilinear reachability sets can be found in [Iba78, HP79, Esp97, FS00, FL02, Ler03, LS05, FS08]. A generalization has been also considered in [BIK10].

Succinct counter automata

In the sequel, we adopt the convention that a counter automaton is a counter system in which the instructions are either zero-tests, increments or decrements, possibly encoded succinctly. A *succinct counter automaton* is a counter system (Q, n, δ) in which the transitions are of the form either $q \xrightarrow{\mathrm{inc}(\vec{b})} q'$ with $\vec{b} \in \mathbb{Z}^n$ or $q \xrightarrow{\mathrm{zero}(\vec{b'})} q'$ with $\vec{b'} \in \{0,1\}^n$ where

- $\mathrm{inc}(\vec{b})$ is a shortcut for $\bigwedge_{i\in[1,n]} \mathsf{x}'_i = \mathsf{x}_i + \vec{b}(i)$,
- $\mathrm{zero}(\vec{b'})$ is a shortcut for $\bigwedge_{i\in[1,n] \text{ s.t. } \vec{b'}(i)=1} \mathsf{x}_i = 0 \wedge \bigwedge_{i\in[1,n]} \mathsf{x}'_i = \mathsf{x}_i$ (as usual, empty conjunction is understood as $\top$).

In succinct counter automaton, each transition either performs zero-tests on a subset of counters or updates counters by adding a vector in $\mathbb{Z}^n$. All the counters are tested or updated simulateously.

It is easy to check that every succinct counter automaton is a relational counter system.

Standard counter automata

A *standard counter automaton* is a counter system (Q, n, δ) in which the transitions are of the form either $q \xrightarrow{\mathrm{inc}(i)} q'$ or $q \xrightarrow{\mathrm{dec}(i)} q'$ or $q \xrightarrow{\mathrm{zero}(i)} q'$ ($i \in [1, n]$) where

- $\mathrm{inc}(i)$ is a shortcut for $(\mathsf{x}'_i = \mathsf{x}_i + 1) \wedge (\bigwedge_{j\neq i} \mathsf{x}'_j = \mathsf{x}_j)$ (also written x_i++),
- $\mathrm{dec}(i)$ is a shortcut for $(\mathsf{x}'_i = \mathsf{x}_i - 1) \wedge (\bigwedge_{j\neq i} \mathsf{x}'_j = \mathsf{x}_j)$ (also written x_i- -),
- $\mathrm{zero}(i)$ is a shortcut for $(\mathsf{x}_i = 0) \wedge (\bigwedge_j \mathsf{x}'_j = \mathsf{x}_j)$ (also written $\mathsf{x}_i = 0?$).

By contrast to succinct counter automata, transitions in standard counter automata can perform a simple operation at once (otherwise, a succession of transitions is needed). Indeed, standard counter automata and succinct counter automata are very similar but when it comes to complexity issues, exponential blow-up may occur when passing from one model to another. In the sequel, unless otherwise stated, by a counter automaton we mean a standard one.

It is easy to check that Minsky machines (with two counters) form a subclass of standard counter automata.

Vector addition systems with states

A *vector addition system with states* [KM69] (VASS for short) is a succinct counter automata without zero-tests, i.e. all the transitions are of the form $q \xrightarrow{\mathrm{inc}(\vec{b})} q'$ with $\vec{b} \in \mathbb{Z}^n$. In the sequel, a VASS is represented by a tuple $\mathcal{V} = (Q, n, \delta)$ where Q is the finite set of control states and δ is a finite subset of $Q \times \mathbb{Z}^n \times Q$.

Standard counter automata can be naturally viewed as vector addition systems with states augmented with zero-tests by simulating transitions of the form $q \xrightarrow{\vec{b}} q'$ by sequences of increments and decrements.

Figure 1.6 presents an example of VASS. As an exercise, one can show that for all $\vec{x} \in \mathbb{N}^4$, the set $\{\vec{y} \in \mathbb{N}^4 : (q_0, \vec{x}) \xrightarrow{*} (q_0, \vec{y})\}$ is finite.

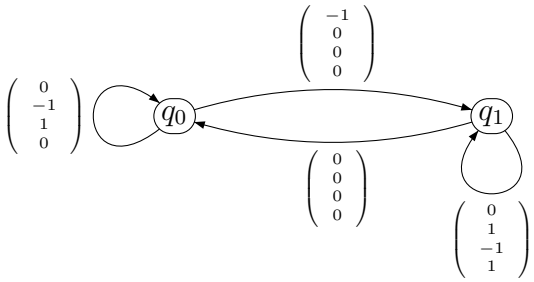


Figure 1.6: A VASS weakly computing multiplication

A *vector addition systems* (VAS for short) is defined as a VASS with a unique control state. In the sequel, a VAS $\mathcal{T}$ is represented by a finite subset of $\mathbb{Z}^n$ corresponding to its set of transitions. VASS and VAS are be viewed as equivalent models, see e.g. [Reu90]. Moreover, for many problems such as covering or boundedness, the problems on VASS and Petri nets are equivalent (see Chapter 3).

Theorem 1.4.5. [May84, Kos82, Reu90, Lam92] The reachability problem for VASS is decidable.

This famous result has been the subject of the book [Reu90] since the proof requires many steps involving expertise in graph theory, logic, theory of well-quasi orderings etc. Nevertheless, the exact complexity of the reachability problem is open: we know it is EXPSPACE-hard [Lip76, CLM76, Esp98] and no primitive recursive upper bound exists. By contrast, the covering problem and boundedness problems seem easier.

Theorem 1.4.6. [Lip76, Rac78] The covering and boundedness problems for VASS are EXPSPACE-complete.

Decidability is established in [KM69] but with a worst-case non primitive recursive bound (see Section 3.3). The EXPSPACE lower bound is due to Lipton and the upper bound to Rackoff (see Section 3.4). In order to be precise, one should explain how vectors in $\mathbb{Z}^n$ are encoded. The upper bound holds true with a binary representation of integers whereas the lower bound holds true already with the values -1, 0 and 1. Consequently, the problem is EXPSPACE-hard even with an unary encoding. In general, the less the encoding is concise, the more difficult hardness results are possible. We shall present the proof for the upper bound in Chapter 3. Observe also that the covering problem can also express the thread-state reachability problem for replicated finite-state programs, see e.g. [KKW10] as well as decision problems for the parameterized verification of ad-hoc networks [DSZ10]. Similarly, the boundedness problem for asynchronous programs has been considered in [GaAR09].

The operation of resetting a counter consists in providing the value zero to the counter. For instance $\mathrm{reset}(i)$ can be defined as the formula $(\mathsf{x}'_i = 0) \wedge (\bigwedge_{j \neq i} \mathsf{x}'_j = \mathsf{x}_j)$ A *reset VASS* is defined as a VASS except that we allow transitions labelled by $\mathrm{reset}(i)$. It is worth noting that the boundedness and the reachability problems for reset VASS become undecidable, see e.g. [DFS98]. By contrast, the covering problem for reset VASS is decidable by using the theory of well-structured transition systems, see e.g. [FS01].

1.5 Exercises

Exercise 1.5.1. Let φ be a Presburger formula with more than one free variable. Define a Presburger formula ψ such that ψ is satisfiable iff $\mathrm{REL}(\varphi)$ is finite.

Exercise 1.5.2. Let $X \subseteq \mathbb{N}^2$ be a semilinear set. Show that $\{k \in \mathbb{N} : (k, k') \in X\}$ (projection) is a semilinear subset of $\mathbb{N}$.

Exercise 1.5.3. Define a Presburger formula φ such that $\mathrm{REL}(\varphi) = \{(n_1, n_2) \in \mathbb{N} \times \mathbb{N} : n_1 \times n_2 \text{ is odd}\}$.

Exercise 1.5.4. Show that any arithmetic progression (viewed as a set of natural numbers) can be defined in Presburger arithmetic.

Exercise 1.5.5. Show that a set X of natural numbers is semilinear iff there are $N, M \in \mathbb{N}$ such that for every $n \geq N$, $n \in X$ iff $n + M \in X$ (X is *ultimately periodic*). Conclude that the sets $\{2^i : i \in \mathbb{N}\}$ and $\{i^2 : i \in \mathbb{N}\}$ are not semilinear.

Exercise 1.5.6. Show that semilinear sets are Presburger definable.

Exercise 1.5.7. Let $X, Y \subseteq \mathbb{N}^n$. We define $X + Y$ as the set $\{\vec{x} + \vec{y} : \vec{x} \in X,\ \vec{y} \in Y\}$. Show that if X and Y are semilinear, then $X + Y$ is also semilinear.

Exercise 1.5.8. Dickson's Lemma [Dic13] states that for any ω-sequence $\vec{x_0}, \vec{x_1}, \ldots$ of tuples in $\mathbb{N}^n$, there are $i < j$ such that $\vec{x_i} \preceq \vec{x_j}$. Show Dickson's Lemma.

Exercise 1.5.9. Let $\mathcal{S} = (Q, n, \delta)$ be a VASS and $(q_0, \vec{x_0})$ be a configuration. Using Dickson's Lemma, show the equivalence between the two statements below:

- ⋆ There is an infinite run with initial configuration $(q_0, \vec{x_0})$.
- ⋆ There exist a finite run $(q_0, \vec{x_0}), \ldots, (q_k, \vec{x_k})$ and $k' < k$ such that $q_{k'} = q_k$ and $\vec{x_{k'}} \preceq \vec{x_k}$.

Does the equivalence hold true for VASS with resets? for standard counter automaton?

Exercise 1.5.10. Let us consider the VASS $\mathcal{S}$ presented in Figure 1.6. Determine the set of initial configurations such that $\mathcal{S}$ is terminating from them.

Exercise 1.5.11. Consider the counter system obtained from Figure 1.1 by deleting the communication labels (of the form either $a!$ or $b?$). Show that the set $\{\vec{x} \in \mathbb{N}^2 : (q_1, \vec{0}) \xrightarrow{*} (q_i, \vec{x}), i \in [1, 6]\}$ is semilinear.

Exercise 1.5.12. Check the statement in Lemma 1.4.1 with

$$\varphi_1 = \mathsf{x}_1' \geq \mathsf{x}_1 + 1 \wedge \mathsf{x}_2' \leq \mathsf{x}_2 \quad \varphi_2 = \mathsf{x}_1' \leq \mathsf{x}_2 \wedge \mathsf{x}_1' = \mathsf{x}_2' \quad \varphi = \mathsf{x}_1' \leq \mathsf{x}_2 \wedge \mathsf{x}_1' = \mathsf{x}_2'$$

Exercise 1.5.13. Show the following equality for the VASS defined in Figure 1.6:

$$\{\begin{pmatrix} a \\ b \\ d \end{pmatrix} \in \mathbb{N}^3 : d \leq a \times b\} =$$

$$\{\begin{pmatrix} a \\ b \\ d \end{pmatrix} \in \mathbb{N}^3 : \exists \begin{pmatrix} a' \\ b' \\ c' \end{pmatrix} \in \mathbb{N}^3,\ \mathrm{run}\ (q_0, \begin{pmatrix} a \\ b \\ 0 \\ 0 \end{pmatrix}) \xrightarrow{*} (q_0, \begin{pmatrix} a' \\ b' \\ c' \\ d \end{pmatrix})\}$$

Chapter 2

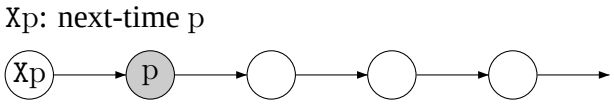
Linear-Time Temporal Logics

This chapter is mainly dedicated to present a linear-time temporal logic $\mathrm{LTL}^{\mathrm{CS}}(\mathrm{PrA})$ whose models are infinite runs from counter systems. First, we recall the definitions for standard linear-time temporal logic LTL as well as its automata-based approach with Büchi automata. Then, we define the very expressive logic $\mathrm{LTL}^{\mathrm{CS}}(\mathrm{PrA})$ and its two fragments $\mathrm{CLTL}(\mathrm{PrA})$ (Presburger LTL) and LTL with registers ($\mathrm{LTL}^{\downarrow}$) obtained by restricting the first-order quantification over counter values. The chapter concludes by providing the decidability status for several fragments for $\mathrm{CLTL}(\mathrm{PrA})$ and $\mathrm{LTL}^{\downarrow}$. In the subsequent chapters, we shall refer to these logics to state the decidability status of satisfiability and model-checking problems.

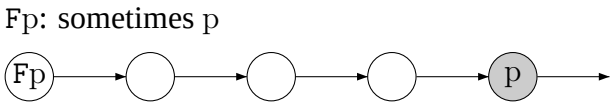
2.1 Temporal Modalities on Computations

Temporal logics contain modalities with a temporal interpretation. A modality is usually defined as a syntactic object (term) that modifies the relationships between a predicate and a subject. For example, in the sentence "Tomorrow, it will rain", the term "Tomorrow" is a temporal modality. Temporal logics make use of different types of modalities and we recall below some of them interpreted over runs (a.k.a. executions or ω-sequences). The temporal modalities (also known as temporal operators) allow one to speak about the sequencing of states along an execution, rather than about the states taken individually. The simplest temporal operators are `X` ("neXt"), `F` ("sometimes") and `G` ("always"). Below, we shall freely use the Boolean operators $\neg$ (negation), $\vee$ (disjunction), $\wedge$ (conjunction) and $\Rightarrow$ (material implication).

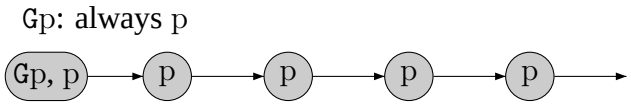
- $\star$ Whereas φ states a property of the current state, $\mathtt{X}\varphi$ states that the next state (`X` for "neXt") satisfies φ. For example, $\varphi \vee \mathtt{X}\varphi$ states that φ is satisfied now or in the next state.



- $\star$ $\mathtt{F}\mathrm{p}$ announces that a future state (`F` for "Future") satisfies φ without specifying which state, and $\mathtt{G}\varphi$ that all the future states satisfy φ. These two operators can be read informally as "φ will hold some day" and "φ will always be".

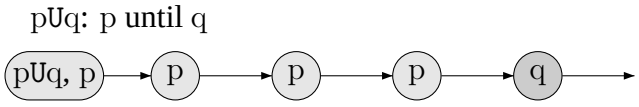


Duality The operator G is the dual of F: whatever the formula φ may be, if φ is always satisfied, then it is not true that $\neg\varphi$ will some day be satisfied, and conversely. Hence $\mathtt{G}\varphi$ and $\neg\mathtt{F}\neg\varphi$ are equivalent.



By way of example, the expression $\mathtt{alert} \Rightarrow \mathtt{F}\ \mathtt{halt}$ means that if we (currently) are in a state of alert, then we will (later) be in a halt state.

$\star$ The temporal operator U (for "Until") is richer and more complicated than the temporal operator F. $\varphi_1\mathtt{U}\varphi_2$ states that φ_1 is true until φ_2 is true. More precisely: φ_2 will be true some day, and φ_1 will hold in the meantime.



The example $\mathtt{G}(\mathtt{alert} \Rightarrow \mathtt{F}\ \mathtt{halt})$ can be refined with the statement that "starting from a state of alert, the alarm remains activated until the halt state is eventually reached":

$$\mathtt{G}(\mathtt{alert} \Rightarrow (\mathtt{alarm}\ \mathtt{U}\ \mathtt{halt})).$$

Sometime operator. The temporal operator F is a special case of U: $\mathtt{F}\varphi$ and $\mathtt{true}\ \mathtt{U}\ \varphi$ are equivalent.

Weak until. There exists also a "weak until", denoted W. The statement $\varphi_1\mathtt{W}\varphi_2$ still expresses "$\varphi_1\mathtt{U}\varphi_2$", but without the inevitable occurrence of φ_2 and if φ_2 never occurs, then φ_1 remains true forever. So, $\varphi_1\mathtt{W}\varphi_2$ is equivalent to $\mathtt{G}\varphi_1 \vee (\varphi_1\mathtt{U}\varphi_2)$.

2.2 Linear-Time Temporal Logic LTL

As far as we know, linear-time temporal logic LTL in the form presented herein has been first considered in [GPSS80] based on the early works [Kam68, Pnu77]. The version of LTL with explicitely the next-time and until operators first appeared in [GPSS80]. Surprisingly, the next-time operator has been introduced in [MP79] in order to define LTL restricted to the next-time and sometime operators (see also a similar language in [Pnu79]). Nowadays, LTL is one of the most used logical formalisms to specify the behaviours of computer systems in view of formal

verification. It has been also the basis for numerous specification languages such as PSL [EF06]. Moreover, it is used as a specification language in tools such as SPIN [Hol97] and SVM [McM93].

Let us provide below a few definitions about LTL. LTL formulae are built from the following abstract grammar:

$$\varphi, \psi ::= \mathrm{p} \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \mathtt{X}\varphi \mid \varphi \mathtt{U} \psi$$

where p ranges over a countably infinite set PROP of propositional variables.

LTL models are intended to be infinite runs from counter systems, i.e. ω-sequences of configurations. In plain LTL, a model ρ is simply a map $\mathbb{N} \to \mathcal{P}(\mathrm{PROP})$. The satisfaction relation $\models$ is defined as follows:

- $\rho, i \models \mathrm{p} \stackrel{\text{def}}{\Leftrightarrow} \mathrm{p} \in \rho(i)$,
- $\rho, i \models \neg\varphi \stackrel{\text{def}}{\Leftrightarrow} \rho, i \not\models \varphi$,
- $\rho, i \models \varphi_1 \wedge \varphi_2 \stackrel{\text{def}}{\Leftrightarrow} \rho, i \models \varphi_1$ and $\rho, i \models \varphi_2$,
- $\rho, i \models \varphi_1 \vee \varphi_2 \stackrel{\text{def}}{\Leftrightarrow} \rho, i \models \varphi_1$ or $\rho, i \models \varphi_2$,
- $\rho, i \models \mathtt{X}\varphi \stackrel{\text{def}}{\Leftrightarrow} \rho, i+1 \models \varphi$,
- $\rho, i \models \varphi_1 \mathtt{U} \varphi_2 \stackrel{\text{def}}{\Leftrightarrow}$ there is $j \geq i$ such that $\rho, j \models \varphi_2$ and $\rho, k \models \varphi_1$ for all $i \leq k < j$.

As usual, we pose $\mathtt{F}\varphi \stackrel{\text{def}}{=} \top \mathtt{U} \varphi$ and $\mathtt{G}\varphi \stackrel{\text{def}}{=} \neg\mathtt{F}\neg\varphi$. Observe that $\psi_1 \mathtt{U} \psi_2$ is equivalent to $\psi_2 \vee (\psi_1 \wedge \mathtt{X}\psi_1 \mathtt{U} \psi_2)$.

Given a LTL formula φ, we write Models(φ) to denote the set of sequences ρ in $\mathcal{P}(\mathrm{PROP})^\omega$ such that $\rho, 0 \models \varphi$. We recall below that Models(φ) can be effectively represented by a Büchi automaton $\mathcal{A}_\varphi$ (see basics in Section 2.3), namely $\mathcal{A}_\varphi$ recognizes exactly the sequences in Models(φ).

2.3 A Brief Introduction to Büchi Automata

Automata-based approach

The automata-based approach consists in reducing logical problems into automata-based decision problems in order to take advantage of known results from automata theory. Alternatively, this can be viewed as a means to transform declarative statements (typically formulae) into operational devices (typically automata with sometimes rudimentary computational power). The most standard target problems on automata used in this approach are the following:

- the nonemptiness problem checks whether an automaton admits at least one accepting computation (in symbols $\mathrm{L}(\mathcal{A}) \neq \emptyset$?),
- the universality problem checks whether an automaton accepts everything (of course this needs to be made more precise depending on the objects we are dealing with, words, trees etc.),
- the inclusion problem checks whether the language accepted by the automaton $\mathcal{A}$ is included in the language accepted by the automaton $\mathcal{B}$ (in symbols $\mathrm{L}(\mathcal{A}) \subseteq \mathrm{L}(\mathcal{B})$?).

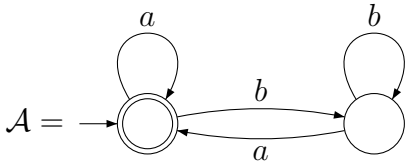


Figure 2.1: $\mathrm{L}(\mathcal{A}) = \{\sigma \in \{a,b\}^\omega \mid |\sigma|_a = \omega\}$

A pioneering work is due to Büchi [Büc62] in which Büchi automata are shown equivalent to formulae in monadic second-order logics (MSO) over $(\mathbb{N}, <)$; models of a formula built over the second-order variables $P_1, \ldots, P_N$ are ω-sequences over the alphabet $\mathcal{P}(\{P_1, \ldots, P_N\})$. In full generality, here are a few desirable properties of the approach.

- ⋆ The reduction should be conceptually simple, apart from being semantically faithful.
- ⋆ The computational complexity of the automata-based target problem should be well-characterized. In that way, one gets a complexity upper bound for solving the source logical problem.
- ⋆ Last but not least, preferrably, the reduction should allow obtaining the optimal complexity for the source logical problem.

In this chapter, we shall present the automata-based approach for solving logical problems involving temporal logics (see also Chapter 5). However, nowadays this approach is quite active and among the trends one can distinguish the development of algorithmic automata theory (for instance to design efficient decision procedures for testing nonemptiness, complementing etc., see e.g. [GS09]) and the appearance of new source problems (see e.g. axiom pinpointing in [BP08]).

Büchi automata in a nutshell

A *Büchi automaton* is defined as a finite-state automaton that accepts ω-words instead of finite words. Formally, a *Büchi automaton* $\mathcal{A}$ is a tuple $\mathcal{A} = (\Sigma, Q, Q_0, \delta, F)$ such that

- ⋆ Σ is a finite *alphabet*,
- ⋆ Q is a finite set of *states*,
- ⋆ $Q_0 \subseteq Q$ is the set of *initial* states,
- ⋆ the *transition relation* δ is a subset of $Q \times \Sigma \times Q$,
- ⋆ $F \subseteq Q$ is a set of *final* states.

Given $q \in Q$ and $a \in \Sigma$, we also write $\delta(q,a)$ to denote the set of states q' such that $(q,a,q') \in \delta$.

A *run* ρ of $\mathcal{A}$ is a sequence $q_0 \xrightarrow{a_0} q_1 \xrightarrow{a_1} q_2 \ldots$ such that $q_0 \in Q_0$ and for every $i \geq 0$, $(q_i, a_i, q_{i+1}) \in \delta$ (also written $q_i \xrightarrow{a_i} q_{i+1}$). The run ρ is *successful* if some state of F is repeated infinitely often in ρ: $\inf(\rho) \cap F \neq \emptyset$ where we let $\inf(\rho) = \{q \in Q : \forall\, i,\ \exists\, j > i,\ q = q_j\}$. The *label* of ρ is the word $\sigma = a_0 a_1 \cdots \in \Sigma^\omega$. The automaton $\mathcal{A}$ *accepts* the language $\mathrm{L}(\mathcal{A})$ made of ω-words $\sigma \in \Sigma^\omega$ such that there exists a successful run of $\mathcal{A}$ on the word σ, i.e., with label σ. For instance, the automaton in Figure 2.1 accepts those words over $\{a,b\}$ having infinitely many a's (the initial states are marked with an incoming arrow and the states in F are doubly circled).

Now, we introduce a standard generalization of the Büchi acceptance condition by considering conjunctions of classical Büchi conditions. A *generalized Büchi automaton* (GBA) is a structure

$$\mathcal{A} = (\Sigma, Q, Q_0, \delta, \{F_1, \ldots, F_k\})$$

such that $F_1, \ldots, F_k \subseteq Q$ and Σ, Q, Q_0 and δ are defined as for Büchi automata. A run is defined as for Büchi automata and a run ρ of $\mathcal{A}$ is *successful* iff for $1 \leq i \leq n$, we have $\inf(\rho) \cap F_i \neq \emptyset$. Lemma 2.3.1 below simply states each GBA $\mathcal{A}$ can be easily translated into a classical BA, preserving the language of accepted ω-words.

Lemma 2.3.1. Let $\mathcal{A} = (\Sigma, Q, Q_0, \delta, \{F_1, \ldots, F_k\})$ be a generalized Büchi automaton. One can compute, in logarithmic space in the size of $\mathcal{A}$, a Büchi automaton $\mathcal{A}^b = (\Sigma, Q^b, Q_0^b, \delta^b, F^b)$ such that $\mathrm{L}(\mathcal{A}^b) = \mathrm{L}(\mathcal{A})$.

Proof: Let $\mathcal{A} = (\Sigma, Q, Q_0, \delta, \{F_1, \ldots, F_k\})$ be a generalized Büchi automaton. The idea of the proof consists in defining $\mathcal{A}^b$ from k copies of $\mathcal{A}$ and to simulate the generalized accepting condition by passing from one copy to another.

1. $Q^b \stackrel{\text{def}}{=} Q \times \{1, \ldots k\}$,
2. $Q_0^b \stackrel{\text{def}}{=} Q_0 \times \{1\}$,
3. $F^b \stackrel{\text{def}}{=} F_1 \times \{1\}$,
4. $\delta^b((q,i),a)$ is defined as the union of the two following sets
   (a) $\{(q',i) : q \xrightarrow{a} q' \in \delta,\ q \notin F_i\}$ (stay in the same copy if no final state in F_i is reached),
   (b) $\{(q', (i \bmod k) + 1) : q \xrightarrow{a} q' \in \delta,\ q \in F_i\}$ (go to the next copy if a final state in F_i is reached).

One can check that $\mathcal{A}$ and $\mathcal{A}^b$ accept the same language. QED

The class of languages accepted by Büchi automata admits various characterizations, for instance it corresponds to the class of ω-regular languages. On the logical side, such languages correspond exactly to the set of models satisfied by formulae from LTL augmented with automata-based temporal operators [Wol83]. There exist alternative characterizations, see e.g. [Var88].

Proposition 2.3.2. The family of ω-regular languages is closed by intersection, union and complementation.

The proof for union is similar to the proof for standard finite-state automata, the proof for intersection uses an idea similar to the proof of Lemma 2.3.1. By contrast, the closure by complementation is much more difficult to show, see e.g. [Büc62, Tho99, Muk09] (see also [FKV04]).

The nonemptiness problem for Büchi automata is defined as follows:

Input: a Büchi automaton $\mathcal{A}$,

Question: is $\mathrm{L}(\mathcal{A}) \neq \emptyset$?

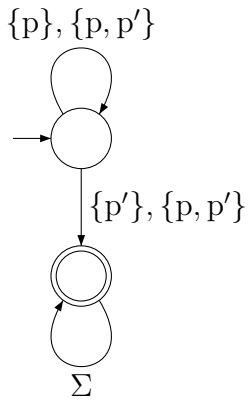


Figure 2.2: An automaton for $\mathrm{p}\mathtt{U}\mathrm{p}'$ models

In order to characterize the computational complexity of the nonemptiness problem, we can use the lemma below.

Lemma 2.3.3. Let $\mathcal{A} = (\Sigma, Q, Q_0, \delta, F)$ be a Büchi automaton. $\mathrm{L}(\mathcal{A}) \neq \emptyset$ iff there is a path in the graph $(Q, \{(q, q') : \exists a \text{ s.t. } q \xrightarrow{a} q' \in \delta\})$ of the form $q_0 \xrightarrow{*} q \xrightarrow{+} q$ with $q_0 \in Q_0$ and $q \in F$.

Proposition 2.3.4. [VW94] The nonemptiness problem for Büchi automata is NLOGSPACE-complete.

By contrast, the universality problem for Büchi automata is PSPACE-complete, see e.g., [SVW87].

2.4 From Formulae to Automata

Here we will show that given an LTL formula φ built over the set of propositional variables $\{\mathrm{p}_1, \ldots, \mathrm{p}_N\}$, it is possible to effectively construct a Büchi automaton $\mathcal{A}_\varphi$ over the alphabet $\{\mathrm{p}_1, \ldots, \mathrm{p}_N\}$ such that $\mathrm{L}(\mathcal{A}_\varphi)$ = Models(φ).

Figure 2.2 presents a Büchi automaton $\mathcal{A}$ such that $\mathrm{L}(\mathcal{A}) =$ Models($\mathrm{p}\mathtt{U}\mathrm{p}'$) with $\Sigma = \{\mathrm{p}, \mathrm{p}'\}$. We wish to define $\mathcal{A}_\varphi$ from φ in a systematic and optimal way. This allows us to obtain optimal complexity bounds and if the construction of automata is optimized, it can also provide efficient algorithms.

Proposition 2.4.1. [VW94] For every LTL formula φ, there is a Büchi automaton $\mathcal{A}_\varphi$ such that

1. $\mathrm{L}(\mathcal{A}_\varphi) =$ Models(φ),
2. $|\mathcal{A}_\varphi|$ is in $2^{\mathcal{O}(|\varphi|)}$ and,
3. $\mathcal{A}_\varphi$ can be effectively computed in polynomial space in $|\varphi|$.

In Proposition 2.4.1, it should be understood that the set of propositional variables PROP is restricted to the atomic formulae occurring in φ. Below, we explain how $\mathcal{A}_\varphi$ is defined from φ without providing the formal proofs.

Definition 2.4.1. Let φ be an LTL formula. The *closure* of φ, denoted by $\mathrm{cl}(\varphi)$ is the smallest set

- $\star$ containing the subformulae of φ,
- $\star$ closed under negation (we identify $\neg\neg\psi$ with ψ),
- $\star$ if $\chi_1 \mathtt{U} \chi_2 \in \mathrm{cl}(\varphi)$, then $\mathtt{X}(\chi_1 \mathtt{U} \chi_2) \in \mathrm{cl}(\varphi)$.

∇

The closure set contains all the formulae we need to consider to check satisfiability and the cardinality of $\mathrm{cl}(\varphi)$ is linear in the size of φ. An *atom* X is a subset of $\mathrm{cl}(\varphi)$ satisfying the conditions below:

1. for all formulae ψ in $\mathrm{cl}(\varphi)$, $\psi \in X$ iff $\neg\psi \notin X$,
2. $\psi_1 \wedge \psi_2 \in X$ iff $\psi_1, \psi_2 \in X$,
3. $\psi_1 \vee \psi_2 \in X$ iff $\psi_1 \in X$ or $\psi_2 \in X$.

An atom is nothing but a maximally consistent subset of $\mathrm{cl}(\varphi)$. A pair of atoms (X, X') is said to be *one-step consistent* iff the conditions below hold true:

- $\star$ if $\psi_1 \mathtt{U} \psi_2 \in X$, then $\psi_2 \in X$ or ($\psi_1 \in X$ and $\psi_1 \mathtt{U} \psi_2 \in X'$),
- $\star$ for $\mathtt{X}\psi \in \mathrm{cl}(\varphi)$, $\mathtt{X}\psi \in X$ iff $\psi \in X'$.

Now, let us define $\mathcal{A}_\varphi$ based on the previous definitions. $\mathcal{A}_\varphi$ is actually a generalized Büchi automaton that can be converted into a standard Büchi automaton. So $\mathcal{A}_\varphi = (\Sigma, Q, Q_0, \delta, F_1, \ldots, F_\alpha)$ with:

- $\star$ $\Sigma = \mathcal{P}(\{\mathrm{p}_1, \ldots, \mathrm{p}_N\})$ (set of propositional variables occurring in φ),
- $\star$ Q is the set of atoms (its cardinality is exponential in the size of φ),
- $\star$ Q_0 is the subset of atoms containing φ,
- $\star$ $X \xrightarrow{a} Y \in \delta$ iff $a = \{\mathrm{p}_1, \ldots, \mathrm{p}_N\} \cap X$ and (X, Y) is one-step consistent,
- $\star$ for each until formula $\psi_1 \mathtt{U} \psi_2$, there is exactly one set F_i such that $F_i = \{X \in Q : \text{either } \psi_1 \mathtt{U} \psi_2 \notin X \text{ or } \psi_2 \in X\}$.

Each control state $X \in Q$ is a set of formulae that are intended to be satisfied at the current position. Either this satisfaction can be checked locally (typically for Boolean formulae using the fact that X is an atom) or the transition relation of $\mathcal{A}_\varphi$ allows us to propagate the constraints. Accepting conditions $F_1, \ldots, F_\alpha$ (indexed by until formulae occurring in φ) guarantee that the search for witnesses is not delayed forever. In particular, they forbid postponing forever the satisfaction of ψ_2 when $\psi_1 \mathtt{U} \psi_2$ has to be satisfied.

Proposition 2.4.2. $\mathrm{L}(\mathcal{A}_\varphi) = \text{Models}(\varphi)$.

The equality is obtained by establishing the two following properties.

- ⋆ Given a model $\rho : \mathbb{N} \to \mathcal{P}(\{\mathrm{p}_1, \ldots, \mathrm{p}_N\}) \in \text{Models}(\varphi)$, i.e. $\rho, 0 \models \varphi$, there is a unique accepting run $X_0 \xrightarrow{\rho(0)} X_1 \xrightarrow{\rho(1)} X_2 \xrightarrow{\rho(2)} \cdots$ in $\mathcal{A}_\varphi$ such that for all $i \geq 0$ and $\psi \in \mathrm{cl}(\varphi)$, we have $\psi \in X_i$ iff $\rho, i \models \psi$.
- ⋆ Conversely, for each accepting run $X_0 \xrightarrow{a_0} X_1 \xrightarrow{a_1} X_2 \xrightarrow{a_2} \cdots$ in $\mathcal{A}_\varphi$, the model ρ defined by $\rho(i) = a_i$ for $i \geq 0$ satisfies that for $i \geq 0$ and $\psi \in \mathrm{cl}(\varphi)$, we have $\psi \in X_i$ iff $\rho, i \models \psi$.

In the sequel, we write $\mathcal{A}_\varphi$ to denote the Büchi automaton recognizing Models(φ).

2.5 Full Presburger LTL for Counter Systems

Linear-time temporal logic LTL equipped with "next-time" operator X, "until" operator U and their past-time counterparts is known to be equivalent to first-order theory of order [Kam68]. Satisfiability and model-checking problems for LTL (even with past-time operators) are known to be PSPACE-complete [SC85]. In spite of these nice features, it is worth recalling that a propositional variable p only represents a property of the current configuration of the system. For instance, p may hold true whenever the value of the variable x is greater than the value of the variable y after running the current instruction. A more satisfying solution is to include in the logical language the possibility to express directly constraints between variables of the program, whence giving up the standard abstraction made with propositional variables. When the variables are typed, they may be interpreted in some specific domain like integers, real numbers, strings and so on; reasoning in such theories can be performed thanks to satisfiability modulo theories proof techniques, see e.g., [BSST08] and [GNRZ07] in which SMT solvers are used for model-checking infinite-state systems. Hence, a proposition like "x is greater than the next value of y" can be encoded in such extended temporal logics by $\mathsf{x} > \mathbf{X}\mathsf{y}$ but this time the models are sequences of configurations. This means that each position comes with a control state and a valuation for variables. Hence, the basic idea behind the design of the logic $\mathrm{LTL}^{\mathrm{CS}}(\mathrm{PrA})$ is to refine the language of atomic formulae and to allow the possibility to compare counter values at successive positions of the run of the counter systems. Similar motivations can be found in the introduction of concrete domains in description logics, that are logic-based formalisms for knowledge representation [BH91, Lut03, Lut04].

We define below a version of linear-time temporal logic LTL dedicated to counter systems in which the atomic formulae are Presburger formulae about counter values, the temporal operators are those of LTL and first-order quantification over natural numbers is allowed, although we shall use it in a restricted way. The main advantage of defining a so general language is that it is then easy to compare the different languages in a uniform framework. Similarly, in [MP95], a mixture of first-order logic and LTL is shown sufficient to precisely state verification problems for the class of reactive systems.

We introduce a countable set of *integer variables*, say $\mathrm{VAR}^{\mathrm{p}} = \{\mathsf{y}_1, \mathsf{y}_2, \ldots\}$, for quantification over natural numbers. Elements of $\mathrm{VAR}^{\mathrm{p}}$ are distinct from the *counter variables* in $\mathrm{VAR} = \{\mathsf{x}_1, \mathsf{x}_2, \ldots\}$ that are free variables, only interpreted by the counter values on configurations. We

also consider a countably infinite set $\mathtt{Q} = \{q_1, q_2, \ldots\}$ of control state symbols. As usual, a formula contains only a finite number of such symbols but *a priori*, we do not bound the number of control states. The $\mathrm{LTL}^{\mathrm{CS}}(\mathrm{PrA})$ formulae are defined as follows:

$$\varphi ::= \psi \mid q \mid \varphi \wedge \varphi \mid \neg\varphi \mid \mathtt{X}\varphi \mid \varphi \mathtt{U} \varphi \mid \exists\, \mathsf{y}\ \varphi$$

where ψ is a Presburger formula with free variables included in $\mathrm{VAR}^{\mathrm{p}} \cup \mathrm{VAR}$ and $q \in \mathtt{Q}$. The symbols $\mathtt{X}$ and $\mathtt{U}$ are respectively the classical operators next-time and until from LTL.

The models of $\mathrm{LTL}^{\mathrm{CS}}(\mathrm{PrA})$ formulae are infinite runs from counter systems whose set of control states is included in the countable set $\mathtt{Q}$, i.e. they are ω-sequences of configurations. A *model ρ of dimension n for* $\mathrm{LTL}^{\mathrm{CS}}(\mathrm{PrA})$ is an element of $(Q \times \mathbb{N}^n)^\omega$ for some finite subset $Q \subseteq \mathtt{Q}$. An *environment* $\mathcal{E}$ is a partial map $\mathrm{VAR}^{\mathrm{p}} \to \mathbb{N}$. The *empty environment* is denoted by $\emptyset$. The satisfiability relation $\models$ is defined as follows between a model ρ of dimension n, a position $i \geq 0$, an environment $\mathcal{E}$ and a formula in which the free variables are among $\mathrm{VAR}^{\mathrm{p}} \cup \{\mathsf{x}_1, \ldots, \mathsf{x}_n\}$.

The satisfaction relation $\models_{\mathcal{E}}$ is defined on runs ρ of the form

$$\rho = (q_0, \vec{x_0}), \ldots, (q_k, \vec{x_k}), \ldots$$

- $\rho, i \models_{\mathcal{E}} q \overset{\text{def}}{\Leftrightarrow} q = q_i$,
- When ψ is a Presburger formula with free variables included in $\mathrm{VAR}^{\mathrm{p}} \cup \{\mathsf{x}_1, \ldots, \mathsf{x}_n\}$, we have $\rho, i \models_{\mathcal{E}} \psi \overset{\text{def}}{\Leftrightarrow} \mathbf{val}_i \models \psi$ in Presburger arithmetic where $\mathbf{val}_i$ is a conservative extension of $\mathcal{E}$ such that for $j \in [1, n]$, $\mathbf{val}_i(\mathsf{x}_j) = \vec{x_i}(j)$,
- $\rho, i \models_{\mathcal{E}} \neg\varphi \overset{\text{def}}{\Leftrightarrow} \rho, i \not\models_{\mathcal{E}} \varphi$,
- $\rho, i \models_{\mathcal{E}} \varphi_1 \wedge \varphi_2 \overset{\text{def}}{\Leftrightarrow} \rho, i \models_{\mathcal{E}} \varphi_1$ and $\rho, i \models_{\mathcal{E}} \varphi_2$,
- $\rho, i \models_{\mathcal{E}} \mathtt{X}\varphi \overset{\text{def}}{\Leftrightarrow} \rho, i+1 \models_{\mathcal{E}} \varphi$,
- $\rho, i \models_{\mathcal{E}} \varphi_1 \mathtt{U} \varphi_2 \overset{\text{def}}{\Leftrightarrow}$ there is $j \geq i$ such that $\rho, j \models_{\mathcal{E}} \varphi_2$ and $\rho, k \models_{\mathcal{E}} \varphi_1$ for all $i \leq k < j$.
- $\rho, i \models_{\mathcal{E}} \exists\, \mathsf{y}\ \varphi$ iff there is a natural number $m \in \mathbb{N}$ such that $\rho, i \models_{\mathcal{E}[\mathsf{y} \mapsto m]} \varphi$.

As usual, $\mathtt{F}\varphi \overset{\text{def}}{=} \top \mathtt{U} \varphi$ and $\mathtt{G}\varphi \overset{\text{def}}{=} \neg\mathtt{F}\neg\varphi$. A *semi-closed formula* is an $\mathrm{LTL}^{\mathrm{CS}}(\mathrm{PrA})$ formula such that no integer variable from $\mathrm{VAR}^{\mathrm{p}}$ is free. By construction, the counter variables x_1, …, x_n are always free and are interpreted as the current counter values. In the decision problems defined below, we shall only consider semi-closed formulae and therefore there is no need to specify an environment in the statements.

For instance, one can express that the first counter strictly increases at every step:

$$\mathtt{G}\, \exists\, \mathsf{y}\ (\mathsf{y} = \mathsf{x}_1 \wedge \mathtt{X}(\mathsf{x}_1 > \mathsf{y}))$$

Similarly, the first counter takes a finite number of values along the run can be expressed by $\exists\, \mathsf{y}\ \mathtt{G}(\mathsf{x}_1 \leq \mathsf{y})$.

SATISFIABILITY PROBLEM FOR $\mathrm{LTL}^{\mathrm{CS}}(\mathrm{PrA})$

Input: An $\mathrm{LTL}^{\mathrm{CS}}(\mathrm{PrA})$ semi-closed formula φ with free counter variables x_1, …, x_n.

Question: Is there a model $\rho \in (\mathtt{Q} \times \mathbb{N}^n)$ of dimension n such that $\rho, 0 \models_{\emptyset} \varphi$?

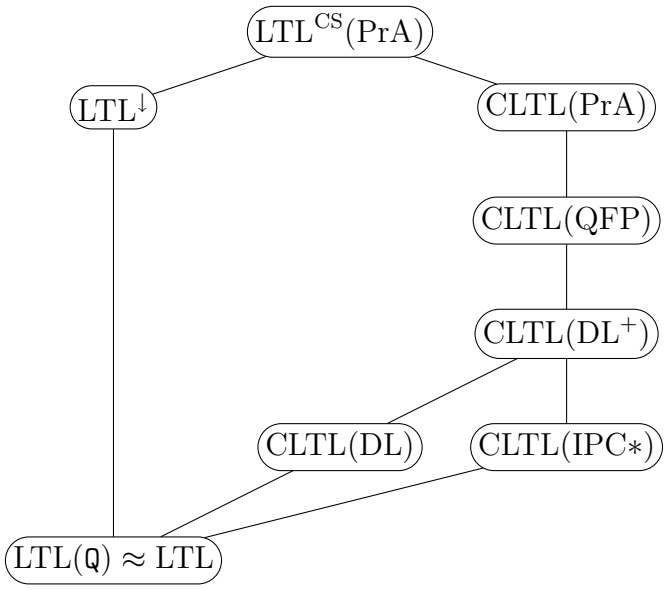


Figure 2.3: Fragments of $\mathrm{LTL}^{\mathrm{CS}}(\mathrm{PrA})$

Observe that for satisfiability checking, it is not necessary that the model is derived from a counter system.

EXISTENTIAL MODEL-CHECKING PROBLEM FOR $\mathrm{LTL}^{\mathrm{CS}}(\mathrm{PrA})$

Input: A counter system $\mathcal{S} = (Q, n, \delta)$, an initial configuration $(q_0, \vec{x}_0)$ and an $\mathrm{LTL}^{\mathrm{CS}}(\mathrm{PrA})$ semi-closed formula φ with free variables among $\{\mathsf{x}_1, \ldots, \mathsf{x}_n\}$.

Question: Is there an infinite run ρ starting at $(q_0, \vec{x}_0)$ such that $\rho, 0 \models_\emptyset \varphi$?

Similarly, one can define the universal model-checking problem for $\mathrm{LTL}^{\mathrm{CS}}(\mathrm{PrA})$.

UNIVERSAL MODEL-CHECKING PROBLEM FOR $\mathrm{LTL}^{\mathrm{CS}}(\mathrm{PrA})$

Input: $\mathcal{S} = (Q, n, \delta)$, $(q_0, \vec{x}_0)$ and φ as above.

Question: It is true that for all the infinite runs ρ starting at $(q_0, \vec{x}_0)$, we have $\rho, 0 \models_\emptyset \varphi$?

Temporal logics with Presburger constraints has been developped, for instance, in [Čer94, BEH95, BGP97, CC00, BDR03, LMP10]. Some of them have quite expressive decidable fragments. Undecidability of the existential model-checking problem for $\mathrm{LTL}^{\mathrm{CS}}(\mathrm{PrA})$ can be shown using the undecidability of the halting problem for Minsky machines. Still, using SMT solvers can be done for checking bounded reachability problems, see e.g., [BFM+10]. A linear-time temporal logic with first-order variables can be also found in [RGL01] for log auditing.

In the rest of this section, we shall present fragments of $\mathrm{LTL}^{\mathrm{CS}}(\mathrm{PrA})$ obtained by restricting first-order quantification over natural numbers. Figure 2.5 illustrates the syntactic fragments based on fragments of Presburger arithmetic defined in Section 1.3.4. Moreover, we write $\mathrm{LTL}(\mathtt{Q})$ to denote the variant of LTL in which the atomic formulae are control states; this is obviously a fragment of $\mathrm{LTL}^{\mathrm{CS}}(\mathrm{PrA})$.

First, let us observe that if we restrict ourselves to formulae in which temporal operators are not in the scope of first-order quantification, then we get a fragment of $\mathrm{LTL}^{\mathrm{CS}}(\mathrm{PrA})$ that is

very similar to plain LTL. Indeed, atomic formulae are arithmetical constraints between counter values and they can be understood as high-level propositional variables; whence the automata-based approach for LTL can be easily adapted to this fragment. In that fragment, the arithmetical constraints are only local and in the construction of Büchi automata, the existence of transitions between states depends on the satisfiability status of Presburger formulae. Below, we provide restrictions in which the temporal operators may occur in the scope of first-order quantification. Observe that variants first-order temporal logics have been introduced in [DSV04] to verify data-driven web applications. The interplay between temporal operators and first-order quantifiers is restricted, which guarantees better computational properties.

Comparing successive counter values. Given a Presburger formula $\psi(\mathsf{z}_1, \ldots, \mathsf{z}_k)$, we shall write $\psi(\mathbf{X}^{i_1}\mathsf{x}_{j_1}, \ldots, \mathbf{X}^{i_k}\mathsf{x}_{j_k})$ to denote the formula below

$$(\exists\, \mathsf{y}_1, \ldots, \mathsf{y}_k\ \mathtt{X}^{i_1}(\mathsf{y}_1 = \mathsf{x}_{j_1}) \wedge \cdots \wedge \mathtt{X}^{i_k}(\mathsf{y}_k = \mathsf{x}_{j_k}) \wedge \psi(\mathsf{y}_1, \ldots, \mathsf{y}_k),$$

where $\mathsf{y}_1, \ldots, \mathsf{y}_k$ are new variables distinct from the free variables that are present in $\psi(\mathsf{z}_1, \ldots, \mathsf{z}_k)$. It is easy to see that $\psi(\mathbf{X}^{i_1}\mathsf{x}_{j_1}, \ldots, \mathbf{X}^{i_k}\mathsf{x}_{j_k})$ is interpreted as the formula $\psi(\mathsf{z}_1, \ldots, \mathsf{z}_k)$ in which each variable z_a takes the value of x_{j_a} at the i_ath next configuration. For instance, $\mathsf{x}_1 = \mathbf{X}\mathsf{x}_2$ specifies that the next value of x_2 is equal to the current value of x_1. Similarly, $\mathtt{G}(\mathsf{x}_1 = \mathbf{X}\mathsf{x}_1)$ states that counter 1 has a constant value along the model. In Section 2.5.1, we present a simple fragment of $\mathrm{LTL}^{\mathrm{CS}}(\mathrm{PrA})$ by allowing first-order quantification only for formulae of the form $\psi(\mathbf{X}^{i_1}\mathsf{x}_{j_1}, \ldots, \mathbf{X}^{i_k}\mathsf{x}_{j_k})$ and Presburger formulae (at the atomic level) are quantifier-free too. It is worth observing that we use '$\mathtt{X}$' as the next-time temporal operator whereas '$\mathbf{X}\mathsf{x}$' refers to the value of x at the next position. Hence $\mathbf{X}$ and $\mathtt{X}$ are of different nature but both refer to the next position (no confusion is possible).

Freeze operator. In order to verify properties on counter systems, we want also to be able to compare counter values. For that, we will define the so-called 'freeze operator'. We shall consider formulae of the form $\downarrow_r^j \varphi$ interpreted as $\exists\, \mathsf{y}_r\ (\mathsf{y}_r = \mathsf{x}_j \wedge \varphi)$ that store counter values. Symmetrically, there are counterpart formulae of the form $\uparrow_r^j$ interpreted as $\mathsf{y}_r = \mathsf{x}_j$ that perform equality tests. Intuitively, the modality $\downarrow_r^j$ is used to store the value of the counter j into the register r; the atomic formula $\uparrow_r^j$ holds true if the value stored in the register r is equal to the current value of the counter j. For instance, the formula $\mathtt{G}(\downarrow_1^1 \mathtt{XG}\neg \uparrow_1^1)$ states that the first counter has distinct values at distinct positions.

2.5.1 Presburger LTL $\mathrm{CLTL}(\mathrm{PrA})$

As mentioned earlier, we define below the logic $\mathrm{CLTL}(\mathrm{PrA})$ as a fragment of $\mathrm{LTL}^{\mathrm{CS}}(\mathrm{PrA})$ with the following restrictions.

1. The Presburger formulae at the atomic level of temporal formulae are quantifier-free.
2. First-order quantification at the level of temporal formulae is restricted to macro formulae of the form below:
$$\psi(\mathbf{X}^{i_1}\mathsf{x}_{j_1}, \ldots, \mathbf{X}^{i_k}\mathsf{x}_{j_k})$$

Consequently, there are no more quantification over integer variables from $\mathrm{VAR}^{\mathrm{p}}$ and no variable in $\mathrm{VAR}^{\mathrm{p}}$ occurs in $\mathrm{CLTL}(\mathrm{PrA})$ formulae.

The logic $\mathrm{CLTL}(\mathrm{PrA})$ is defined as an extension of $\mathrm{LTL}(\mathtt{Q})$ where the atomic formulae are constraints from Presburger arithmetic built over expressions of the form $\mathbf{X}^i\mathsf{x}$ where $\mathsf{x} \in \mathrm{VAR}$ is a variable and $\mathbf{X}^i$ is understood as a sequence of i consecutive symbols $\mathbf{X}$. The expression $\mathbf{X}^i\mathsf{x}$ is interpreted as the value of x at the ith next state. The $\mathrm{CLTL}(\mathrm{PrA})$ formulae are defined as follows (first, we need to define *temporal terms*)

$$t ::= 0 \mid 1 \mid \mathbf{X}^i\mathsf{x} \mid t + t$$

$$\varphi ::= t \equiv_k t \mid t < t \mid q \mid \varphi \wedge \varphi \mid \neg\varphi \mid \mathtt{X}\varphi \mid \varphi\mathtt{U}\varphi.$$

with $\mathsf{x} \in \mathrm{VAR}$, $k > 1$ and $q \in \mathtt{Q}$. For instance, a formula of the form $\mathtt{G}(\mathsf{x}_1 < \mathbf{X}\mathsf{x}_1)$ states that counter 1 is strictly increasing, that is a satisfiable formula. By constrast, the formula $\mathtt{G}(\mathsf{x}_1 > \mathbf{X}\mathsf{x}_1)$ is not satisfiable. Atomic formulae of $\mathrm{CLTL}(\mathrm{PrA})$ can be also written $\psi(\mathbf{X}^{l_1}\mathsf{x}_1, \ldots, \mathbf{X}^{l_n}\mathsf{x}_n)$ where ψ is a Presburger formula and its variables are substituted by expressions of the form $\mathbf{X}^i\mathsf{x}$. A one-step constraint is an atomic formula of the form either $t_1 \equiv_k t_2$ or $t_1 < t_2$ such that each variable is prefixed by at most one symbol $\mathbf{X}$. Given a $\mathrm{CLTL}(\mathrm{PrA})$ formula φ, we define its $\mathbf{X}$-length $|\varphi|_{\mathbf{X}}$ as the maximal number i such that an expression of the form $\mathbf{X}^i\mathsf{x}$ occurs in φ. Intuitively, the $\mathbf{X}$-length defines the size of a frame of consecutive states that can be compared. The models of $\mathrm{CLTL}(\mathrm{PrA})$ are pairs of sequences $\sigma = (\sigma_1, \sigma_2)$ such that $\sigma_1 : \mathbb{N} \to (\mathrm{VAR} \to \mathbb{N})$, $\sigma_2 : \mathbb{N} \to Q$ for a finite subset $Q \subseteq \mathtt{Q}$. The satisfaction relation is defined as for LTL except at the atomic level:

- $\star$ $\sigma, i \models q$ iff $\sigma_2(i) = q$,
- $\star$ $\sigma, i \models \psi(\mathbf{X}^{l_1}\mathsf{x}_1, \ldots, \mathbf{X}^{l_n}\mathsf{x}_n)$ iff $(\sigma_1(i + l_1)(\mathsf{x}_1), \ldots, \sigma_1(i + l_n)(\mathsf{x}_n)) \in \mathrm{REL}(\psi)$.
- $\star$ $\sigma, i \models \mathtt{X}\varphi$ iff $\sigma, i + 1 \models \varphi$,
- $\star$ $\sigma, i \models \varphi\mathtt{U}\varphi'$ iff there is $j \geq i$ such that $\sigma, j \models \varphi'$ and for every $i \leq l < j$, we have $\sigma, l \models \varphi$.

As usual, a formula $\varphi \in \mathrm{CLTL}(\mathrm{PrA})$ is satisfiable whenever there exists a model σ such that $\sigma, 0 \models \varphi$. We write $\mathrm{CLTL}^l_n(\mathrm{PrA})$ to denote the restriction of $\mathrm{CLTL}(\mathrm{PrA})$ to formulae with at most n variables and $\mathbf{X}$-length less or equal to l (below the value ω is used for some syntactic resource when there is no restriction). Moreover, observe that infinite runs from counter systems can be viewed as $\mathrm{CLTL}(\mathrm{PrA})$ models. Besides, control states as atomic formulae can be easily removed from $\mathrm{CLTL}(\mathrm{PrA})$, as far as satisfiability problems are concerned; indeed they can be encoded by atomic Presburger formulae, for instance of the form $\mathsf{z} = \mathsf{z}'$ (where z and z' are uniquely dedicated to encode one control). $\mathsf{z} = 0$ would also be fine.

Lemma 2.5.1. There is a logspace reduction from the satisfiability problem for $\mathrm{CLTL}(\mathrm{PrA})$ to the satisfiability problem for $\mathrm{CLTL}_\omega(\mathrm{PrA})$ restricted to formulae of $\mathbf{X}$-length at most 1 ($\mathrm{CLTL}^1_\omega(\mathrm{PrA})$).

The proof of Lemma 2.5.1 is done by renaming terms and requires an unbounded amount of variables in $\mathrm{CLTL}^1_\omega(\mathrm{PrA})$. For instance, the expressions $\mathsf{x}_1, \ldots, \mathbf{X}^3\mathsf{x}_1$ are encoded by the formula

$$\mathtt{G}(\mathsf{x}'' = \mathbf{X}\mathsf{x}' \wedge \mathsf{x}' = \mathbf{X}\mathsf{x} \wedge \mathsf{x} = \mathbf{X}\mathsf{x}_1)$$

(assuming that x, x' and x'' are new variables) and each occurrence of $\mathbf{X}\mathsf{x}_1$ [resp. $\mathbf{X}^2\mathsf{x}_1$, $\mathbf{X}^3\mathsf{x}_1$] is replaced by x [resp. x', x'']. For reductions between satisfiability problems, the introduction of new variables is harmless.

We recall below the existential version of $\mathrm{CLTL}(\mathrm{PrA})$ model-checking over counter systems.
EXISTENTIAL MODEL-CHECKING PROBLEM FOR $\mathrm{CLTL}(\mathrm{PrA})$:

Input: a counter system $\mathcal{S}$ of dimension n, a configuration $(q, \vec{x})$ and an $\mathrm{CLTL}(\mathrm{PrA})$ formula φ with n free variables,

Question: Is there an infinite run ρ with initial configuration $(q, \vec{x})$ such that $\rho, 0 \models \varphi$?

The halting problem for Minsky machines can be easily reduced to the satisfiability problem for $\mathrm{CLTL}(\mathrm{PrA})$ or to the existential model-checking problem for $\mathrm{CLTL}(\mathrm{PrA})$, leading to simple undecidability proofs. In the sequel, we show how to restrict the class of counter systems or the logical language in order to regain decidability.

Let us define below several fragments of $\mathrm{CLTL}(\mathrm{PrA})$. Given a fragment L from Presburger arithmetic (see e.g., Section 1.3.4), we write $\mathrm{CLTL}(L)$ to denote the restriction of $\mathrm{CLTL}(\mathrm{PrA})$ to atomic formulae of the form $\psi(\mathbf{X}^{i_1}\mathsf{x}_{j_1}, \ldots, \mathbf{X}^{i_k}\mathsf{x}_{j_k})$ with $\psi \in L$. Similarly, we write $\mathrm{CLTL}^l_n(L)$ ($n \geq 1$, $l \geq 0$) to denote the restriction of $\mathrm{CLTL}(L)$ to formulae such that the variables are among $\{\mathsf{x}_1, \ldots, \mathsf{x}_n\}$ and the $\mathbf{X}$-length is bounded by l. We use ω when there is no restriction, either on the number of variables or on the $\mathbf{X}$-length. For instance, $\mathsf{x}_1 = \mathbf{X}^8\mathsf{x}_2 + 1 \in \mathrm{CLTL}^8_2(\mathrm{DL})$ and $\mathtt{XXX}(5\mathbf{X}\mathsf{x}_1 + 2\mathsf{x}_2 \geq 27) \in \mathrm{CLTL}^1_2(\mathrm{QFP})$. The logic CLTL defined in [CC00] is precisely $\mathrm{CLTL}^1_\omega(\mathrm{DL})$.

Similarly, an *L-counter system* is defined as counter system such that the Presburger formulae labelling the transitions are in L. The table below summarizes the complexity status of three decision problems (existential model-checking problems restricted to DL-counter systems, restricted to standard counter automata and satisfiability problem) restricted to different fragments of $\mathrm{CLTL}(\mathrm{PrA})$. 'U' stands for undecidability

	MC (DL)	SAT	MC (CA)
$\mathrm{CLTL}^1_3(\mathrm{DL})$	U	U	U
$\mathrm{CLTL}^\omega_2(\mathrm{DL})$	U	U	U
$\mathrm{CLTL}^2_1(\mathrm{DL})$	U	U	PSPACE-c
$\mathrm{CLTL}^1_2(\mathrm{DL})$	U	U	U
$\mathrm{CLTL}^1_1(\mathrm{DL}$ or $\mathrm{DL}^+)$	PSPACE-c.	PSPACE-c.	PSPACE-c
$\mathrm{CLTL}^1_1(\mathrm{QFP})$	U	U	PSPACE-c
$\mathrm{CLTL}^\omega_1(\mathrm{QFP})$	U	U	PSPACE-c.

By way of example, let us provide the main simple ideas to show the proposition below.

Proposition 2.5.2. The satisfiability problem for $\mathrm{CLTL}^1_3(\mathrm{DL})$ is undecidable.

Proof: Let us consider a slight (and indeed standard) variant of Minsky machines $\mathcal{S}$ in which the last instruction n is `halt`. The halting problem checks whether the Minsky machine can reach this instruction. Since the Minsky machine is deterministic, either the Minsky machine has a unique infinite run (and never visits the instruction n) or it has a unique finite (and halts at

instruction n). We build an $\mathrm{CLTL}^1_3(\mathrm{DL})$ formula $\varphi = (\bigwedge_{i\in[1,n]} \psi_i) \wedge \psi_{init} \wedge \mathtt{G}\neg q_n$ such that φ is satisfiable iff $\mathcal{S}$ has an infinite run. The formula ψ_{init} is of the form $\mathsf{x}_1 = 0 \wedge \mathsf{x}_2 = 0 \wedge \mathsf{x}_3 = 1$, so the third counter encodes the instruction ordinal (the run starts with two counters equal to zero). If the ith instruction increments the counter j and go to instruction i', then ψ_i is of the form below:

$$\mathtt{G}(\mathsf{x}_3 = i \Rightarrow (\mathbf{X}\mathsf{x}_3 = i') \wedge (\mathsf{x}_j + 1 = \mathbf{X}\mathsf{x}_j) \wedge (\mathsf{x}_{3-j} = \mathbf{X}\mathsf{x}_{3-j})).$$

Decrements are encoded similarly. Moreover, if the ith instruction performs a zero-test on counter j and go to instruction i' otherwise go to instruction i'', then ψ_i is of the form below:

$$\mathtt{G}(\mathsf{x}_3 = i \Rightarrow ((\mathsf{x}_j = 0 \Rightarrow (\mathbf{X}\mathsf{x}_3 = i') \wedge (\mathsf{x}_j = \mathbf{X}\mathsf{x}_j) \wedge (\mathsf{x}_{3-j} = \mathbf{X}\mathsf{x}_{3-j})) \wedge$$

$$(\mathsf{x}_j \neq 0 \Rightarrow (\mathbf{X}\mathsf{x}_3 = i'') \wedge (\mathsf{x}_j - 1 = \mathbf{X}\mathsf{x}_j) \wedge (\mathsf{x}_{3-j} = \mathbf{X}\mathsf{x}_{3-j}))))$$

So, the formulae in $\mathrm{CLTL}^1_3(\mathrm{DL})$ can easily internalize the instructions of Minsky machines. Of course, if we restrict further the number of variables or the arithmetical constraints, undecidability might be a bit less easy to obtain. QED

Theorem 2.5.3. [DD07, DG08] Satisfiability problem for $\mathrm{CLTL}(\mathrm{IPC}^*)$ is PSPACE-complete.

The proof uses the automata-based approach for LTL with an alphabet of symbolic valuations (constraints). Its difficulty comes from the fact that the sets of symbolic models that admit concrete models are not necessarily ω-regular, i.e. definable by a Büchi automaton.

2.5.2 LTL with registers: $\mathrm{LTL}^{\downarrow}$

In this section, we present the logic LTL with registers (also known as Freeze LTL) as a fragment of $\mathrm{LTL}^{\mathrm{CS}}(\mathrm{PrA})$ by restricting first-order quantification to the operators $\downarrow$ and $\uparrow$.

Logical formalisms with the freeze operator. The freeze quantifier in real-time logics has been introduced in the logic TPTL, see e.g. [AH94, Hen90, HLP90]. In spite of its rich language of constraints, TPTL model-checking is decidable [AH94] (discrete version). In this case, decidability is due to the subtle combination of the constraint system and the semantical restrictions. The formula $\mathsf{x} \cdot \varphi(\mathsf{x})$ binds the variable x to the time t of the current state: $\mathsf{x} \cdot \varphi(\mathsf{x})$ is semantically equivalent to $\varphi(t)$.

This variable-binding mechanism, quite natural when rephrased in first-order logic, is present in various logical formalisms including for example hybrid logics [Gor94, Gor96, ABM99, Are00, ABM01]: $\downarrow_{\mathsf{x}} \varphi(\mathsf{x})$ holds true iff $\varphi(\mathsf{x})$ holds true when the propositional variable x is interpreted as a singleton containing the current state. The downarrow binder in such hybrid logics records the value of the current state. Similarly, in temporal logic with forgettable past [LMS02], the effect of the `Now` operator is that the origin of time takes the value of the current state: the states before the current state are forgotten. Identical mechanisms are used in navigation logics for object structures, see e.g., [dBvE01]. In the context of spatio-temporal logics, Wolter and Zakharyaschev [WZ00, Section 7] advocate the need to consider operators expressing constraints of the form $\bigwedge_{i\in\mathbb{N}} \mathsf{x} = \mathbf{X}^i\mathsf{y}$ and $\bigvee_{i\in\mathbb{N}} \mathsf{x} = \mathbf{X}^i\mathsf{y}$. They are simple to express in LTL with registers and the above-mentioned operators have been considered in LTL-like formalisms in [DDG07].

A number of decidability and undecidability results for half-order modal logics (to be compared with [Hen90]) are presented in [Fit02]. The half-order aspect of such logics is due to a predicate λ-abstraction mechanism, which solves the famous problem of interpreting constants in modal logic. Several undecidability results for LTL-like logics with predicate λ-abstraction have recently been obtained in [LP05], independently and concurrently with [DLN05].

Last, but not least, the temporal semantics for imperative programs, may use first-order temporal logics, see e.g. [MP92]. For instance, the statement that the program variable x never decreases below its initial value can be expressed by the formula below that uses a form of freeze operator: $\exists \mathsf{y}\ (\mathsf{x} = \mathsf{y})\ \wedge\ \mathtt{G}(\mathsf{x} \geq \mathsf{y})$.

Automata for data languages and logics. In [BPT03, Bou02], data languages are defined as sets of finite data words in $(\Sigma \times D)^*$ where Σ is a finite alphabet and D is an infinite domain (generalizing the concept of timed languages), and automata which recognise data languages are introduced. First-order logic over finite data word models is considered in [BMS$^+$06], with motivations stemming from query languages for semistructured data. More precisely, the carrier of a model is the set of positions in a data word, there are no function symbols, the unary predicates correspond to elements of Σ, and there are binary predicates $<$, $+1$, as well as $\sim$ which is interpreted as equality of elements of D at given positions. FO$_k(\sim, <, +1)$ denotes such a logic with k variables. The main result of [BMS$^+$06] is that satisfiability of FO$_2(\sim, <, +1)$ is decidable, by a doubly exponential-time reduction to emptiness of multicounter automata without zero-tests (more details are provided in Chapter 3).

Definitions. The datum stored in a register is the current counter value and equality tests are performed between a register value and the current counter value. When dealing with counter systems, a register can store the value of a counter and test it later against the value of counter (possibly different from the first one). Below, we present different ways to restrict the equality tests between registers and counters. Given $n \geq 1$, the formulae of the logic LTL$^{\downarrow}[n]$ are defined as follows:

$$\varphi \ ::= \ q \ \mid \ \uparrow_r^j \mid \ \neg\varphi \ \mid \ \varphi \wedge \varphi \ \mid \ \varphi \vee \varphi \ \mid \ \varphi \mathtt{U} \varphi \ \mid \ \mathtt{X}\varphi \ \mid \ \downarrow_r^j \varphi$$

where $q \in \mathtt{Q}$, $j \in \{1, \ldots, n\}$ and $r \in \mathbb{N}^+$. An occurrence of $\uparrow_r^j$ within the scope of some freeze quantifier $\downarrow_r^c$ is *bound* by it; otherwise it is *free*. A sentence is a formula with no free occurrence of any $\uparrow_r^c$.

Models of LTL$^{\downarrow}[n]$ are ω-sequences in $(Q \times \mathbb{N}^n)^\omega$ for some finite subset $Q \subseteq \mathtt{Q}$. A *register valuation* f is a finite partial map from $\mathbb{N}^+$ to $\mathbb{N}$. Note that whenever $f(r)$ is undefined, the atomic formula $\uparrow_r^j$ is interpreted as false. Given an infinite run $\rho = (q_0, \vec{x_0}), (q_1, \vec{x_1}), \ldots$ and a position i, the satisfaction relation $\models$ is defined as follows (Boolean clauses are omitted):

$$\begin{array}{rcl}
\rho, i \models_f \ q & \stackrel{\text{def}}{\Leftrightarrow} & q_i = q \\
\rho, i \models_f \ \uparrow_r^j & \stackrel{\text{def}}{\Leftrightarrow} & r \in \mathrm{dom}(f) \text{ and } f(r) = \vec{x_i}(j) \\
\rho, i \models_f \ \mathtt{X}\varphi & \stackrel{\text{def}}{\Leftrightarrow} & \rho, i+1 \models_f \ \varphi \\
\rho, i \models_f \ \varphi_1 \mathtt{U} \varphi_2 & \stackrel{\text{def}}{\Leftrightarrow} & \text{for some } i \leq j,\ \rho, j \models_f \ \varphi_2 \\
 & & \text{and for all } i \leq j' < j, \text{ we have } \rho, j' \models_f \ \varphi_1 \\
\rho, i \models_f \ \downarrow_r^j \varphi & \stackrel{\text{def}}{\Leftrightarrow} & \rho, i \models_{f[r \mapsto \vec{x_i}(j)]} \ \varphi
\end{array}$$

$f[r \mapsto \vec{x_i}(j)]$ denotes the register valuation equal to f except that the register r is mapped to $\vec{x_i}(j)$. In the sequel, we omit the subscript "f" in $\models_f$ when sentences are involved.

For example, the formula below states that sometimes there is a value of the counter 1 such that

(1) infinitely often counter 2 takes that value if and only if infinitely often counter 3 takes that value and,

(2) from some future position, the counter 4 has always that value.

$$\mathtt{F} \downarrow_1^1 [(\mathtt{GF} \uparrow_1^2 \Leftrightarrow \mathtt{GF} \uparrow_1^3) \wedge \mathtt{FG} \uparrow_1^4]$$

We define below fragments of $\mathrm{LTL}^{\downarrow}[n]$ by restricting the use of the freeze operators. The *strict* fragment, written $\mathrm{LTL}^{\downarrow,s}[n]$, consists in associating a unique counter to each register (to store and to test). More precisely, a formula φ in $\mathrm{LTL}^{\downarrow,s}[n]$ verifies the following syntactic property: if $\downarrow_r^j \psi$ is a subformula of φ, then φ has no subformulae of the form either $\uparrow_r^{j'}$ or $\downarrow_r^{j'} \psi'$ with $j \neq j'$.

EXISTENTIAL MODEL-CHECKING PROBLEM $\mathrm{MC}^{\omega}(\mathrm{LTL}^{\downarrow})$

Input: A counter system $\mathcal{S} = (Q, n, \delta)$, an initial configuration $(q, \vec{x})$, a sentence $\varphi \in \mathrm{LTL}^{\downarrow}[n]$.

Question: Is there an infinite run ρ starting at $(q, \vec{x})$ such that $\rho, 0 \models \varphi$?

Given $n \geq 1$, we write $\mathrm{MC}^{\omega}(\mathrm{LTL}^{\downarrow}[\mathrm{n}])$ to denote the subproblem of $\mathrm{MC}^{\omega}(\mathrm{LTL}^{\downarrow})$ with counter systems of dimension at most n. In this existential version of model checking, this problem can be viewed as a variant of satisfiability in which satisfaction of a formula can be only witnessed within a specific class of data words, namely the runs of the counter machine. Note that results for the universal version of model checking will follow easily from those for the existential version when considering fragments closed under negation or deterministic counter systems.

Let us mention main results about the decidability status of satisfiability and model-checking problems. By default, satisfiability problem is related to infinite models, usually ω-sequences. The finitary satisfiability problem considers models of nonzero finite length and the temporal operators are defined accordingly in the usual way.

Theorem 2.5.4.

(I) Infinitary satisfiability problem for $\mathrm{LTL}^{\downarrow}[1]$ restricted to the temporal operators X and F and to a single register is undecidable [DL09].

(II) Finitary satisfiability problem for $\mathrm{LTL}^{\downarrow}[1]$ restricted to a single register is decidable [DL09] and its restriction to the temporal operator F and to a unique register is nonprimitive recursive [FS09].

(III) Finitary satisfiability problem for $\mathrm{LTL}^{\downarrow}[1]$ restricted to the temporal operator F and to two registers is undecidable [FS09].

By contrast, the infinitary satisfiability problem for the safety fragment of $\mathrm{LTL}^{\downarrow}[1]$ restricted to a single register is EXPSPACE-complete [Laz06].

Proof: By way of example, we show (I) by reducing the control state repeated reachability problem for gainy counter automata, shown undecidable in Section 5.2.3.

A *gainy counter automaton* is a standard counter automaton (Q, n, δ) such that for $q \in Q$ and $i \in [1, n]$, $q \xrightarrow{\mathrm{inc}(i)} q \in \delta$ (which allows us to simulate gains). In the sequel, we shall not represent these transitions. Instead, we consider that the one-step derivation relation is modified as follows: $(q, \vec{x}) \xrightarrow{t}_g (q', \vec{x'})$ iff there are $\vec{y}$ and $\vec{y'}$ in $\mathbb{N}^n$ such that $\vec{x} \preceq \vec{y}$, $(q, \vec{y}) \xrightarrow{t} (q', \vec{y'})$ (exact step) and $\vec{y'} \preceq \vec{x'}$. Observe that we can restrict ourselves to gains that occur in a lazy way: decrement on zero has no effect.

Let $\mathcal{S}$ be a gainy counter automaton with initial configuration $(q_0, \vec{0})$. For each transition t, we write $\Sigma(t)$ to denote the letter in $\Sigma = \{\mathrm{inc}(i), \mathrm{dec}(i), \mathrm{zero}(i) : i \in [1, n]\}$ labelling the transition t. We build a formula φ in $\mathrm{LTL}^{\downarrow}[1]$ such that φ is satisfiable iff $(\mathcal{S}, (q_0, \vec{0}))$ has an infinite run with q_f occurring infinitely often. The formula φ shall be satisfiable only in models in which each position is labelled by a transition from δ and by a value in $\mathbb{N}$. Infinite models of φ are of the form $(t_0, y_0), (t_1, y_1), (t_2, y_2), \cdots$ with $t_i \in \delta$ and $y_i \in \mathbb{N}$. For $I, J \in \mathbb{N}$, we write $I \sim J$ iff $y_I = y_J$.

Let us explain how the run from $(q_0, \vec{0})$

$$(q_0, \vec{x_0}) \xrightarrow{a_0} (q_1, \vec{x_1}) \xrightarrow{a_1} \cdots \xrightarrow{a_{K-1}} (q_K, \vec{x_K}) \cdots$$

is encoded. The projection of the model over δ will be precisely

$$t_0 t_1 t_2 \cdots = q_0 \xrightarrow{a_0} q_1, q_1 \xrightarrow{a_1} q_2, \cdots$$

and q_f is repeated infinitely often. This is taken care by the formulae below as conjuncts of φ:

$\star$ Initial state is q_0:

$$\bigvee_{t = q_0 \xrightarrow{a} q} t$$

$\star$ The sequence of transitions respects the control graph of the gainy counter automaton:

$$\mathtt{G}\Big(\bigwedge_{t = q \xrightarrow{a} q' \in \delta} (t \Rightarrow \mathtt{X} \bigvee_{t' = q' \xrightarrow{a} q''} t') \Big)$$

$\star$ Control state q_f is visited infinitely often:

$$\mathtt{GF} \bigvee_{t = q \xrightarrow{a} q_f} t$$

We write $\downarrow$ [resp. $\uparrow$] to denote $\downarrow^1_1$ [resp. $\uparrow^1_1$] and for each $a \in \Sigma$, we write below a to denote the following disjunction:

$$\bigvee_{t = q \xrightarrow{b} q' \in \delta,\ a = b} t$$

Moreover, we are considering the following constraints and the additional formulae are conjunctively considered in φ:

$\star$ For $i, j \in [1, n]$, there are no two positions for increments having the same value:

$$\mathtt{G}(\mathrm{inc}(i) \Rightarrow \neg(\downarrow \mathtt{XF}(\uparrow \wedge \mathrm{inc}(j))))$$

- ⋆ For $i, j \in [1, n]$, there are no two positions for decrements having the same value:

$$\mathtt{G}(\mathrm{dec}(i) \Rightarrow \neg(\downarrow \mathtt{XF}(\uparrow \wedge \mathrm{dec}(j))))$$

- ⋆ For $i \in [1, n]$ and $J > I$, if $\Sigma(t_I) = \mathrm{inc}(i)$ and $\Sigma(t_J) = \mathrm{zero}(i)$, then there is no $K > J$ such that $\Sigma(t_K) = \mathrm{dec}(i)$ and $I \sim K$:

$$\mathtt{G}(\mathrm{inc}(i) \Rightarrow\downarrow \neg(\mathtt{F}(\mathrm{zero}(i) \wedge (\mathtt{F}(\uparrow \wedge \mathrm{dec}(i))))))$$

- ⋆ For $i \in [1, n]$, if there are $J > I$ such that $\Sigma(t_I) = \mathrm{inc}(i)$ and $\Sigma(t_J) = \mathrm{zero}(i)$, then there is $K > I$ such that $\Sigma(t_K) = \mathrm{dec}(i)$ and $I \sim K$.

$$\mathtt{G}((\mathrm{inc}(i) \wedge \mathtt{F}\mathrm{zero}(i)) \Rightarrow\downarrow (\mathtt{F}(\mathrm{dec}(i)\wedge \uparrow)))$$

The last conditions are formulated in such a way to avoid using the until operator U. So, one can show that φ is satisfiable iff $(\mathcal{S}, (q_0, \vec{0}))$ has an infinite run such that q_f occurs infinitely often. QED

Now, as far as model-checking problems are concerned, the situation is not better. Reversal-bounded counter automata are defined in Chapter 4 (see also Theorem 4.4.3).

Theorem 2.5.5.

(I) The existential model-checking problem $\mathrm{MC}^{\omega}(\mathrm{LTL}^{\downarrow}[1])$ restricted to one register and to one-counter automata is undecidable [DLS10].

(II) The existential model-checking problem $\mathrm{MC}^{\omega}(\mathrm{LTL}^{\downarrow,\mathrm{s}}[4])$ restricted to reversal-bounded VASS is undecidable [DS10].

Reversal-bounded counter automata are formally defined in Chapter 4.

2.6 Exercises

Exercise 2.6.1. Show that the class of languages accepted by Büchi automata is closed under union and intersection.

Exercise 2.6.2. In the proof of Lemma 2.3.1, show that $\mathcal{A}$ and $\mathcal{A}^b$ accept the same language.

Exercise 2.6.3. Show that $\mathrm{p}\ \mathtt{U}\ \mathrm{p}'$ is equivalent to $\mathrm{p}' \vee (\mathrm{p} \wedge \mathtt{X}(\mathrm{p}\mathtt{U}\mathrm{p}'))$ in a sense that these formulae hold true exactly at the same positions in every run.

Exercise 2.6.4. Build a Büchi automaton over the alphabet $\Sigma = \{q_1, q_2, q_3\}$ that recognizes the infinite words in Σ^{ω} such that q_1 occurs infinitely often implies q_2 occurs too infinitely often.

Exercise 2.6.5. Construct a Büchi automaton for the LTL formula

$$\mathtt{G\,F}\ \mathrm{p}_1 \wedge \mathtt{G\,F}\ \mathrm{p}_2$$

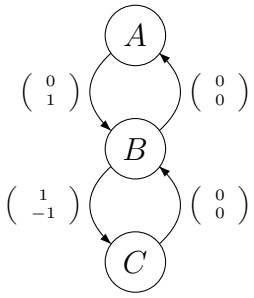
over the alphabet $\{\mathrm{p}_1, \mathrm{p}_2, \mathrm{p}_3\}$.

Exercise 2.6.6. Prove Lemma 2.5.1.

Exercise 2.6.7. Which decision problems on counter systems can be viewed as subproblems of the model-checking problem restricted to $\mathrm{LTL}^{\mathrm{CS}}(\mathrm{PrA})$ formulae without first-order quantification and the atomic formulae are reduced to control states?

Exercise 2.6.8. Let LTL^{+} be the fragment of $\mathrm{LTL}^{\mathrm{CS}}(\mathrm{PrA})$ in which the atomic formulae are either control states or Presburger formulae of the form "$\mathsf{x}_j = 0$" (zero-test on the jth counter).

1. Let us consider the VASS below.



   For each formula φ below, determine whether there is an infinite run ρ starting at $(A, \vec{0})$ such that $\rho, 0 \models \varphi$.

   (a) $\varphi = \ \mathtt{GF}\ A$,
   (b) $\varphi = \ \mathtt{GF}\ (\mathsf{x}_2 = 0)$,
   (c) $\varphi = \ \mathtt{GF}\ (\mathsf{x}_1 = 0) \wedge \ \mathtt{GF}\ C$,
   (d) $\varphi = \mathtt{G}(C \Rightarrow \mathtt{XG}\neg(\mathsf{x}_1 = 0))$,
   (e) $\varphi = \ (\mathtt{GF}\ A) \wedge \ (\mathtt{GF}\ B) \wedge \ (\mathtt{GF}\ C) \wedge \ (\mathtt{GF}\ \mathsf{x}_2 = 0) \wedge \ (\mathtt{GF}\ \neg(\mathsf{x}_1 = 0))$.

2. For which formulae φ among (a)-(e) is it the case that for all infinite runs ρ starting at $(A, \vec{0})$, we have $\rho, 0 \models \varphi$?

3. Show the LTL^{+} existential model-checking problem restricted to VASS is undecidable. For instance, reduce the halting problem for Minsky machines to it by simulating zero-tests in formulae.

Exercise 2.6.9. Design a decision procedure for the satisfiability problem for $\mathrm{LTL}^{\mathrm{CS}}(\mathrm{PrA})$ formulae in which temporal operators are not in scope of first-order quantification.

Exercise 2.6.10. Let us consider the two following fragments of $\mathrm{LTL}^{\mathrm{CS}}(\mathrm{PrA})$.

- $\star$ L_1 is a fragment of $\mathrm{LTL}^{\downarrow}[n]$ such that for any formula φ in L_1 (flatness condition):
  - $*$ if $\psi_1 \mathtt{U} \psi_2$ occurs positively in φ, then $\downarrow$ does not occur in ψ_1;
  - $*$ if $\psi_1 \mathtt{U} \psi_2$ occurs negatively in φ, then $\downarrow$ does not occur in ψ_2.
- $\star$ L_2 is the fragment of $\mathrm{CLTL(DL)}$ such that the only atomic formulae are either control states or equality tests of the form either $\mathsf{x} = \mathbf{X}\mathsf{x}'$ or $\mathsf{x} = \mathsf{x}'$.

1. Define a logspace reduction from the satisfiability problem for L_1 to the satisfiability problem for L_2.
2. Show that the satisfiability problem for L_2 can be solved in polynomial space.

Exercise 2.6.11. Show that finitary satisfiability problem for $\mathrm{LTL}^{\downarrow}[1]$ restricted to one register with the past-time temporal operator $\mathtt{F}^{-1}$ is undecidable ($\rho, i \models_f \mathtt{F}^{-1}\varphi$ iff there is $j \leq i$ such that $\rho, j \models_f \varphi$). Hint: adapt the proof of Theorem 2.5.4(I).

Chapter 3

Vector Addition Systems

In this chapter, we first show relationships between vector addition systems with states and other systems and formalisms (VAS, Petri nets, FO_2 over data words). Then, we briefly present fundamental structures to solve decision problems on VASS, namely coverability graphs. This chapter concludes by presenting a proof for the EXPSPACE upper bound of the covering problem for VASS/VAS. Such a proof is performed by an induction on the dimension that allows to shorten the length of witness runs.

3.1 VASS vs. FO_2 on Data Words

Properties on data structures can be specified by formulae from temporal logics, such as LTL, CTL or the μ-calculus, possibly equipped with predicates about data. Logics for data trees can be found in [BDM$^+$06, JL07, Fig09, Fig10] for which predicates are reduced to equality (see the survey paper [Seg06]). Similarly, logics for data words can be found in [BMS$^+$06, DL06, FS09]. These works have shed some new light between data logics and classes of counter automata. For instance, in [BMS$^+$06] it is shown how the satisfiability problem for a fragment of data logic is equivalent to the reachability problem in VASS (over finite data words), whose exact complexity is still open. The exact relationships counter automata and data logics still need to be formalized (see the recent work [BL10]). We recall below a few basic definitions and results.

Formulae of the logic $\mathrm{FO}^{\Sigma}(\sim, <, +1)$ [BMS$^+$06] where Σ is a finite alphabet are defined as follows:

$$\varphi ::= a(\mathsf{x}) \mid \mathsf{x} \sim \mathsf{y} \mid \mathsf{x} < \mathsf{y} \mid \mathsf{x} = \mathsf{y} + 1 \mid \neg\varphi \mid \varphi \wedge \varphi \mid \exists \mathsf{x}\, \varphi$$

where $a \in \Sigma$ and x, y range over a countably infinite set VAR' of variables. Variables are interpreted as positions in a (data) word; so apart from atomic formulae of the form $\mathsf{x} \sim \mathsf{y}$, formulae are similar to those of first-order logic on words, see e.g. [Str94]. We write $\mathrm{FO}(\sim, <, +1)$ to denote $\mathrm{FO}^{\Sigma}(\sim, <, +1)$ for some unspecified finite alphabet Σ. Models for $\mathrm{FO}^{\Sigma}(\sim, <, +1)$ are (finite or infinite) sequences of pairs from $\mathbb{N} \times \Sigma$ (also known as *data words* in [Bou02, BMS$^+$06]). A *variable valuation* $\mathbf{val}$ for a model σ is a map from VAR' to the indices of σ. We write $\mathbb{N}(\mathsf{x})$ to denote the natural number in the pair $\sigma(\mathbf{val}(\mathsf{x}))$ (the *datum*) and $\Sigma(\mathsf{x})$ to denote its letter (the *label*).

Given a finite alphabet $\Sigma = \{a_1, \ldots, a_N\}$ and an infinite domain D, a finite data word $(a_{i_1}, d_1) \cdots (a_{i_K}, d_K)$ can be viewed as the structure $(\{1, \ldots, K\}, <, \sim, +1, P_1, \ldots, P_N)$ such that

- for $j, j' \in \{1, \ldots, K\}$, $j \sim j'$ iff $d_j = d_{j'}$,
- for $j \in \{1, \ldots, K\}$ and $l \in \{1, \ldots, N\}$, $j \in P_l$ iff $a_{i_j} = a_l$.

The satisfaction relation $\models$ is defined as follows (Boolean clauses are omitted):

$$\begin{array}{rcl}
\sigma \models_{\mathbf{val}} a(\mathsf{x}) & \stackrel{\text{def}}{\Leftrightarrow} & \Sigma(\mathsf{x}) = a \\
\sigma \models_{\mathbf{val}} \mathsf{x} \sim \mathsf{y} & \stackrel{\text{def}}{\Leftrightarrow} & \mathbb{N}(\mathsf{x}) = \mathbb{N}(\mathsf{y}) \\
\sigma \models_{\mathbf{val}} \mathsf{x} < \mathsf{y} & \stackrel{\text{def}}{\Leftrightarrow} & \mathbf{val}(\mathsf{x}) < \mathbf{val}(\mathsf{y}) \\
\sigma \models_{\mathbf{val}} \mathsf{x} = \mathsf{y} + 1 & \stackrel{\text{def}}{\Leftrightarrow} & \mathbf{val}(\mathsf{x}) = \mathbf{val}(\mathsf{y}) + 1 \\
\sigma \models_{\mathbf{val}} \exists\, \mathsf{x}\, \varphi & \stackrel{\text{def}}{\Leftrightarrow} & \text{there is } i < |\sigma| \text{ such that } \sigma \models_{\mathbf{val}[\mathsf{x} \mapsto i]} \varphi.
\end{array}$$

Here $\mathbf{val}[\mathsf{x} \mapsto i]$ denotes the variable valuation equal to $\mathbf{val}$ except that the variable x is mapped to the position i. $\mathrm{FO}_2^{\Sigma}(\sim, <, +1)$ is defined as the fragment of $\mathrm{FO}^{\Sigma}(\sim, <, +1)$ restricted to two individual variables. For example, the formula below in $\mathrm{FO}_2^{\Sigma}(\sim, <, +1)$ states that there are no two distinct positions labelled by a having the same datum:

$$\forall \mathsf{x}\, \mathsf{y}\ (\mathsf{x} < \mathsf{y} \wedge a(\mathsf{x}) \wedge a(\mathsf{y})) \Rightarrow \neg(\mathsf{x} \sim \mathsf{y}).$$

The finitary [resp. infinitary] satisfiability problem for $\mathrm{FO}_2^{\Sigma}(\sim, <, +1)$ is to check whether a sentence from $\mathrm{FO}_2^{\Sigma}(\sim, <, +1)$ has a finite [resp. infinite] model. Even though satisfiability for $\mathrm{FO}(\sim, <, +1)$ restricted to three individual variables is undecidable [BMS+06, Dav09], decidability can be regained with only two variables.

Theorem 3.1.1. [BMS+06, Dav09, BMSS09] The finitary and infinitary satisfiability problems for $\mathrm{FO}_2(\sim, <, +1)$ are decidable.

Let us sketch how decidability for finitary satisfiability is shown in [BMS+06]. Satisfiability is first reduced to nonemptiness for data automata, which in turn is reduced to reachability problem for vector addition systems with states. There is also a reduction in the other direction.

Theorem 3.1.2. [BMS+06, Dav09] There is a polynomial-space reduction from the reachability problem for vector addition systems with states to finitary satisfiability for $\mathrm{FO}_2(\sim, <, +1)$.

Proof: First, the reachability problem for vector addition systems with states can be reduced in polynomial-space to its restriction such that the initial and final configurations have all the counters equal to zero and each transition can only increment or decrement a single counter. In the sequel, we consider an instance of this subproblem: $\mathcal{S} = (Q, n, \delta)$ is a VASS, the initial configuration is $(q_i, \vec{0})$, and the final configuration is $(q_f, \vec{0})$. Indeed, transitions can be restricted to increments of decrements of a single counter. For example, the translation $\left(\begin{smallmatrix} 2 \\ -3 \end{smallmatrix}\right)$ can be encoded by 2 increments of the first counter followed by 3 decrements of the second counter. Similarly, initial and final configurations $(q_0, \left(\begin{smallmatrix} 2 \\ 1 \end{smallmatrix}\right))$ and $(q_f, \left(\begin{smallmatrix} 1 \\ 1 \end{smallmatrix}\right))$ can be reduced to $(q_0', \left(\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}\right))$ and $(q_f', \left(\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}\right))$ respectively, by adding the transitions

$$q_0' \xrightarrow{\mathrm{inc}(1)} q_0^1 \xrightarrow{\mathrm{inc}(1)} q_0^2 \xrightarrow{\mathrm{inc}(2)} q_0 \quad q_f \xrightarrow{\mathrm{dec}(1)} q_f^1 \xrightarrow{\mathrm{dec}(2)} q_f'$$

All these reductions require only polynomial space. So, we can assume that $\mathcal{S}$ is a standard counter automaton without zero-tests.

Now, we shall build a formula φ in $\mathrm{FO}_2(\sim, <, +1)$ such that the final configuration is reachable from the initial configuration iff φ is satisfiable. To do so, we encode runs of $\mathcal{S}$ by data words in the following way. The alphabet Σ is defined as the set $Q \uplus \{\mathrm{inc}(i), \mathrm{dec}(i) : i \in [1, n]\}$.

Let us explain how the run

$$(q_0, \vec{x_0}) \xrightarrow{a_0} (q_1, \vec{x_1}) \xrightarrow{a_1} \cdots \xrightarrow{a_{K-1}} (q_K, \vec{x_K})$$

is encoded. The projection of the data word over the alphabet Σ will be precisely

$$q_0 a_0 q_1 a_1 \cdots a_{K-1} q_K$$

The formula φ_{proj} states that the projection corresponds to an accepting run of $\mathcal{S}$, leaving away the fact that counter values must belong to $\mathbb{N}$:

- ⋆ The first letter is q_i: $\exists \mathsf{x}\ (\neg \exists \mathsf{y}\ \mathsf{y} < \mathsf{x}) \wedge q_i(\mathsf{x})$.
- ⋆ The last letter is q_f: $\exists \mathsf{x}\ (\neg \exists \mathsf{y}\ \mathsf{x} < \mathsf{y}) \wedge q_f(\mathsf{x})$.
- ⋆ The sequence of locations and actions respects the control graph of $\mathcal{S}$:

$$\forall\, \mathsf{x}\ (\bigvee_{q \in Q} q(\mathsf{x})) \Rightarrow ((\neg \exists \mathsf{y}\ \mathsf{x} < \mathsf{y}) \vee$$

$$\bigvee_{q \xrightarrow{a} q' \in \delta} (q(\mathsf{x}) \wedge (\exists \mathsf{y}\ \mathsf{y} = \mathsf{x} + 1 \wedge a(\mathsf{y})) \wedge (\exists \mathsf{y}\ \mathsf{y} = \mathsf{x} + 1 \wedge (\exists \mathsf{x}\ \mathsf{x} = \mathsf{y} + 1 \wedge q'(\mathsf{x})))))$$

Observe the nice (and standard) recycling of variables. Moreover, we shall impose constraints on data values. For instance, the run

$$\begin{array}{ccccccc} q_0 & q_1 & q_2 & q_3 & q_4 & q_5 & q_6 \\ \left(\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}\right) & \left(\begin{smallmatrix} 1 \\ 0 \end{smallmatrix}\right) & \left(\begin{smallmatrix} 2 \\ 0 \end{smallmatrix}\right) & \left(\begin{smallmatrix} 2 \\ 1 \end{smallmatrix}\right) & \left(\begin{smallmatrix} 1 \\ 1 \end{smallmatrix}\right) & \left(\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}\right) & \left(\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}\right) \end{array}$$

corresponds to a data word of the form below

q_0	$\mathrm{inc}(1)$	q_1	$\mathrm{inc}(1)$	q_2	$\mathrm{inc}(2)$	q_3	$\mathrm{dec}(1)$	q_4	$\mathrm{dec}(1)$	q_5	$\mathrm{dec}(2)$	q_6
⋆	k_1	⋆	k_2	⋆	k_3	⋆	k_1	⋆	k_2	⋆	k_3	⋆

⋆ denotes an arbitrary data value. The main idea is to observe that each action is attached to a data value and the data value for a decrement has to occur already in the past for an increment. We impose the following constraints, taken conjunctively with φ_{proj} we obtain the formula φ:

- ⋆ For $i, j \in [1, n]$, there are no two positions labelled by $\mathrm{inc}(i)$ and $\mathrm{inc}(j)$ having the same datum:
$$\forall \mathsf{x}\ \mathsf{y}\ (\mathsf{x} < \mathsf{y} \wedge \mathrm{inc}(i)(\mathsf{x}) \wedge \mathrm{inc}(j)(\mathsf{y})) \Rightarrow \neg(\mathsf{x} \sim \mathsf{y}).$$
- ⋆ For $i, j \in [1, n]$, there are no two positions labelled by $\mathrm{dec}(i)$ and $\mathrm{dec}(j)$ having the same datum:
$$\forall \mathsf{x}\ \mathsf{y}\ (\mathsf{x} < \mathsf{y} \wedge \mathrm{dec}(i)(\mathsf{x}) \wedge \mathrm{dec}(j)(\mathsf{y})) \Rightarrow \neg(\mathsf{x} \sim \mathsf{y}).$$

- ⋆ For $i \in [1,n]$, for every position labelled by $\mathrm{dec}(i)$, there is a past position labelled by $\mathrm{inc}(i)$ with the same datum:

$$\forall \mathsf{x}\ \mathrm{dec}(i)(\mathsf{x}) \Rightarrow (\exists\ \mathsf{y}\ \mathsf{y} < \mathsf{x} \wedge \mathsf{x} \sim \mathsf{y} \wedge \mathrm{inc}(i)(\mathsf{y}))$$

- ⋆ Since in the final configuration, any counter value is zero, we impose that for $i \in [1,n]$, for every position labelled by $\mathrm{inc}(i)$, there is a future position labelled by $\mathrm{dec}(i)$ with the same datum:

$$\forall \mathsf{x}\ \mathrm{inc}(i)(\mathsf{x}) \Rightarrow (\exists\ \mathsf{y}\ \mathsf{x} < \mathsf{y} \wedge \mathsf{x} \sim \mathsf{y} \wedge \mathrm{dec}(i)(\mathsf{y}))$$

One can then show that $(q_f, \vec{0})$ is reachable from $(q_i, \vec{0})$ iff φ is satisfiable. QED

It is worth noting that a fragment of $\mathrm{LTL}^{\downarrow}[1]$ with one register has been shown equivalent to $\mathrm{FO}_2(\sim, <, +1)$ in [DL09].

3.2 Relationships with Petri Nets

In this section, we show how Petri nets (see e.g. [RR98]) are related to VASS and VAS. First, let us recall that a *Petri net* N is a structure (S, T, W, m_I) such that S is a finite set of *places*, T is a finite set of *transitions*, $W : (S \times T) \cup (T \times S) \to \mathbb{N}$ is a *weight function*. A *marking* m is a map of the form $S \to \mathbb{N}$: for each place, we specify a number of *tokens* (possibly none). In the Petri net N, $m_I : S \to \mathbb{N}$ is the initial marking (initial distribution of tokens). We assume that the reader is familiar with the semantics of this model (otherwise see e.g., [Pet81, RR98]). We just recall below a few definitions. A transition $t \in T$ is *m-enabled*, written $m \xrightarrow{t}$, whenever for all places $p \in S$, $m(p) \geq W(p,t)$. An m-enabled transition t may fire and produce the marking m', written $m \xrightarrow{t} m'$, with for all places $p \in S$, $m'(p) = m(p) - W(p,t) + W(t,p)$. A marking m' is *reachable* from m whenever there is a sequence of the form $m_0 \xrightarrow{t_0} m_1 \xrightarrow{t_1} \cdots \xrightarrow{t_{k-1}} m_k$ with $m_0 = m$ and $m_k = m'$ (also written $m \xrightarrow{t_0 \cdots t_{k-1}} m'$).
Here are standard problems for Petri nets.
REACHABILITY PROBLEM FOR PETRI NETS:

Input: a Petri net (S, T, W, m_I) and a marking m.

Question: is m reachable from m_I?

COVERING PROBLEM FOR PETRI NETS:

Input: a Petri net (S, T, W, m_I) and a marking m.

Question: is there a marking m' reachable from m_I such that for all $p \in S$, we have $m'(p) \geq m(p)$?

BOUNDEDNESS PROBLEM FOR PETRI NETS:

Input: a Petri net (S, T, W, m_I).

Question: is the set of markings reachable from m_I infinite?

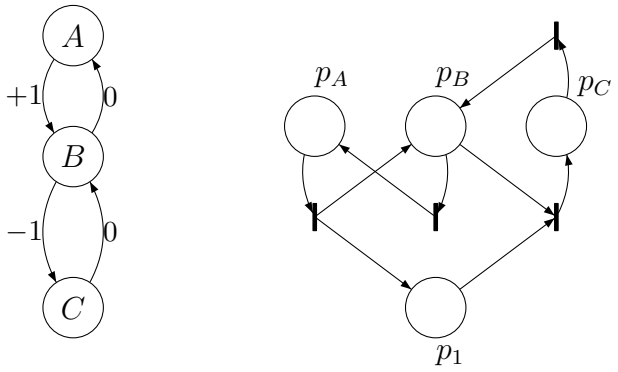


Figure 3.1: A VASS and an equivalent Petri net

Lemma 3.2.1. The reachability problem for VASS is equivalent to the reachability problem for Petri nets. The same holds true for the covering and boundedness problems.

Proof: First, let us show how to simulate a VASS by a Petri net, which allows us to get a logspace many-one reduction for the reachability, boundedness and covering problems. Let $\mathcal{V}$ be a VASS (Q, n, δ), $(q_I, \vec{x_I})$ and $(q_F, \vec{x_F})$ be two configurations. We can build a Petri net $N_{\mathcal{V}}$ that simulates $\mathcal{V}$, by using a standard translation from VASS to Petri nets. For every control state q in $\mathcal{V}$, we introduce a place p_q in $N_{\mathcal{V}}$ and for $i \in [1, n]$, we introduce a place p_i. An initial marking m_I contains one token in the place p_{q_I} and for $i \in [1, n]$, $m_I(p_i) = \vec{x_I}(i)$. From this marking, we only obtain markings where a unique token belongs to a place of the form p_q ($q \in Q$) which means that a unique control state is active for every marking. For every transition in $\mathcal{V}$, say $t = q \xrightarrow{\vec{b}} q'$, we consider a transition $\mathbf{t}$ in $N_{\mathcal{V}}$ that consumes a token in p_q, produces a token in $p_{q'}$ and produces [resp. consumes] $\vec{b}(i)$ tokens in the place p_i when $\vec{b}(i) \geq 0$ [resp. when $\vec{b}(i) < 0$].

In Figure 3.1, we present a VASS of dimension 1 with three control states and its equivalent Petri net with the above-mentionned construction. In order to ensure the correctness of our reduction, we need first to handle transitions of the form $t = q \xrightarrow{\vec{b}} q$ (*self-loop*) in $\mathcal{V}$ separately as follows. We transform a transition of the form $t = q \xrightarrow{\vec{b}} q$ in $\mathcal{V}$ by the two transitions $q \xrightarrow{\vec{0}} q_{new}$ and $q_{new} \xrightarrow{\vec{b}} q$ in $N_{\mathcal{V}}$, where q_{new} is a new location designed for q and $\vec{0}$ is the zero vector.

When $\mathcal{V}$ is self-loop free, for all configurations $(q, \vec{x})$, the propositions below are equivalent:

- there is a run of the form $(q_I, \vec{x_I}) \xrightarrow{t_1} \cdots \xrightarrow{t_k} (q, \vec{x})$ in $\mathcal{V}$,
- there exists a sequence of transitions $u = \mathbf{t_1} \cdots \mathbf{t_k}$ such that $m_0 \xrightarrow{u} m$ where for $q' \in Q \setminus \{q\}$, $m(p_{q'}) = 0$, $m(p_q) = 1$ and for $i \in [1, n]$, $m(p_i) = \vec{x}(i)$.

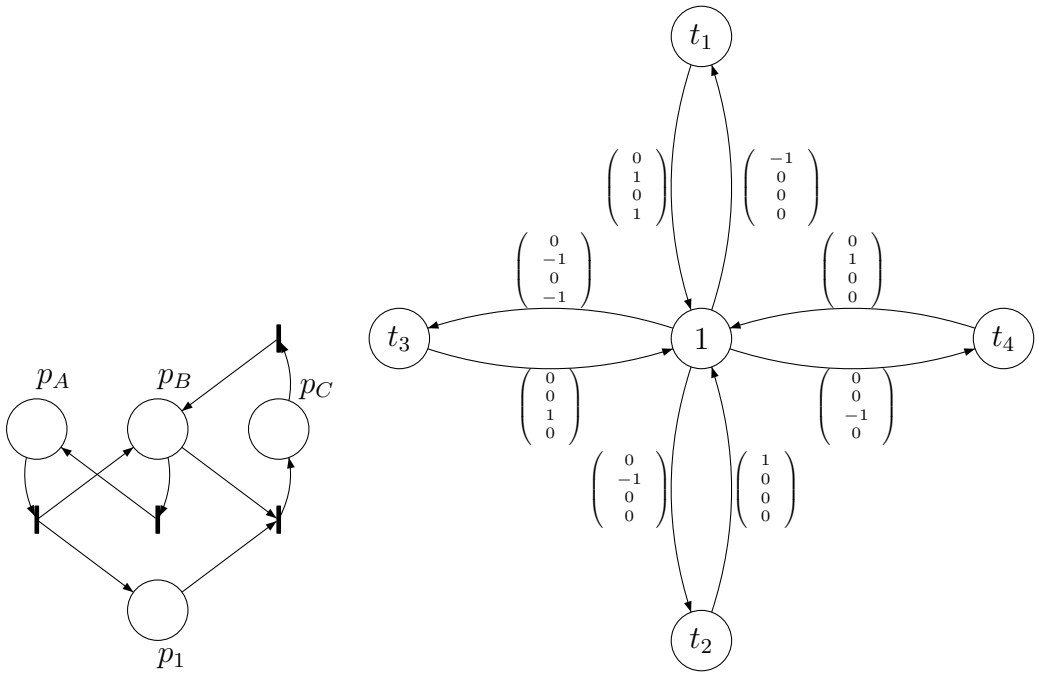


Figure 3.2: A Petri net and an equivalent VASS

Now, let (S, T, W, m_I) be a Petri net. Let us define the VASS $\mathcal{V} = (Q, n, \delta)$ as follows (assuming an arbitrary bijection $f : \{1, \ldots, \mathrm{card}(S)\} \to S$):

- $Q = \{1\} \uplus T$,
- $n = \mathrm{card}(S)$,
- for each transition t, we introduce two transitions in $\mathcal{V}$, namely $\mathbf{t} = 1 \xrightarrow{\vec{b-}} t$ and $\mathbf{t}' = t \xrightarrow{\vec{b+}} 1$ such that for $i \in [1, n]$, $\vec{b-}(i) = -W(f(i), t)$ and $\vec{b+}(i) = W(t, f(i))$.

Let $(q_I, \vec{x_I})$ be the configuration of $\mathcal{V}$ such that $q_I = 1$ and for $i \in [1, n]$, $\vec{x_I}(i) = m_I(f(i))$.

In Figure 3.2, we present a Petri net with four places and four transitions and its equivalent VASS with the above-mentionned construction (with $f(p_A) = 1$, $f(p_B) = 2$, $f(p_C) = 3$ and $f(p_1) = 4$). Again, we can show that for all markings m, the propositions below are equivalent:

- there exists a sequence of transitions $u = \mathbf{t_1} \cdots \mathbf{t_k}$ such that $m_0 \xrightarrow{u} m$,
- there is a run of the form $(q_I, \vec{x_I}) \xrightarrow{\mathbf{t_1 t_1}'} \cdots \xrightarrow{\mathbf{t_k t_k}'} (q_k, \vec{x_k})$ in $\mathcal{V}$, where $q_k = 1$ and for $i \in [1, n]$, $\vec{x_k}(i) = m(f(i))$.

QED

Lemma 3.2.2. The reachability problem for VASS is equivalent to the reachability problem for VAS. The same holds true for the covering and boundedness problems.

Proof: Since a VAS is a VASS with a unique control state, we only need to show how to simulate a VASS by a VAS. As in the proof of Lemma 3.2.1, without any loss of generality, we can assume that the VASS is without self-loop.

Let $\mathcal{V} = (Q, n, \delta)$ be a VASS without self-loop and h be an arbitrary bijection from Q to $\{n+1, \ldots, n+\mathrm{card}(Q)\}$. The bijection h is dedicated to relate each control state of $\mathcal{V}$ with a unique component in the VAS we shall build.

Let X be subset of $\mathbb{N}^{n+\mathrm{card}(Q)}$ such that

$$X = \{\vec{x} \in \mathbb{N}^{n+\mathrm{card}(Q)} : \vec{x}([n+1, n+\mathrm{card}(Q)]) = e_i \in \mathbb{N}^{\mathrm{card}(Q)} \text{ for some } i \in [1, \mathrm{card}(Q)]\},$$

where $\vec{x}([n+1, n+\mathrm{card}(Q)])$ is the tuple in $\mathbb{N}^{\mathrm{card}(Q)}$ restricted to the $\mathrm{card}(Q)$ last components of $\vec{x}$ and $e_i \in \mathbb{N}^{\mathrm{card}(Q)}$ is a unit element with 1 for the ith component and zero otherwise.

For the VASS in Figure 3.1, we have $X = \mathbb{N} \times \{(1,0,0), (0,1,0), (0,0,1)\}$.

Let $\mathcal{T}$ be a VAS such that for $t = q \xrightarrow{\vec{b}} q' \in \delta$ ($q \neq q'$ by hypothesis), the transition $t' \in \mathcal{T}$ is defined as follows:

- $(t')([1,n]) = \vec{b}$,
- for $q'' \in Q \setminus \{q, q'\}$, $t'(h(q'')) = 0$,
- $t'(h(q)) = -1$ and $t'(h(q')) = 1$.

For the VASS in Figure 3.1, we have following set of transitions in the corresponding VAS:

$$(1,-1,1,0),\ (-1,0,-1,1),\ (0,0,1,-1),\ (0,1,-1,0)$$

Let f be the bijection between the configurations of $\mathcal{V}$ and X such that

$$f((q,\vec{x}))([n+1, n+\mathrm{card}(Q)]) = e_{h(q)} \text{ and } f((q,\vec{x}))([1,n]) = \vec{x}.$$

For the VASS in Figure 3.1, we have for instance that $f((B,3)) = (3,0,1,0)$ and $f((A,8)) = (8,1,0,0)$.

For each run $(q_0, \vec{x}_0) \ldots (q_k, \vec{x}_k)$ of $\mathcal{V}$, it is possible to associate the run

$$f((q_0, \vec{x}_0)) \ldots f((q_k, \vec{x}_k))$$

in $\mathcal{T}$. One can check that each configuration $f((q_i, \vec{x}_i))$ belongs to X. Similarly, for each run $\vec{x}_0 \cdots \vec{x}_k$ in $\mathcal{T}$, the sequence $f^{-1}(\vec{x}_0) \cdots f^{-1}(\vec{x}_k)$ is a run of $\mathcal{V}$.

First, observe that $(q', \vec{x'})$ is reachable from $(q, \vec{x})$ in $\mathcal{V}$ iff $f((q', \vec{x'}))$ is reachable from $f((q, \vec{x}))$ in $\mathcal{V}$. This can be easily shown by induction on the lenght of the run. So, the reachability problem for VASS can be reduced to the reachability problem for VAS.

Furthermore, given a configuration $(q, \vec{x})$ for $\mathcal{V}$ and a control state q', the propositions below are equivalent:

- in $\mathcal{V}$, there is a run of the form $(q, \vec{x}), \ldots, (q', \vec{x'})$,
- in $\mathcal{T}$, there is a run of the form $f((q, \vec{x})), \ldots, \vec{x''}$ with $f((q', \vec{0})) \preceq \vec{x''}$ (which is equivalent to the fact that the control state of $f^{-1}(\vec{x''})$ is q').

Consequently, the covering problem for VASS can be reduced to the covering problem for VAS. Similarly, given a configuration $(q, \vec{x})$ for $\mathcal{V}$, the propositions below are equivalent:

- $\star$ the set of configurations reachable from $(q, \vec{x})$ (in $\mathcal{V}$) is infinite,
- $\star$ the set of configurations reachable from $f((q, \vec{x}))$ (in $\mathcal{T}$) is infinite.

Whereas one direction is obvious, observe that for any configuration reachable from $f((q, \vec{x}))$, its restriction to the $\mathrm{card}(Q)$ last components can take at most $\mathrm{card}(Q)$ distinct values. Consequently, if the set of configurations reachable from $f((q, \vec{x}))$ is infinite, then there is a location q' such that there are an infinite amount of configurations reachable from $f((q, \vec{x}))$ of the form $f((q', \vec{y}))$. Hence, this entails that the set of configurations reachable from $(q, \vec{x})$ is infinite. So, the boundedness problem for VASS can be reduced to the boundedness problem for VAS. QED

In [HP79], it is shown that VAS can simulate VASS by adding at most 3 to the dimension, whereas in the above proof the dimension is augmented by the number of control states.

3.3 Coverability Graphs in a Nutshell

We recall that a VAS $\mathcal{T}$ of dimension n can be encoded as a finite subset of $\mathbb{Z}^n$.

In this section, we recall the main definitions and properties about coverability graphs, see e.g. [KM69]; for instance decidability of the covering and boundedness problems can be obtained from its properties. However, in Section 3.4, we shall provide the optimal complexity upper bound for the covering problem. A coverability graph shall approximate the set of reachable configurations from a given configuration and it is a finite structure that can be effectively computed.

Let us start by preliminary definitions. Let us consider the structure $(\mathbb{N} \cup \{\infty\}, \leq)$ such that for $k, k' \in \mathbb{N} \cup \{\infty\}$, $k \leq k' \stackrel{\text{def}}{\Leftrightarrow}$ either $k, k' \in \mathbb{N}$ and $k \leq k'$ or $k' = \infty$. We write $k < k'$ whenever $k \leq k'$ and $k \neq k'$. The ordering $\leq$ can be naturally extended to tuples in $(\mathbb{N} \cup \{\infty\})^n$ by defining it component-wise: $\vec{x}, \vec{x'} \in (\mathbb{N} \cup \{\infty\})^n$, $\vec{x} \leq \vec{x'} \stackrel{\text{def}}{\Leftrightarrow}$ for $i \in [1, n]$, either $\vec{x}(i), \vec{x'}(i) \in \mathbb{N}$ and $\vec{x}(i) \leq \vec{x'}(i)$ or $\vec{x'}(i) = \infty$. We also write $\vec{x} < \vec{x'}$ when $\vec{x} \leq \vec{x'}$ and $\vec{x} \neq \vec{x'}$. Given $\vec{x}, \vec{x'} \in (\mathbb{N} \cup \{\infty\})^n$ such that $\vec{x} < \vec{x'}$, we write $acc(\vec{x}, \vec{x'})$ to denote the element of $(\mathbb{N} \cup \{\infty\})^n$ such that for $i \in [1, n]$, if $\vec{x}(i) = \vec{x'}(i)$ then $acc(\vec{x}, \vec{x'})(i) \stackrel{\text{def}}{=} \vec{x'}(i)$, otherwise $acc(\vec{x}, \vec{x'})(i) \stackrel{\text{def}}{=} \infty$. For instance,

$$acc(\begin{pmatrix} 2 \\ 3 \\ 1 \end{pmatrix}, \begin{pmatrix} 2 \\ 4 \\ 1 \end{pmatrix}) = \begin{pmatrix} 2 \\ \infty \\ 1 \end{pmatrix}.$$

Let us conclude this paragraph by a last definition. For $\vec{x} \in (\mathbb{N} \cup \{\infty\})^n$ and $t \in \mathbb{Z}^n$, $\vec{x} + t$ is defined as an element of $(\mathbb{Z} \cup \{\infty\})^n$ such that for $i \in [1, n]$, if $\vec{x}(i) \in \mathbb{N}$ then $(\vec{x} + t)(i) \stackrel{\text{def}}{=} \vec{x}(i) + t(i)$, otherwise $(\vec{x} + t)(i) \stackrel{\text{def}}{=} \infty$. For instance, $\begin{pmatrix} 2 \\ \infty \\ 1 \end{pmatrix} + \begin{pmatrix} -3 \\ -6 \\ 2 \end{pmatrix} = \begin{pmatrix} -1 \\ \infty \\ 3 \end{pmatrix}$.

Given a VAS $\mathcal{T}$ of dimension n and a configuration $\vec{x_0}$, we shall define a coverability graph $CG(\mathcal{T}, \vec{x_0})$ as a structure (V, E) such that $V \subseteq (\mathbb{N} \cup \{\infty\})^n$ and $E \subseteq V \times \mathcal{T} \times V$. Here are essential properties of $CG(\mathcal{T}, \vec{x_0})$:

(a) $CG(\mathcal{T}, \vec{x_0})$ is a finite structure. This a consequence of König's Lemma (any infinite finite-branching tree has an infinite branch) and Dickson's Lemma (for any infinite sequence $\vec{z_0}, \ldots \vec{z_i}, \ldots$ of tuples of $\mathbb{N}^n$, there exist $i < j$ such that $\vec{z_i} \preceq \vec{z_j}$).

(b) For any configuration $\vec{y}$ reachable from $\vec{x_0}$ in $\mathcal{T}$, there is $\vec{y'}$ in $CG(\mathcal{T}, \vec{x_0})$ such that $\vec{y} \preceq \vec{y'}$. Otherwise said, any reachable configuration can be covered by an element of $CG(\mathcal{T}, \vec{x_0})$. Moreover, if $\vec{y}$ reachable from $\vec{x_0}$ with the sequence of transitions σ, then there is path labelled by σ between $\vec{x_0}$ and $\vec{y'}$.

(c) For every extended configuration $\vec{y'}$ in $CG(\mathcal{T}, \vec{x_0})$ and bound $B \in \mathbb{N}$, there is a configuration $\vec{y}$ reachable from $\vec{x_0}$ in $\mathcal{T}$ such that for $i \in [1, n]$, if $\vec{y'}(i) = \infty$ then $\vec{y}(i) \geq B$ otherwise $\vec{y}(i) = \vec{y'}(i)$.

Consequently, we can solve the boundedness and covering problems from the construction of the coverability graph thanks to the following equivalences.

Lemma 3.3.1. We have the following charaterizations.

(I) There is some configuration $\vec{x''}$ reachable from $\vec{x_0}$ such that $\vec{x'} \preceq \vec{x''}$ iff there is $\vec{y}$ in $CG(\mathcal{T}, \vec{x_0})$ such that $\vec{x'} \preceq \vec{y}$.

(II) The set of configurations reachable from $\vec{x_0}$ is infinite iff there is an extended configuration in $CG(\mathcal{T}, \vec{x_0})$ with at least one component equal to ∞.

(III) Every run from $\vec{x_0}$ terminates iff there is no cycle in $CG(\mathcal{T}, \vec{x_0})$.

Unfortunately, even though $CG(\mathcal{T}, \vec{x_0})$ is finite, in the worst-case its number of nodes can be nonprimitive recursive [VVN81]. Let us precise what it means by recalling a variant of Ackermann function:

- $A_0(m) = 2m + 1$, $A_{n+1}(0) = 1$.
- $A_{n+1}(m + 1) = A_n(A_{n+1}(m))$.
- $A(n) = A_n(2)$.

The function $A(n)$ majorizes the primitive recursive functions and the size of the coverability graph can be in $\mathcal{O}(A(n))$ where n is the size of $\mathcal{T}$ and $\vec{x_0}$.

Proof: By way of example, we show (I) and (II).

(I) Suppose that $\vec{x''}$ reachable from $\vec{x_0}$ and $\vec{x'} \preceq \vec{x''}$. By (b), there is $\vec{y}$ in $CG(\mathcal{T}, \vec{x_0})$ such that $\vec{x''} \preceq \vec{y}$. Since $\preceq$ is transitive on $(\mathbb{N} \cup \{\infty\})^n$, we get $\vec{x'} \preceq \vec{y}$. Conversely, suppose that there is $\vec{y}$ in $CG(\mathcal{T}, \vec{x_0})$ such that $\vec{x'} \preceq \vec{y}$. Let B be the maximal value occurring in $\vec{x'}$. By (c), there is a configuration $\vec{y'}$ reachable from $\vec{x_0}$ in $\mathcal{T}$ such that for $i \in [1, n]$, if $\vec{y}(i) = \infty$ then $\vec{y'}(i) \geq B$ otherwise $\vec{y'}(i) = \vec{y}(i)$. Hence, $\vec{x'} \preceq \vec{y'}$.

(II) Suppose that the set of configurations reachable from $\vec{x_0}$ is infinite. *Ad absurdum*, assume that ∞ does not occur in $CG(\mathcal{T}, \vec{x_0})$. By (b), there is $\vec{y}$ in $CG(\mathcal{T}, \vec{x_0})$ such that for an infinite amount of configurations $\vec{x}$ reachable from $\vec{x_0}$, we have $\vec{x} \preceq \vec{y}$. This leads to a contradiction since there are at most $(1 + \max(\vec{y}))^n$ distinct configurations smaller than $\vec{y}$ since $\vec{y} \in \mathbb{N}^n$. If ∞ occurs in $CG(\mathcal{T}, \vec{x_0})$, then by (c) the set of configurations reachable from $\vec{x_0}$ is infinite (consider bounds B greater and greater when applying (c)).

(III) Use the second part of (b). QED

Boundedness problem as a subproblem of existential model-checking problem. Thanks to Lemma 3.3.1(II), an instance of the boundedness problem can be viewed as an instance of the existential model-checking with the $\mathrm{LTL}^{\mathrm{CS}}(\mathrm{PrA})$ formula

$$\bigvee_{q\in Q} \mathtt{F}(\exists\, \mathsf{y}_1,\ldots,\mathsf{y}_n\ (\bigwedge_i \mathsf{y}_i = \mathsf{x}_i) \wedge q \wedge \mathtt{F}((\bigwedge_i \mathsf{y}_i \leq \mathsf{x}_i) \wedge (\bigvee_i \mathsf{y}_i < \mathsf{x}_i) \wedge q))$$

Existential model-checking for $\mathrm{LTL}^{\mathrm{CS}}(\mathrm{PrA})$ for VASS can be easily shown to be undecidable and the boundedness problem is known to be EXPSPACE-complete. It is unclear what are the interesting maximal fragments of $\mathrm{LTL}^{\mathrm{CS}}(\mathrm{PrA})$ for which existential model-checking problem is decidable or in EXPSPACE (see related works in [Yen92, AH09, Dem10]).

Construction of coverability graphs. Let us define $CG(\mathcal{T}, \vec{x_0})$ by building incrementally V and E.

1. $E := \emptyset$; $V := \emptyset$;
2. $ToBeTreated := \{\vec{x_0}\}$;
3. while $ToBeTreated \neq \emptyset$ do
   - ⋆ Select an element $\vec{x}$ from $ToBeTreated$;
   - ⋆ $ToBeTreated := ToBeTreated \setminus \{\vec{x}\}$;
   - ⋆ for $t \in \mathcal{T}$ such that $\vec{x} + t \in (\mathbb{N} \cup \{\infty\})^n$ do
     - ∗ $\vec{x'} := \vec{x} + t$;
     - ∗ if there is $\vec{y} \in V$ s.t. $\vec{y} \xrightarrow{*} \vec{x}$ in (V, E) and $\vec{y} < \vec{x'}$ then
       - · Let $\vec{y_0}$ be the extended configuration the closest to $\vec{x}$ in (V, E) such that $\vec{y_0} < \vec{x'}$;
       - · $\vec{x'} := acc(\vec{y_0}, \vec{x'})$;
     - ∗ if $\vec{x'} \notin V$ then
       - · $V := V \cup \{\vec{x'}\}$;
       - · $ToBeTreated := ToBeTreated \cup \{\vec{x'}\}$;
     - ∗ $E := E \cup \{\vec{x} \xrightarrow{t} \vec{x'}\}$;

Figure 3.3 contains a VASS (that can be easily seen as a VAS using previous developments) and a coverability graph for the initial configuration $(0, 1, 0, 0)$.

3.4 Solving the Covering Problem in Exponential Space

In this section, we present the proof establishing that the covering problem for VAS can be solved in exponential space [Rac78]. This completes the worst-case complexity charaterization of the problem since Lipton has shown earlier that the problem is EXPSPACE-hard [Lip76] (see also a more accessible proof in [Esp98]). The result is not only interesting for complexity purpose but

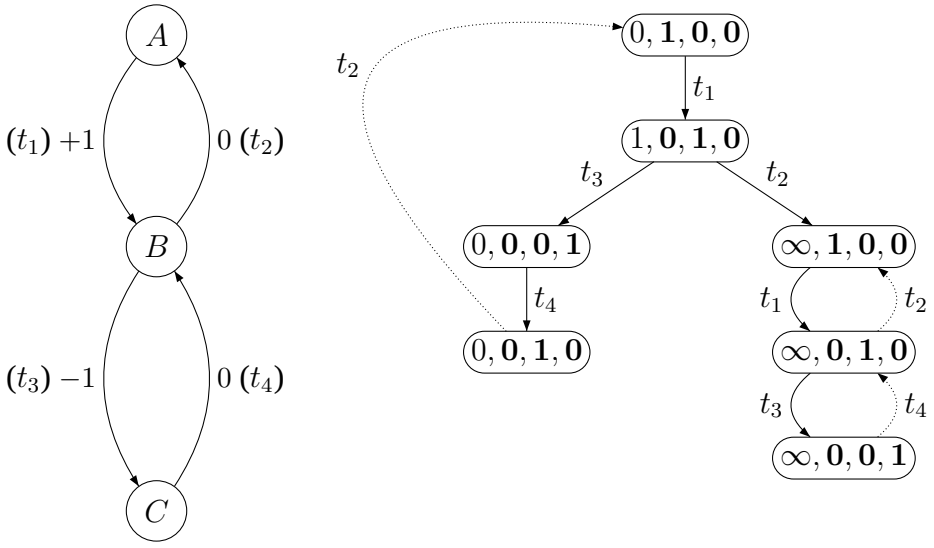


Figure 3.3: A VASS/VAS and a coverability graph

also because its proof uses a simple induction on the dimension. By contrast, the decidability proof from Karp and Miller tree [KM69] requires more work and does not provide the optimal upper bound as far as worst-case complexity is concerned.

Let $\mathcal{T}$ be a VAS of dimension n, and $\vec{x}$ and $\vec{x'}$ be two configurations in $\mathbb{N}^n$. We will show that if there is a run from $\vec{x}$ leading to $\vec{y}$ such that $\vec{x'} \preceq \vec{y}$, then there is a *small* run from $\vec{x}$ leading to $\vec{y'}$ such that $\vec{x'} \preceq \vec{y'}$ and its length is at most double-exponential in the size of the instance $\mathcal{T}$, $\vec{x}$ and $\vec{x'}$. Of course, we need to specify what a size means; no surprise is expected here since this will be a reasonably succinct encoding with a binary representation for integers. For instance, consider the VASS in Figure 3.4. There are various options to cover $(A, (1, K))$ from $(A, (0, 0))$ for some $K \geq 0$. For instance, here is a first covering

$$(A, (0, 0)) \xrightarrow{(t_1 t_2)^{2^K} t_1} (B, (0, 2^K + 1)) \xrightarrow{t_3 t_4 t_2} (A, (1, 2^K)) \succeq (A, (1, K))$$

Nevertheless, the covering below is obviously much shorter:

$$(A, (0, 0)) \xrightarrow{(t_1 t_2)^{K} t_1} (B, (0, K + 1)) \xrightarrow{t_3 t_4 t_2} (A, (1, K)) \succeq (A, (1, K))$$

The idea of the proof by Rackoff [Rac78] is to shorten systematically long coverings by using an induction on the dimension.

Still, we need to establish that the small run property entails the exponential space upper bound announced earlier, which is not immediate since a run of double-exponential length requires double-exponential space to be fully encoded and *a priori* there is a triple-exponential amount of such runs. It is the place where we use Savitch's theorem [Sav70]. Indeed, one can

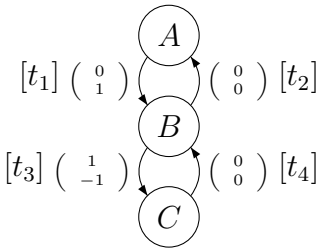


Figure 3.4: A simple VASS

design a decision procedure that nondeterministically guesses the small run (if it exists) and that requires only exponential space. Here is the principle of the nondeterministic algorithm with inputs $\mathcal{T}$, $\vec{x}$, $\vec{x'}$ and $L \geq 0$ (bound on the length of the witness run):

1. $i := 0$; $\vec{x_c} := \vec{x}$ (current configuration);
2. While $\vec{x'} \not\preceq \vec{x_c}$ and $i < L$ do
   (a) Guess a transition $t \in \mathcal{T}$;
   (b) If $\vec{x_c} + t \notin \mathbb{N}^n$ then abort;
   (c) $i := i + 1$; $\vec{x_c} := \vec{x_c} + t$.
3. If $\vec{x'} \preceq \vec{x_c}$ then accept else abort (i.e., $i = L$).

Observe that if the maximal absolute value in $\mathcal{T}$, $\vec{x}$, $\vec{x'}$ is 2^N where N is intended to be the size of the instance of the covering problem and $L = 2^{2^{N^3}}$, then the maximal absolute value appearing in the algorithm is $2^N + 2^N \times 2^{2^{N^3}}$ (it can be encoded with exponential space in N). Moreover, determinism can be regained with recursive calls to a function $F(\mathcal{T}, \vec{x}, \vec{x'}, L)$ since the number of transitions is finite (and will be bounded by N). Basically, $F(\mathcal{T}, \vec{x}, \vec{x'}, L)$ returns true whenever $\vec{x'}$ can be covered from $\vec{x}$ by a run with at most L transitions.

Therefore, exponential space shall be obtained for the following reasons:

- ⋆ a counter bounded by a double-exponential value (for the maximal length of the small run) needs only an exponential amount of bits,
- ⋆ we only need to store in memory two successive configurations (since we use nondeterminism),
- ⋆ given a size N, $2^{2^{N^3}} \times 2^N$ is still of double-exponential magnitude and can be encoded with an exponential amount of bits (so each configuration requires at most an exponential amount of bits),

- ⋆ checking the ordering between two natural numbers can be done in logarithmic space in their size (useful for the final check $\vec{x'} \preceq \vec{y}$ and for verifying that no component has negative value),
- ⋆ similarly, adding two natural numbers can be done in logarithmic space in their size (useful for going one step further).

Now, Savitch's theorem states that given a nondeterministic procedure for a given problem using space $f(N)$ greater than $log(N)$, there exists a deterministic procedure solving the same problem using $f(N) \times f(N)$ space [Sav70]. Since exponential functions are closed under multiplication, we obtain that checking whether an instance of the covering problem for VAS admits a small run can be done in exponential space in its size.

It is time to fix a few definitions. Let us start by defining the size of some VAS $\mathcal{T}$ of dimension $n \geq 1$. Given $\vec{x} \in \mathbb{Z}^n$, we write $\mathrm{maxneg}(\vec{x})$ [resp. $\max(\vec{x})$] to denote $\max(\{\max(0, -\vec{x}(i)) : i \in [1, n]\})$ [resp. $\max(\{\vec{x}(i) : i \in [1, n]\})$]. For example, $\mathrm{maxneg}(\begin{pmatrix} -1 \\ -2 \\ -8 \\ 7 \end{pmatrix}) = \max(0, -(-8)) = 8$ and $\max(\begin{pmatrix} -1 \\ -2 \\ -8 \\ 7 \end{pmatrix}) = 7$. By extension, we write $\mathrm{maxneg}(\mathcal{T})$ to denote $\max\{\mathrm{maxneg}(t) : t \in \mathcal{T}\}$. Furthermore, we write $\mathrm{scale}(\mathcal{T})$ to denote the value

$$\max(\{|t(i)| : t \in \mathcal{T},\ i \in [1, n]\}).$$

For instance, $\mathrm{scale}(\begin{pmatrix} -1 \\ -2 \\ -8 \\ 7 \end{pmatrix}) = |-8| = 8$. More generally, we write $\mathrm{scale}(X)$ to denote $\max(\{|\vec{y}(i)| : \vec{y} \in X,\ i \in [1, n]\})$ when X is a finite subset of $\mathbb{Z}^n$. The size of $\mathcal{T}$, written $|\mathcal{T}|$, is defined by the value below:

$$n \times \mathrm{card}(\mathcal{T}) \times (2 + \lceil log_2(1 + \mathrm{scale}(\mathcal{T})) \rceil)$$

Observe that $2 + \lceil log_2(1 + K) \rceil$ is a sufficient number of bits to encode integers in $[-K, K]$ for $K > 0$. Given a finite subset X of $\mathbb{Z}^n$, we also write $|X|$ to denote

$$n \times \mathrm{card}(X) \times (2 + \lceil log_2(1 + \mathrm{scale}(X)) \rceil)$$

In the sequel, given an instance $\mathcal{T}$, $\vec{x}, \vec{x'} \in \mathbb{N}^n$ of the covering problem we define its size by $N = |\mathcal{T}| + |\{\vec{x}\}| + |\{\vec{x'}\}|$. Observe that $\mathrm{maxneg}(\mathcal{T}), \mathrm{card}(\mathcal{T}), \max(\vec{x'}) \leq 2^N$.

A *path* is a finite sequence of transitions. A path π' is a *subpath* of the path $\pi = t_1 \dots t_k \overset{\text{def}}{\Leftrightarrow}$ there are $1 \leq j_1 < j_2 \cdots < j_{k'} \leq k$ such that $\pi' = t_{j_1} \dots t_{j_{k'}}$.

Whereas a configuration for $\mathcal{T}$ is an element of $\mathbb{N}^n$, a *pseudo-configuration* is defined as an element of $\mathbb{Z}^n$. When $\pi = t_1 \dots t_k$ is a path, the *pseudo-run* $(\pi, \vec{x})$ is the sequence of pseudo-configurations $\vec{x}_0 \cdots \vec{x}_k$ such that $\vec{x}_0 = \vec{x}$ and for $i \in [1, k]$, $\vec{x}_i = \vec{x}_{i-1} + t_i$. The pseudo-run $\vec{x}_0 \cdots \vec{x}_k$ is *induced* by the path π and of *length* $k + 1$; the path π is of *length* k. $\vec{x}_0$ is called the *initial* pseudo-configuration and $\vec{x}_k$ is called the *final* pseudo-configuration in the pseudo-run $\vec{x}_0 \cdots \vec{x}_k$. A pseudo-run $\vec{x}_0 \cdots \vec{x}_k$ is a covering of $\vec{x'}$ when $\vec{x'} \preceq \vec{x}_k$.

Let π be a path and $\vec{x}$ be a configuration such that $(\pi, \vec{x})$ is a run covering $\vec{x'}$. We write $m(\mathcal{T}, \vec{x}, \vec{x'}, \pi)$ to denote the length of the shortest subpath π' of π such that $(\pi', \vec{x})$ is also a run covering $\vec{x'}$. Obviously, $m(\mathcal{T}, \vec{x}, \vec{x'}, \pi)$ is less or equal to the length of π.

Let $M_B(n)$ be the supremum of the set below ($B, n \geq 1$):

$$\Big\{ m(\mathcal{T}, \vec{y}, \vec{y'}, \pi) \ : \ (\pi, \vec{y}) \text{ is a run covering } \vec{y'} \\ \mathcal{T} \text{ is a VAS of dimension } n \text{ and } \mathrm{maxneg}(\mathcal{T}) + \max(\vec{y'}) \leq B \Big\}$$

We show below that $M_B(n)$ is finite. More precisely, let us show that $M_B(n) \leq g_B(n)$ for all $n, B \geq 1$ with

$$g_B(n) = \begin{cases} B & \text{if } n = 1, \\ \big(B \cdot g_B(n-1)\big)^n + g_B(n-1) & \text{if } n \geq 2. \end{cases}$$

Lemma 3.4.1. For $n \geq 1$ and $B \geq 2$, $g_B(n) \leq B^{3n!}$.

Proof: The proof is by induction on the dimension n.The base case $n = 1$ is by an easy verification and $g_B(1) \leq B \leq B^6$. Now suppose that the property holds true for $n-1$ with $n-1 \geq 1$. Then, we have

$$g_B(n) \leq \big(B \cdot g_B(n-1)\big)^n + g_B(n-1) \leq \big(B \cdot g_B(n-1)\big)^{n+1} \leq \dots \\ \dots \leq \big(B^{1+(3(n-1))!}\big)^{n+1} \leq B^{(3n)!}$$

QED

Lemma 3.4.2. We have the following inequalities for $n, B \geq 1$:

$$M_B(n) \leq \begin{cases} B & \text{if } n = 1, \\ \big(B \cdot g_B(n-1)\big)^n + g_B(n-1) & \text{if } n \geq 2. \end{cases}$$

Consequently, $M_B(n) \leq g_B(n)$ for all $n, B \geq 1$.

Proof: Let us treat the base case with $n = 1$. Consider the instance $\mathcal{T}$, $\vec{x}$ and $\vec{x'}$ with $\mathcal{T}$ of dimension 1. In order to cover $\vec{x'}$, there is no need to use negative values from $\mathcal{T}$ and therefore $m(\mathcal{T}, \vec{x}, \vec{x'}, \pi)$ is bounded by $\max(\vec{x'})$ for any path π, which provides a bound B to $M_B(1)$.

Let us treat the induction step and suppose the property holds true for $n - 1 \geq 1$. Let us show that

$$m(\mathcal{T}, \vec{x}, \vec{x'}, \pi) \leq \big(B \cdot g_B(n-1)\big)^n + g_B(n-1)$$

whenever $\mathrm{maxneg}(\mathcal{T}) + \max(\vec{x'}) \leq B$.

First observe that in a pseudo-run, at each single step, a component may decrement by at most $\mathrm{maxneg}(\mathcal{T})$. So, if at some stage a component has value greater than $g_B(n-1)\mathrm{maxneg}(\mathcal{T}) + \max(\vec{x'})$ then after $g_B(n-1)$ steps, that component has a value greater or equal to $\max(\vec{x'})$. We pose $B' = g_B(n-1)\mathrm{maxneg}(\mathcal{T}) + \max(\vec{x'}) \leq B g_B(n-1)$.

A pseudo-run $\vec{x}_0 \cdots \vec{x}_k$ is said to be *r-bounded* for some $r > 0$ when for $i \in [0, k]$, we have $\vec{x}_i \in [0, r-1]^n$. Let π be a path and $\vec{x}$ be a configuration such that $(\pi, \vec{x})$ is a run covering $\vec{x'}$ for the VAS $\mathcal{T}$. Suppose $\pi = t_1 \cdots t_k$ and $(\pi, \vec{x}) = \vec{x}_0 \cdots \vec{x}_k$.

Case 1: $(\pi, \vec{x})$ is B'-bounded.

If there are $0 \leq i < j \leq k$ such that $\vec{x}_i = \vec{x}_j$, then $(\pi', \vec{x})$ is also a run covering $\vec{x'}$ where

$\pi' = t_1 \cdots t_i t_{j+1} \cdots t_k$. Observe that π' is a subpath of π. This situation occurs necessarily when $k \geq (B')^n$ and we can repeat the above transformation (pigeonhole principle). Consequently, when $(\pi, \vec{x})$ is B'-bounded, there is a subpath π' such that $(\pi', \vec{x})$ is also a run covering $\vec{x'}$ and its length is bounded by $(B')^n$, that is by $(Bg_B(n-1))^n$.
Case 2: $(\pi, \vec{x})$ is not B'-bounded.
The path π can be uniquely divided into two paths π_1 and π_2 of respective length k_1 and k_2 ($k = k_1 + k_2$) such that the only values in $\vec{x}_0 \cdots \vec{x}_{k_1}$ that are greater or equal to B' are in $\vec{x}_{k_1}$, $(\pi_1, \vec{x})$ is not B'-bounded and $\pi = \pi_1\pi_2$. So, π has unique decomposition $\pi = \pi_1\pi_2$ such that

- π_1 is of length k_1,
- all values in $\vec{x}_0 \cdots \vec{x}_{k_1-1}$ are strictly smaller than B'.
- $(\pi_1, \vec{x})$ is not B'-bounded ("faulty" configuration $\vec{x}_{k_1}$).

$$\underbrace{\vec{x_0} \xrightarrow{t_1} \cdots \xrightarrow{t_{k_1-1}} \vec{x_{k_1-1}}}_{B'-\text{bounded}} \xrightarrow{t_{k_1}} \vec{x_{k_1}} \notin [0, B'-1]^n \xrightarrow{t_{k_1+1}} \vec{x_{k_1+1}} \cdots \xrightarrow{t_K} \vec{x_K}$$

$$\pi_1 = t_1 \cdots t_{k_1} \quad \pi_2 = t_{k_1+1} \cdots t_K$$

By using a reasoning similar to the one in Case 1, there is a path π'_1, subpath of π_1, such that its length is bounded by $(Bg_B(n-1))^n + 1$ and, $(\pi_1, \vec{x})$ and $(\pi'_1, \vec{x})$ have the same final configuration, say $\vec{y} = \vec{x_{k_1}}$. Hence, $(\pi'_1\pi_2, \vec{x})$ and $(\pi_2, \vec{y})$ are also runs covering $\vec{x'}$. By construction of π_1, there is $i \in [1, n]$ such that $\vec{y}(i) \geq B'$. The run $(\pi_2, \vec{y})$ can be illustrated as follows:

$$\vec{x_{k_1}} = \begin{pmatrix} \vdots \\ \cdot \geq B' \\ \vdots \end{pmatrix} \xrightarrow{t_{k_1+1}} \cdots \xrightarrow{t_K} \vec{x_K} = \begin{pmatrix} \vdots \\ \cdot \\ \vdots \end{pmatrix} \succeq \vec{x'}$$

Let $\mathcal{T}^-$, π_2^-, $\vec{y}^-$ and $\vec{x'}^-$ be the respective restrictions of $\mathcal{T}$, π_2, $\vec{y}$ and $\vec{x'}$ to the components in $[1, n] \setminus \{i\}$. Similar notations are used for any element of $\mathbb{Z}^n$ or for any subset of $\mathbb{Z}^n$. $(\pi_2^-, \vec{y}^-)$ is a run covering $\vec{x'}^-$ in $\mathcal{T}^-$ as illustrated below:

$$\vec{x_{k_1}}^- = \begin{pmatrix} \vdots \\ \vdots \end{pmatrix} \xrightarrow{t^-_{k_1+1}} \cdots \xrightarrow{t^-_K} \vec{x_K} = \begin{pmatrix} \vdots \\ \vdots \end{pmatrix} \succeq \vec{x'}^-$$

Observe that $\text{maxneg}(\mathcal{T}^-) + \max(\vec{x'}^-)$ is also less than B and therefore we can apply the induction hypothesis. By the induction hypothesis, there is a path π'_2, subpath of π_2^- such that $(\pi'_2, \vec{y}^-)$ is a run covering $\vec{x'}^-$ and its length is less than $g_B(n-1)$. From π'_2, we can obtain a path π''_2 for $\mathcal{T}$, such that $(\pi''_2, \vec{y})$ is a pseudo-run with final pseudo-configuration $\vec{z}$ such that for $j \in ([1, n] \setminus \{i\})$, $\vec{z}(j) \geq \vec{x'}(j)$. The path π''_2 is obtained from π'_2 by adding the ith missing component. However since $\vec{y}(i) \geq B'$ and after $g_B(n-1)$ steps, the ith component is greater or equal to $\max(\vec{x'})$, $(\pi''_2, \vec{y})$ is a run covering $\vec{x'}$. The length of the path $\pi'_1\pi''_2$ is at most $(B \times g_B(n-1))^n + g_B(n-1)$ and $\pi'_1\pi''_2$ is a subpath of π. QED

Corollary 3.4.3. Let $\mathcal{T}$, $\vec{x}$ and $\vec{x'}$ be an instance of the covering problem. There is a run from $\vec{x}$ leading to $\vec{y}$ such that $\vec{x'} \preceq \vec{y}$ iff there is a run from $\vec{x}$ leading to $\vec{y'}$ such that $\vec{x'} \preceq \vec{y'}$ and its length is bounded by $(\mathrm{maxneg}(\mathcal{T}) + \max(\vec{x'}) + 2)^{(3n)!}$.

Let us pose $N = |\mathcal{T}| + |\{\vec{x}\}| + |\{\vec{x'}\}|$. So $N \geq 3$, $\mathrm{maxneg}(\mathcal{T}) \leq 2^N$, $\max(\vec{x'}) \leq 2^N$ and $n \leq N$. We obtain the following inequalities (with rough simplifications)

$$(\mathrm{maxneg}(\mathcal{T}) + \max(\vec{x'}) + 2)^{(3n)!} \leq (2^N + 2^N + 2)^{2^{Nlog_2(N)}} \leq (2^{N+2})^{2^{N^2}} \leq 2^{2^{N^3}}.$$

Theorem 3.4.4. [Rac78] The covering problem for VAS, VASS and Petri nets can be solved in exponential space.

Corollary 3.4.5. For any fixed $n \geq 1$, the covering problem restricted to VAS of dimension at most n can be solved in polynomial space.

Boundedness problem for VASS has also been shown in EXPSPACE in [Rac78] and a generalization of the proof technique has been shown in [AH09] (we cannot exactly rely on [Yen92, Theorem 3.8] for decidability since [Yen92, Lemma 3.7] contains a flaw, as observed in [AH09]).

3.5 Further reading

- ⋆ A very well-presented proof of EXPSPACE-hardness of covering, boundedness and reachability problems for VAS can be found in [Esp98] (based on [Lip76], see also in [CLM76] how the lower bound is preserved for reversible Petri nets). EXPSPACE-hardness is obtained by reduction from the halting problem for counter automata when counters are bounded by 2^{2^n}. A counter C in the counter automaton is represented by two components i_{C} and $\overline{i_{\mathsf{C}}}$. Whenever a configuration $\vec{x}$ is reached, we require the invariant $\vec{x}(i_{\mathsf{C}}) + \vec{x}(\overline{i_{\mathsf{C}}}) = 2^{2^n}$. Increments and decrements C are easy to simulate while preserving the equality $\vec{x}(i_{\mathsf{C}}) + \vec{x}(\overline{i_{\mathsf{C}}}) = 2^{2^n}$. The simulation of zero-test on C is more delicate: one should be able to decrement $\vec{x}(\overline{i_{\mathsf{C}}})$ by 2^{2^n}. In order to perform this large decrement, auxiliary components with values 2^{2^α} with $\alpha \leq n$ are needed and initialized by using concentric loops with

$$2^{2^{\alpha-1}} \times 2^{2^{\alpha-1}} = 2^{2^\alpha}$$

- ⋆ Complexity and decidability issues for Petri nets are considered in [Had01] (in French but an English version exists also).
- ⋆ Useful references about the decidability of the reachability problem for VAS include the following: [Kos82, May84, Reu90, Lam92, Had01, Ler09, Ler11]. Relationships between semilinear sets and reachability sets in VAS can be found in [HP79], [Mog01, Theorem 9] and [Ler09].
- ⋆ Computations with VAS are words and systems with tree-like computations have been introduced, extending what exists for VAS, and leading to the model of branching vector addition systems (BVAS). In recent years, it has turned out that BVAS have interesting connections to a number of formalisms:

- BVAS correspond to a class of linear index grammars in computational linguistics, see e.g. an up-to-date presentation in [Sch10a].
- Reachability problem for BVAS is decidable iff provability in multiplicative exponential linear logic (MELL) is decidable [dGGS04].
- Verma and Goubault-Larrecq [VGL05] have extended the computation of Karp and Miller trees to BVAS, and used it to draw conclusions about a class of equational tree automata which are useful for analysing cryptographic protocols.

Covering and boundedness for BVAS are decidable using a branching extension of Karp and Miller's procedure [VGL05] and the complexity has been recently characterized in [DJLL09]. Moreover, the current presentation of the proof from [Rac78] for the EXPSPACE upper bound for the covering problem for VASS follows [DJLL09]. Nevertheless, BVAS model in full generality is not so well understood. For instance, the decidability status of the reachability problem is open whereas it has been recently shown 2EXPSPACE-hard [Laz10]. Moreover, we ignore which standard restrictions existing for VAS could be relevantly adapted to BVAS in order to weaken the computational complexity of various problems. A serious candidate is to adapt developments from [PL09] for BVAS. Finally, observe that a stronger model of branching VASS has been considered in [LMSS92, Urq99].

3.6 Exercises

Exercise 3.6.1. Complete the end of the proof of Theorem 3.1.2.

Exercise 3.6.2. Show that satisfiability problem for $\mathrm{FO}(\sim, <, +1)$ restricted to three individual variables is undecidable. Answer can be found in [BMS$^+$06, Dav09].

Exercise 3.6.3. Answer the following questions for the Petri net presented in Figure 3.2.

- $\star$ Is $(1000, 0, 0, 1)$ reachable from $(0, 1, 0, 0)$ (with implicit ordering of the places p_A, p_B, p_C, p_1) ?
- $\star$ Is $(2, 1, 0, 1)$ reachable from $(0, 1, 0, 0)$?
- $\star$ Is the Petri net with initial marking $(0, 1, 0, 0)$ bounded?
- $\star$ Is there some marking m reachable from $(1, 0, 0, 0)$ such that $(1000, 1, 0, 0) \preceq m$?

Exercise 3.6.4. Show that the simulations in the proof of Lemma 3.2.1 allow us to obtain equivalence for reachability, boundedness and covering problems.

Exercise 3.6.5. By using the proof of Lemma 3.2.2, explain how to reduce the control state reachability problem for VASS to the covering problem for VAS.

Exercise 3.6.6. Show that the construction of coverability graphs terminates.

Exercise 3.6.7. Show Lemma 3.3.1(III).

Exercise 3.6.8. Show that the covering problem restricted to VAS of dimension 1 can be solved in linear time.

Exercise 3.6.9. Show that the covering problem for VASS augmented the multiplication by 2 can be solved in exponential space.

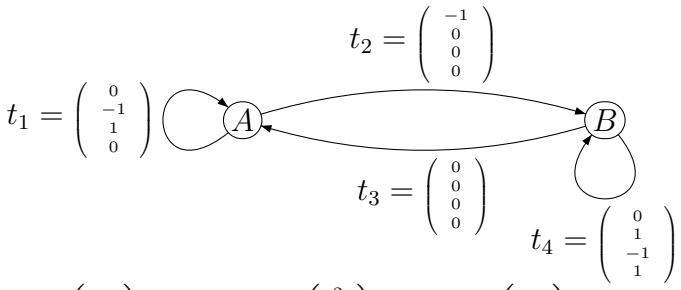
Exercise 3.6.10. Define a reduction from the covering problem for counter systems to existential model-checking problem for $\mathrm{LTL}^{\mathrm{CS}}(\mathrm{PrA})$.

Exercise 3.6.11. We have seen that $(\mathcal{T}, \vec{x_0})$ is unbounded iff there is a run of the form $\vec{x_0} \xrightarrow{*} \vec{y} \xrightarrow{*} \vec{y'}$ with $\vec{y} \prec \vec{y'}$. Assuming that $\mathcal{T}$ has dimension n, we say that $(\mathcal{T}, \vec{x_0})$ is i-unbounded ($i \in [1, n]$) $\overset{\text{def}}{\Leftrightarrow}$ $\{\vec{y}(i) : \vec{x_0} \xrightarrow{*} \vec{y}\}$ is infinite. Hence, $(\mathcal{T}, \vec{x_0})$ is unbounded iff there is some $i \in [1, n]$ such that $(\mathcal{T}, \vec{x_0})$ is i-unbounded. Are the propositions below equivalent?

1. $(\mathcal{T}, \vec{x_0})$ is i-unbounded.
2. There is a run of the form $\vec{x_0} \xrightarrow{*} \vec{y} \xrightarrow{*} \vec{y'}$ with $\vec{y} \preceq \vec{y'}$ and $\vec{y}(i) < \vec{y'}(i)$.

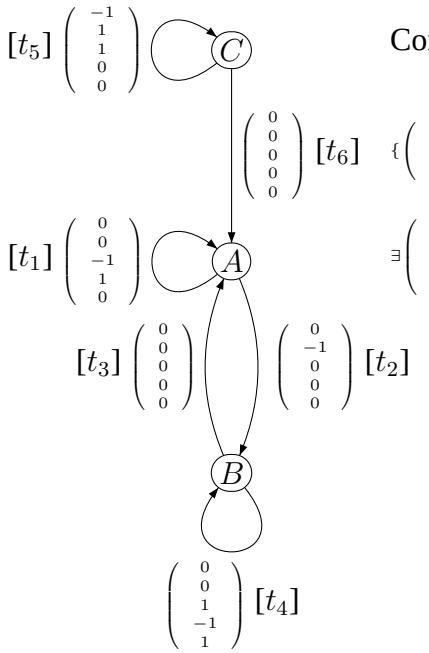
Exercise 3.6.12. Define a polynomial-time reduction from the covering problem for VASS to the reachability problem for VASS.

Exercise 3.6.13.

1. 

   Compute $\{\begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} \in \mathbb{N}^4 : (A, \begin{pmatrix} 2 \\ 1 \\ 0 \\ 0 \end{pmatrix}) \xrightarrow{*} (A, \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix})\}$.
2. Show that
$$\{\begin{pmatrix} a \\ b \\ d \end{pmatrix} \in \mathbb{N}^3 : d \leq a \times b\} = \{\begin{pmatrix} a \\ b \\ d \end{pmatrix} \in \mathbb{N}^3 : \exists \begin{pmatrix} a' \\ b' \\ c' \end{pmatrix} \in \mathbb{N}^3, (A, \begin{pmatrix} a \\ b \\ 0 \\ 0 \end{pmatrix}) \xrightarrow{*} (A, \begin{pmatrix} a' \\ b' \\ c' \\ d \end{pmatrix})\}$$
3. Is there $\begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} \in \mathbb{N}^4$ such that $\{\begin{pmatrix} a' \\ b' \\ c' \\ d' \end{pmatrix} \in \mathbb{N}^4 : (A, \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix}) \xrightarrow{*} (A, \begin{pmatrix} a' \\ b' \\ c' \\ d' \end{pmatrix})\}$ is infinite?

4.



Compute the map f such that

$$\{\begin{pmatrix}a\\b\\e\end{pmatrix}\in\mathbb{N}^3 : e\leq f(a,b)\}=\{\begin{pmatrix}a\\b\\e\end{pmatrix}\in\mathbb{N}^3 : \exists\begin{pmatrix}a'\\b'\\c'\\d'\end{pmatrix}\in\mathbb{N}^4, (C,\begin{pmatrix}a\\0\\b\\0\\0\end{pmatrix})\xrightarrow{*}(A,\begin{pmatrix}a'\\b'\\c'\\d'\\e\end{pmatrix})\}$$

Compute the map f such that

$$\{\begin{pmatrix}a\\b\\e\end{pmatrix}\in\mathbb{N}^3 : e\leq f(a,b)\}=\{\begin{pmatrix}a\\b\\e\end{pmatrix}\in\mathbb{N}^3 : exists\begin{pmatrix}a'\\b'\\c'\\d'\end{pmatrix}\in\mathbb{N}^4, (C,\begin{pmatrix}a\\0\\b\\0\\0\end{pmatrix})\xrightarrow{*}(A,\begin{pmatrix}a'\\b'\\c'\\d'\\e\end{pmatrix})\}$$

Exercise 3.6.14. Below, we assume that we have a terminating procedure such that given a VASS $\mathcal{V}$, and two configurations $(q,\vec{x})$ and $(q',\vec{x'})$, returns 'yes' iff there is a run from $(q,\vec{x})$ to $(q',\vec{x'})$ respecting the transitions from $\mathcal{V}$.

1. Let us consider a first model that extends VASS by allowing transitions of the form

$$t = q \xrightarrow{\geq\vec{b}} q' \quad \text{with } \vec{b}\in\mathbb{N}^n$$

such that $(q,\vec{a})\xrightarrow{t}(q',\vec{a'})$ iff $\vec{b}\preceq\vec{a}$ and $\vec{a}=\vec{a'}$. As usual, $\vec{b}\preceq\vec{a}\overset{\text{def}}{\Leftrightarrow}$ for $i\in[1,n]$, we have $\vec{b}(i)\leq\vec{a}(i)$. Show that the covering problem for this extended class of VASS can be solved in exponential space.

2. Let us consider another model that extends VASS by allowing transitions of the form

$$t = q \xrightarrow{\mathtt{x}_i\leq k} q' \quad \text{with } i\in[1,n],\ k\in\mathbb{N}$$

such that $(q,\vec{a})\xrightarrow{t}(q',\vec{a'})$ iff $\vec{a}(i)\leq k$ and $\vec{a}=\vec{a'}$. Show that the covering problem for this extended class of VASS is undecidable.

3. Let us consider a third model that extends VASS by allowing transitions of the form

$$t = q \xrightarrow{\leq \vec{b}} q' \quad \text{with } \vec{b} \in \mathbb{N}^n$$

such that $(q, \vec{a}) \xrightarrow{t} (q', \vec{a'})$ iff $\vec{a} \preceq \vec{b}$ and $\vec{a} = \vec{a'}$. Define a polynomial-time reduction from the reachability problem for VASS into the covering problem for this extended class of VASS. Comment this result.

4. Let $\mathcal{V}$ be an extended VASS of dimension n (with set of locations Q) such that the extended transitions are exactly

$$q_1 \xrightarrow{\leq \vec{b_1}} q'_1, \ \ldots, \ q_N \xrightarrow{\leq \vec{b_N}} q'_N$$

with $\vec{b_1}, \ldots, \vec{b_N} \in \mathbb{N}^n$. Show that if there is a run from $(q, \vec{x})$ to $(q', \vec{x'})$, then there is a run such that the number of times extended transitions are fired is at most exponential in the size of $\mathcal{V}$. Provide a precise upper bound.

5. Given an initial configuration $(q, \vec{x})$, design an algorithm that computes the set below:

$$\{(q_i, \vec{a}) \in Q \times \mathbb{N}^n : i \in [1, N], \ \vec{a} \preceq \vec{b_i}, \ (q, \vec{x}) \xrightarrow{*} (q_i, \vec{a}) \text{ in } \mathcal{V}\}$$

6. Conclude that the reachability problem and the covering problem for this extended class of VASS are decidable.

Exercise 3.6.15. (another variant) Let us consider an extension of VASS by allowing extended transitions of the form $t = q \xrightarrow{=\vec{b}} q' \quad \text{with } \vec{b} \in \mathbb{N}^n$ such that $(q, \vec{a}) \xrightarrow{t} (q', \vec{a'})$ iff $\vec{a} = \vec{b}$ (equality test) and $\vec{a'} = \vec{a}$ (update is identity).

Question 3.6.15.1 Let $\mathcal{V} = (Q, n, \delta)$ be an extended VASS such that the extended transitions of $\mathcal{V}$ are exactly those below (apart from the standard transitions):

$$\mathbf{q_1} \xrightarrow{=\vec{b_1}} \mathbf{q'_1}, \ \ldots, \ \mathbf{q_N} \xrightarrow{=\vec{b_N}} \mathbf{q'_N}$$

Show that if there is a run from $(q, \vec{x})$ to $(q', \vec{x'})$, then there is a run from $(q, \vec{x})$ to $(q', \vec{x'})$ such that the number of times extended transitions are fired is at most N.

Question 3.6.15.2 Given an initial configuration $(q, \vec{a})$, design an algorithm that computes the set below:

$$\{(\mathbf{q_i}, \vec{b_i}) : i \in [1, N], (q, \vec{a}) \xrightarrow{*} (\mathbf{q_i}, \vec{b_i}) \text{ in } \mathcal{V}\}$$

Hint: use as a subroutine an algorithm for solving the reachability problem for VASS (taken for granted).

Question 3.6.15.3 Conclude that the reachability problem for this class of extended VASS is decidable.

Chapter 4

Reversal-Bounded Counter Automata

In this chapter, we present and study the class of reversal-bounded counter automata introduced in [Iba78] and we present several decision problems based on temporal logics. This class of counter systems, as well as slight extensions, are known to admit Presburger-definable accessibility relations and one can effectively build the Presburger formulae. The reachability sets defined by reversal-bounded (initialized) counter automata are effectively Presburger-definable [Iba78] and this is the main result presented in this chapter. In order to prove it, we shall use the fact that the Parikh images of context-free languages are effectively semilinear. These results extend to weak reversal-boundedness, a relaxed version of reversal-boundedness introduced in [FS08]. This chapter also deals with linear properties such as control state repeated reachability and infinite repetition of Presburger properties. Borderlines of undecidability are discussed leading to various temporal fragments with decidable model-checking problems.

The content of this chapter is the following.

- ⋆ We present the class of reversal-bounded counter automata for which reachability sets can be shown effectively semilinear (i.e., definable in Presburger arithmetic).
- ⋆ We show how semilinearity can be preserved for various extensions (weak reversal-boundedness, addition of a free counter).
- ⋆ Although the reachability problem for reversal-bounded (initialized) counter automata is decidable by effective semilinearity, we consider linear-time properties on reversal-bounded counter automata and present decidable problems as well as undecidable ones, providing rough borders for decidability.

4.1 What is reversal-boundedness?

A *reversal* for a counter occurs in a run when there is an alternation from nonincreasing mode to nondecreasing mode and vice-versa. For instance, in the sequence below, there are three reversals identified by an upper line:

$$00112233344443\overline{3}33222\overline{3}3344444555554\overline{4}$$

Similarly, the sequence 00111222223333334444 has no reversal. Figure 4.1 presents schematically the behavior of a counter with 5 reversals. A counter automaton is *reversal-bounded* when-

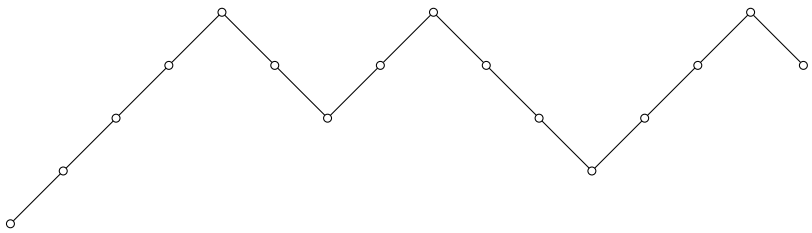


Figure 4.1: 5 reversals in a row

ever there is $r \geq 0$ such that for any run from a given initial configuration, every counter makes no more than r reversals. This class of counter automata has been introduced and studied in [Iba78], partly inspired by similar restrictions on multistack automata [BB74]. A formal definition will follow, but before going any further, it is worth pointing out a few peculiarities of this subclass. Indeed, reversal-boundedness is defined for initialized counter automata (a counter automaton augmented with an initial configuration) and the bound r depends on the initial configuration. Secondly, this class is not defined from the class of counter automata by imposing syntactic restrictions but rather semantically. In spite of the fact that the problem of deciding whether a counter automaton is reversal-bounded is undecidable [Iba78] (see Theorem 4.1.1), we shall see that reversal-bounded counter automata have numerous fundamental properties.

Let $\mathcal{S} = (Q, n, \delta)$ be a standard counter automaton. Let us define the auxiliary (succinct) counter automaton $\mathcal{S}_{rb} = (Q', 2n, \delta')$ such that $Q' = Q \times \{\mathrm{DEC}, \mathrm{INC}\}^n$ and $(q, \vec{mode}) \xrightarrow{\varphi'} (q', \vec{mode}') \in \delta' \overset{\text{def}}{\Leftrightarrow}$ there is $q \xrightarrow{\varphi} q' \in \delta$ such that if φ does not deal with the jth component, then $\vec{mode}(j) = \vec{mode}'(j)$ and for every $i \in [1, n]$, one of the conditions below is satisfied:

- $\varphi = \mathrm{zero}(i)$, $\vec{mode}(i) = \vec{mode}'(i)$, $\varphi' = \varphi \wedge \bigwedge_{j \in [1,n]} \mathsf{x}'_{n+j} = \mathsf{x}_{n+j}$,
- $\varphi = \mathrm{dec}(i)$, $\vec{mode}(i) = \vec{mode}'(i) = \mathrm{DEC}$ and $\varphi' = \varphi \wedge \bigwedge_{j \in [1,n]} \mathsf{x}'_{n+j} = \mathsf{x}_{n+j}$,
- $\varphi = \mathrm{dec}(i)$, $\vec{mode}(i) = \mathrm{INC}$, $\vec{mode}'(i) = \mathrm{DEC}$ and

$$\varphi' = \varphi \wedge (\mathsf{x}'_{n+i} = \mathsf{x}_{n+i} + 1) \wedge \bigwedge_{j \in [1,n] \setminus \{i\}} \mathsf{x}'_{n+j} = \mathsf{x}_{n+j},$$

- $\varphi = \mathrm{inc}(i)$, $\vec{mode}(i) = \vec{mode}'(i) = \mathrm{INC}$ and $\varphi' = \varphi \wedge \bigwedge_{j \in [1,n]} \mathsf{x}'_{n+j} = \mathsf{x}_{n+j}$,
- $\varphi = \mathrm{inc}(i)$, $\vec{mode}(i) = \mathrm{DEC}$, $\vec{mode}'(i) = \mathrm{INC}$ and

$$\varphi' = \varphi \wedge (\mathsf{x}'_{n+i} = \mathsf{x}_{n+i} + 1) \wedge \bigwedge_{j \in [1,n] \setminus \{i\}} \mathsf{x}'_{n+j} = \mathsf{x}_{n+j}.$$

Essentially, the n new components in $\mathcal{S}_{rb}$ count the number of reversals for each component from $\mathcal{S}$. Observe that $\mathcal{S}_{rb}$ is succinct because two counters may be updated in one step. However, it is easy to turn $\mathcal{S}_{rb}$ into a standard counter automaton by adding intermediate control states.

Initialized counter automaton $(\mathcal{S}, (q, \vec{x}))$ is *reversal-bounded* [Iba78] $\overset{\text{def}}{\Leftrightarrow}$ for every $i \in [n+1, 2n]$, $\{\vec{y}(i) : \text{run } (q_{rb}, \vec{x}_{rb}) \xrightarrow{*} (q', \vec{y}) \text{ in } \mathcal{S}_{rb}\}$ is finite with $q_{rb} = (q, \vec{\mathrm{INC}})$, $\vec{x}_{rb}$ restricted to the

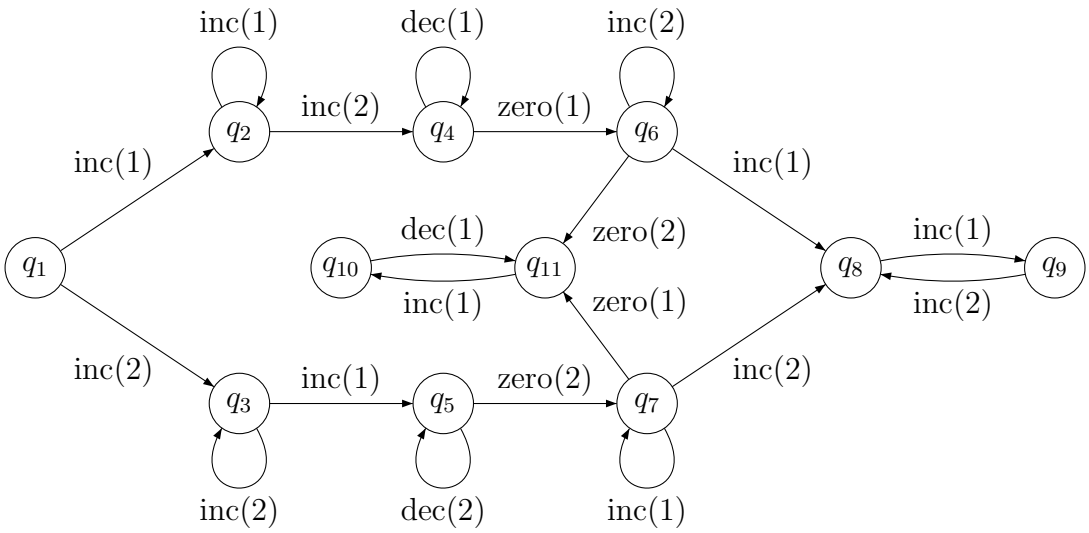


Figure 4.2: A counter automaton that bounds the numbers of reversals

n first components is $\vec{x}$ and $\vec{x}_{rb}$ restricted to the n last components is $\vec{0}$. When $r \geq \max(\{\vec{y}(i) : \text{run } (q_{rb}, \vec{x}_{rb}) \xrightarrow{*} (q', \vec{y}) \text{ in } \mathcal{S}_{rb}\} : i \in [n+1, 2n])$ $\mathcal{S}$ is said to be *r-reversal-bounded* from $(q, \vec{x})$. For a fixed $i \in [1, n]$, when $\{\vec{y}(n+i) : \text{ run } (q_{rb}, \vec{x}_{rb}) \xrightarrow{*} (q', \vec{y}) \text{ in } \mathcal{S}_{rb}\}$ is finite, we say that $(\mathcal{S}, (q, \vec{x}))$ is *reversal-bounded with respect to i*.

Figure 4.2 contains a counter automaton $\mathcal{S}$ such that any initialized counter automaton of the form $(\mathcal{S}, (q_1, \vec{x}))$ with $\vec{x} \in \mathbb{N}^2$ is reversal-bounded.

Since reversal-boundedness is not defined from counter automata by a syntactic criterion, the following problem makes sense and indeed it happens to be undecidable.

REVERSAL-BOUNDEDNESS DETECTION PROBLEM

Input: Initialized counter automaton $(\mathcal{S}, (q, \vec{x}))$ of dimension n and $i \in [1, n]$.

Question: Is $(\mathcal{S}, (q, \vec{x}))$ reversal-bounded with respect to the component i?

Theorem 4.1.1. [Iba78] Reversal-boundedness detection problem is undecidable.

Proof: Let us consider a slight (and indeed standard) variant of Minsky machines in which the last instruction n is `halt`. The halting problem checks whether the Minsky machine can reach this instruction. Since the Minsky machine is deterministic, either the Minsky machine has a unique infinite run (and never visits the instruction n) or it has a unique finite (and halts at instruction n).

Let us consider a Minsky machine $\mathcal{S}$ with the above-mentioned last instruction, which can be defined as a standard counter automaton. Let us build the counter automaton $\mathcal{S}'$ of dimension 1 obtained from $\mathcal{S}$ by replacing every transition $t = q_i \xrightarrow{\varphi} q_j$ (each instruction is attached to a dedicated control state) by $q_i \xrightarrow{\text{inc}(1)} q_{1,t}^{new} \xrightarrow{\text{dec}(1)} q_{2,t}^{new} \xrightarrow{\varphi} q_j$ where $q_{1,t}^{new}$ and $q_{2,t}^{new}$ are new control states associated to the transition t. We have the following equivalences, which allows us to get undecidability:

- ⋆ The Minsky machine $\mathcal{S}$ halts.

- $\star$ For the counter automaton $\mathcal{S}'$, the control state q_n can be reached from the initial configuration $(q_1, \vec{0})$.
- $\star$ The unique run of $\mathcal{S}'$ with initial configuration $(q_1, \vec{0})$ is finite.
- $\star$ $\mathcal{S}'$ is reversal-bounded with respect to component 1 from the initial configuration $(q_1, \vec{0})$.

QED

Even though reversal-boundedness detection problem is undecidable in full generality, it is shown in [FS08] that the problem is decidable for counter automata without zero-tests, and more generally for vector addition systems with states (by adapting in the obvious way the concept of reversal-boundedness to VASS). More recently, it has been shown that the reversal-boundedness detection problem restricted to VASS is EXPSPACE-complete [Dem10]. Furthermore, checking whether a counter automaton $\mathcal{S}$ is r-reversal-bounded is undecidable [Iba78, Theorem 4.1(b)] (the inputs are $\mathcal{S}$ and $r \geq 0$) whereas the problem is decidable for VASS, as a consequence of [FS08].

Reversal-boundedness for counter automata is very appealing because reachability sets are semilinear as stated below.

Theorem 4.1.2. [Iba78] Let $(\mathcal{S}, (q, \vec{x}))$ be an initialized counter automaton that is r-reversal-bounded for some $r \geq 0$. For each control state q', the set $\{\vec{y} \in \mathbb{N}^n : \exists \text{ run } (q, \vec{x}) \xrightarrow{*} (q', \vec{y})\}$ is effectively semilinear.

This means that one can compute effectively a Presburger formula that characterizes precisely the reachable configurations whose control state is q'. The original proof for reversal-boundedness can be found in [Iba78]. Main part of this chapter is dedicated to the proof of Theorem 4.1.2.

A counter automaton $\mathcal{S}$ is *uniformly reversal-bounded* iff there is $r \geq 0$ such that for every initial configuration, the initialized counter automaton is r-reversal-bounded. The question of checking whether a counter automaton $\mathcal{S}$ is uniformly reversal-bounded can be reduced to reversal-boundedness. Indeed, it is sufficient to introduce a new control state q_{new} that contains as many self-loops as the dimension n and each self-loop i increments the ith component. Then, nondeterministically we jump to the rest of the counter automaton with no effect on the counters. In this way, $(\mathcal{S}', (q_{new}, \vec{0}))$ is reversal-bounded ($\mathcal{S}'$ is the new counter automaton obtained as a variant of $\mathcal{S}$) iff $\mathcal{S}$ is uniformly reversal-bounded. As an exercise, one can check that the counter automaton in Figure 4.2 is not uniformly reversal-bounded.

Let us consider the following problem.

REACHABILITY PROBLEM WITH BOUNDED NUMBER OF REVERSALS:

Input: a counter automaton $\mathcal{S}$, a bound $r \in \mathbb{N}$, an initial configuration $(q_0, \vec{x_0})$ and a final configuration $(q, \vec{x})$,

Question: Is there a finite run of $\mathcal{S}$ with initial configuration $(q_0, \vec{x_0})$ and final configuration $(q, \vec{x})$ such that each counter has at most r reversals?

Observe that when $(\mathcal{S}, (q_0, \vec{x_0}))$ is r'-reversal-bounded for some $r' \leq r$, we get an instance of the reachability problem with initial configuration $(q_0, \vec{x_0})$.

Corollary 4.1.3. The reachability problem with bounded number of reversals is decidable.

Proof: Here is the decidability proof that uses Theorem 4.1.2. Let $\mathcal{S} = (Q, n, \delta)$, $r \in \mathbb{N}$, $(q_0, \vec{x_0})$ and $(q, \vec{x})$ be an instance of the reachability problem with bounded reversals. First, we build a counter automaton $\mathcal{S}' = (Q', n, \delta')$ with $Q' = Q \times \{\mathrm{DEC}, \mathrm{INC}\}^n \times [0, r]^n$.

By construction of $\mathcal{S}'$, we guarantee that $(\mathcal{S}', ((q_0, \vec{\mathrm{INC}}, \vec{0}), \vec{x_0}))$ is r-reversal-bounded. Indeed, for each counter, we shall count the number of reversals and by construction of $\mathcal{S}'$ we shall enforce that it is bounded by r on each run. The set of transitions δ' is defined as follows: $(q, \vec{mode}, \sharp\vec{alt}) \xrightarrow{\varphi} (q', \vec{mode}', \sharp\vec{alt}') \in \delta' \overset{\text{def}}{\Leftrightarrow} q \xrightarrow{\varphi} q' \in \delta$ and for $i \in [1, n]$, the relation described by the following table is verified. The values of two first columns induce values for the two last columns (when it is possible, see e.g. the condition $\sharp\vec{alt}(i) < r$).

φ	$\vec{mode}(i)$	$\vec{mode}'(i)$	$\sharp\vec{alt}'(i)$
$\mathrm{dec}(i)$	DEC	DEC	$\sharp\vec{alt}(i)$
$\mathrm{dec}(i)$	INC	DEC	$\sharp\vec{alt}(i) + 1$ and $\sharp\vec{alt}(i) < r$
$\mathrm{inc}(i)$	INC	INC	$\sharp\vec{alt}(i)$
$\mathrm{inc}(i)$	DEC	INC	$\sharp\vec{alt}(i) + 1$ and $\sharp\vec{alt}(i) < r$
$\mathrm{zero}(i)$	DEC	DEC	$\sharp\vec{alt}(i)$
$\mathrm{zero}(i)$	INC	INC	$\sharp\vec{alt}(i)$

By construction, $\mathcal{S}'$ is r-reversal bounded and the properties below are equivalent:

1. there is a run of $\mathcal{S}$ with initial configuration $(q_0, \vec{x_0})$ and final configuration $(q, \vec{x})$ such that each counter has at most r reversals,
2. $((q, \vec{mode}, \sharp\vec{alt}), \vec{x})$ is reachable from $((q_0, \vec{\mathrm{INC}}, \vec{0}), \vec{x_0})$ in $\mathcal{S}'$ for some $\vec{mode}$, $\sharp\vec{alt}$.

The number of distinct pairs $(\vec{mode}, \sharp\vec{alt})$ is bounded by $2^n \times (r+1)^n$ and therefore (1.) is equivalent to the existence of $(\vec{mode}, \sharp\vec{alt})$ among a finite set such that

3. $((q, \vec{mode}, \sharp\vec{alt}), \vec{x})$ is reachable from $((q_0, \vec{\mathrm{INC}}, \vec{0}), \vec{x_0})$ in $\mathcal{S}'$.

By Theorem 4.1.2, the set

$$X_{(\vec{mode}, \sharp\vec{alt})} = \{\vec{x'} \in \mathbb{N}^n : ((q_0, \vec{\mathrm{INC}}, \vec{0}), \vec{x_0}) \xrightarrow{*} ((q, \vec{mode}, \sharp\vec{alt}), \vec{x'})\}$$

is effectively semilinear. This means that one can construct a Presburger formula $\varphi_{(\vec{mode}, \sharp\vec{alt})}$ such that $\mathrm{REL}(\varphi_{(\vec{mode}, \sharp\vec{alt})}) = X_{(\vec{mode}, \sharp\vec{alt})}$ and checking whether $\vec{x} \in X_{(\vec{mode}, \sharp\vec{alt})}$ amounts to verify the satisfiability of the formula

$$(\bigwedge_{i=1}^{i=n} \mathsf{x}_i = \vec{x}(i)) \wedge \varphi_{(\vec{mode}, \sharp\vec{alt})}.$$

Since the satisfiability problem for Presburger arithmetic is decidable, we get an algorithm to solve the reachability problem with bounded reversals. Indeed, it amounts to checking satisfiability of some Presburger formula made a disjunction with at most $2^n(r+1)^n$ disjuncts. QED

In the sequel, when we consider a uniformly reversal-bounded counter automaton or a reversal-bounded initialized counter automaton, it comes with a maximal number of reversals $r \geq 0$ that

has been computed by some means. Remember that in full generality, the reversal-boundedness detection problem is undecidable. Nevertheless, the situation is not that bad, since the problem restricted to VASS is decidable [FS08] and can be solved in exponential space [Dem10]. Hence, for VASS, in case of reversal-boundedness, the value r can be effectively computed. Alternatively, given a counter automaton and a bound $r \geq 0$, it is possible to build a new counter automaton such that each counter has at most r reversals on each runs, possibly at the cost of increasing exponentially the cardinal of the set of control states (see the proof of Corollary 4.1.3). It is sufficient to take the product between $\mathcal{S}$ and a finite-state automaton with number of control states in $\mathcal{O}(r^n)$.

4.2 Reachability sets are semilinear

In this section, we shall show that reachability sets in reversal-bounded (initialized) counter automata are effectively semilinear. Moreover, when uniform reversal-boundedness is satisfied, one can show that the reachability relation is also effectively semilinear. Effectiveness refers here to the possibility to construct Presburger formulae defining exactly those sets or binary relations. The first part of the proof amounts to showing that we can restrict ourselves to 1-reversal-bounded counter automata at the cost of introducing additional counters; this restriction is indeed based on [BB74] for reversal-bounded multistack automata. Then, the second part shows that reachability sets for 1-reversal-bounded counter automata are effectively semilinear, essentially based on Parikh's theorem [Par66] restricted to regular languages. Section 4.2.2 provides the main ingredient of the proof establishing that the commutative image of any regular language is effectively semilinear. Semilinearity is obtained by expressing Birkhoff's equations about control states augmented by connectivity constraints. This analysis allows us to conclude that when a counter automaton is uniformly reversal-bounded, then the reachability relation is effectively definable in Presburger arithmetic.

4.2.1 Hints of the proof

In this section, we briefly provide hints to understand the proof below that establishes that reversal-bounded initialized counter automata have effectively semilinear reachability sets.

The first part of the proof shows that given a reversal-bounded initialized counter automaton $\mathcal{S}$, one can effectively define a uniformly 1-reversal-bounded counter automaton $\mathcal{S}'$ such that the reachability set for $\mathcal{S}$ can be defined as a finite union of reachability sets in $\mathcal{S}'$. Each reachability set in the finite union is parameterized by a control state from $\mathcal{S}'$. This is fine since Presburger arithmetic has disjunction. Moreover, $\mathcal{S}'$ has more counters than $\mathcal{S}$ and in order to be precise each reachability set is obtained by projection which is still fine since Presburger arithmetic has existential quantification (which allows to perform a projection at the level of tuples of natural numbers). This part is based on [BB74] for multistack systems.

The second part of the proof shows that the reachability set of any 1-reversal-bounded initialized counter automaton $(\mathcal{S}, (q_0, \vec{x_0}))$ is effectively semilinear. Given a control state $q \in Q$, we aim at characterizing the counter values $\vec{x_k} \in \mathbb{N}^n$ such that there is a run

$$(q_0, \vec{x_0}) \xrightarrow{a_1} (q_1, \vec{x_1}) \xrightarrow{a_2} (q_2, \vec{x_2}) \cdots \xrightarrow{a_k} (q_k, \vec{x_k})$$

with $q_k = q$ and $u = a_1 \cdots a_k$ over the symbolic alphabet

$$\Sigma = \{\mathrm{inc}(i), \mathrm{dec}(i), \mathrm{zero}(i) : i \in [1, n]\}.$$

We write $\Pi(u)(a)$ to denote the number of occurrences of the letter a in u ($\Pi(u)$ being the Parikh image of u). Observe that for $i \in [1, n]$, we have that $\vec{x_k}(i) = \vec{x_0}(i) + \Pi(u)(\mathrm{inc}(i)) - \Pi(u)(\mathrm{dec}(i))$. Consequently, characterizing the set of counter values $\vec{x_k}$ amounts to determine which sequences u of instructions from $(q_0, \vec{x_0})$ can lead to the control state q. The sequence u satisfies the following simple properties:

1. Control graph of $\mathcal{S}$ allows to perform the sequence of intructions u from the control state q_0 until control state q.
2. Because of 1-reversal-boundedness, for $i \in [1, n]$, the projection of u on the subalphabet $\Sigma_i = \{\mathrm{inc}(i), \mathrm{dec}(i), \mathrm{zero}(i)\}$, written u_{Σ_i}, belongs to

$$\mathrm{zero}(i)^* \mathrm{inc}(i)^* \mathrm{dec}(i)^* \mathrm{zero}(i)^*.$$

These two properties can be checked by building the product finite-state automaton $\mathcal{A}$ between $\mathcal{S}$ understood as a finite-state automaton over Σ and a finite-state automaton over Σ such that the projection of each accepted word is in u_{Σ_i}. Since the Parikh image of $\mathrm{L}(\mathcal{A})$ is effectively semilinear by Parikh Theorem, the set

$$\{\vec{x_0} + (\Pi(u)(\mathrm{inc}(1)), \ldots, \Pi(u)(\mathrm{inc}(n))) - (\Pi(u)(\mathrm{dec}(1)), \ldots, \Pi(u)(\mathrm{dec}(n))) : u \in \mathrm{L}(\mathcal{A})\}$$

is effectively semilinear too. Indeed, if u labels a run then the final configuration has counter values:

$$(u(\vec{x_0}) \stackrel{\text{def}}{=}) \vec{x_0} + \begin{pmatrix} \Pi(u)(\mathrm{inc}(1)) \\ \Pi(u)(\mathrm{inc}(2)) \\ \vdots \\ \Pi(u)(\mathrm{inc}(n)) \end{pmatrix} - \begin{pmatrix} \Pi(u)(\mathrm{dec}(1)) \\ \Pi(u)(\mathrm{dec}(2)) \\ \vdots \\ \Pi(u)(\mathrm{dec}(n)) \end{pmatrix}$$

We have the following inclusion (overapproximation):

$$\{\vec{x} : \exists \text{ run } (q_0, \vec{x_0}) \xrightarrow{*} (q_f, \vec{x})\} \subseteq \{u(\vec{x_0}) : u \in \mathrm{L}(\mathcal{A})\}$$

Let $\varphi_{\mathcal{A}}(\mathsf{z}_1^{inc}, \mathsf{z}_1^{dec}, \mathsf{z}_1^{zero}, \ldots, \mathsf{z}_n^{inc}, \mathsf{z}_n^{dec}, \mathsf{z}_n^{zero})$ capturing the Parikh image of $\mathrm{L}(\mathcal{A})$ and ψ be

$$\exists\, \mathsf{z}_1^{inc}, \ldots, \mathsf{z}_n^{zero}\ (\mathsf{x}_1 = \vec{x_0}(1) + \mathsf{z}_1^{inc} - \mathsf{z}_1^{dec}) \wedge \cdots$$

$$\cdots \wedge (\mathsf{x}_n = \vec{x_0}(n) + \mathsf{z}_n^{inc} - \mathsf{z}_n^{dec}) \wedge \varphi_{\mathcal{A}}(\mathsf{z}_1^{inc}, \ldots, \mathsf{z}_n^{zero})$$

We have $\mathrm{REL}(\psi) = \{u(\vec{x_0}) : u \in \mathrm{L}(\mathcal{A})\}$ but this is not sufficient ! Not every word accepted by $\mathcal{A}$ corresponds to a sequence of instructions from $(q_0, \vec{x_0})$ leading to q. The sequence u satisfies also the following properties for $i \in [1, n]$:

1. For every prefix v of u_{Σ_i}, $\vec{x_0}(i) + \Pi(v)(\mathrm{inc}(i)) - \Pi(v)(\mathrm{dec}(i)) \geq 0$ (counter values are nonnegative). Since $u_{\Sigma_i} \in \mathrm{zero}(i)^* \mathrm{inc}(i)^* \mathrm{dec}(i)^* \mathrm{zero}(i)^*$, it is sufficient to satisfy $\vec{x_0}(i) + \Pi(u)(\mathrm{inc}(i)) - \Pi(u)(\mathrm{dec}(i)) \geq 0$.

2. If $\vec{x_0}(i) \neq 0$, then the first letter of u_{Σ_i} is different from $\mathrm{zero}(i)$.
3. If the last letter of u_{Σ_i} is equal to $\mathrm{zero}(i)$, then

$$\vec{x_0}(i) + \Pi(u)(\mathrm{inc}(i)) - \Pi(u)(\mathrm{dec}(i)) = 0.$$

Conditions (1.) and (3.) can be easily expressed with the Presburger formula characterizing the Parikh image of $\mathrm{L}(\mathcal{A})$. Condition (2.) is taken care by the initial states of $\mathcal{A}$ (see the definition of $\mathcal{A}^{(q_0,\vec{v_0})}$ in Section 4.2.4). Now, it remains to provide the details of the proof. Let us start by Parikh Theorem for regular languages.

4.2.2 Parikh image of regular languages

We recall that a *finite-state automaton* is a tuple $\mathcal{A} = (\Sigma, Q, Q_0, \delta, F)$ such that

- Σ is a finite *alphabet*,
- Q is a finite set of *states*,
- $Q_0 \subseteq Q$ is the set of *initial* states,
- the *transition relation* δ is a subset of $Q \times \Sigma \times Q$,
- $F \subseteq Q$ is a set of *final* states.

Given $q \in Q$ and $a \in \Sigma$, we also write $\delta(q, a)$ to denote the set of states q' such that $(q, a, q') \in \delta$. A *run* ρ of $\mathcal{A}$ is a sequence $q_0 \xrightarrow{a_0} q_1 \xrightarrow{a_1} q_2 \ldots$ such that for every $i \geq 0$, $(q_i, a_i, q_{i+1}) \in \delta$ (also written $q_i \xrightarrow{a_i} q_{i+1}$). The finite run $\rho = q_0 \xrightarrow{a_0} q_1 \xrightarrow{a_1} q_2 \ldots \xrightarrow{a_{n-1}} q_n$ is *successful* if $q_0 \in Q_0$ is initial and $q_n \in F$ is final. The *label* of ρ is the finite word $\sigma = a_0 a_1 \cdots a_n$. The automaton $\mathcal{A}$ *accepts* the language $\mathrm{L}(\mathcal{A})$ of finite words $u \in \Sigma^*$ such that there exists a successful run of $\mathcal{A}$ on the word u, i.e., with label u.

Let $\Sigma = \{a_1, \ldots, a_k\}$ be an finite alphabet equipped with an arbitrary linear ordering of the letters, say $a_1 < \cdots < a_k$. Given a word $u \in \Sigma^*$, its *Parikh image* $\Pi(u)$ is defined as the tuple $\Pi(u) \in \mathbb{N}^k$ such that for $i \in [1, k]$, $\Pi(u)(i)$ is the number of occurrences of the letter a_i in the word u. For instance, the Parikh of the word $abaab$ under the ordering $a < b$ is the tuple $\begin{pmatrix} 3 \\ 2 \end{pmatrix}$. Naturally, the *Parikh image* of the language $\mathrm{L} \subseteq \Sigma^*$, written $\Pi(\mathrm{L})$ is the set $\{\Pi(u) \in \mathbb{N}^k : u \in \mathrm{L}\}$. Parikh's remarkable result states that the Parikh image of any context-free language is semilinear [Par66] and that its representation is effectively computable from a pushdown automaton. Below, we provide the proof for regular languages only, which is sufficient to deal with reversal-boundedness in the simple case. By the way, the proof of Parikh's Theorem can be also found in [Koz97, Chapter H]. An alternative proof is also given in [Esp97] based on the result that the reachability relation for communication-free Petri nets is effectively semilinear.

Theorem 4.2.1. [Par66] Let Σ be a finite alphabet (equipped with a linear ordering) and $\mathcal{A}$ be a finite-state automaton over Σ. Then, one can compute effectively a Presburger formula $\varphi_{\mathcal{A}}(\mathsf{x}_1, \ldots, \mathsf{x}_k)$ such that for every valuation $\mathbf{val}$, we have $\mathbf{val} \models \varphi_{\mathcal{A}}(\mathsf{x}_1, \ldots, \mathsf{x}_k)$ iff there is a finite word $u \in \mathrm{L}(\mathcal{A})$ such that $\Pi(u) = (\mathbf{val}(\mathsf{x}_1), \ldots, \mathbf{val}(\mathsf{x}_k))$.

Proof: Given a direct graph $G = (V, E)$, the proof below essentially determines when a map $f : E \to \mathbb{N}$ is the Parikh image of a path in G. Indeed, regular languages are definable from finite-state automata and words accepted by such automata are strongly related to paths (runs). Roughly speaking, f corresponds to a path iff the subgraph induced by f is connected and the number of edges entering in a node is equal to the number of edges going out of the node. This may be slightly different for the initial node and for the final node of the path (see details below).

Let $\mathcal{A} = (\Sigma, Q, Q_0, \delta, F)$ be a finite-state automaton. Given a transition $t = q \xrightarrow{a} q'$, we write $beg(t)$ to denote q, $end(t)$ to denote q' and $\Sigma(t)$ to denote a.

A *path* of $\mathcal{A}$ is a finite sequence $\pi = t_1 \cdots t_k$ of transitions such that for $i \in [1, k-1]$, $end(t_{i+1}) = beg(t_i)$. We say that π is a path from $beg(t_1)$ to $end(t_k)$. We admit empty paths of length 0, one for each state q. Two paths π and π' are *consecutive* if $end(\pi) = beg(\pi')$. When π and π' are consecutive, we write $\pi\pi'$ to denote the path obtained by concatenation. The *image* of $\pi = t_1 \cdots t_k$ is a map $\mathcal{I}_\pi : \delta \to \mathbb{N}$ that counts how many times each transition is used in π, i.e., $\mathcal{I}_\pi(t) = \mathrm{card}(\{i \in [1, k] : t_i = t\})$. Given $\mathcal{I} : \delta \to \mathbb{N}$, we write $\mathcal{A}_\mathcal{I}$ to denote its restriction to the transitions in $\mathcal{I}$, i.e. to denote the directed labelled graph (Σ, Q', δ') such that

- $\delta' = \{t \in \delta : \mathcal{I}(t) > 0\}$.
- Q' is the set of states q for which at least a transition in δ' begins or ends by q. So, $Q' = \{beg(t), end(t) : t \in \delta'\}$.

A directed labelled graph (Σ, Q, δ) is *connected* iff for all $q, q' \in Q$, there is a path from q to q' in $(\Sigma, Q, \delta \cup \overline{\delta})$ with $\overline{\delta} = \{end(t) \xrightarrow{\Sigma(t)} beg(t) : t \in \delta\}$. Basically, we forget about the direction of transitions.

The two following properties can be easily shown:

(P1) Let π_1 and π_2 be two paths sharing at least one state such that $beg(\pi_2) = end(\pi_2)$. Then, there is a path π such that $\mathcal{I}_\pi = \mathcal{I}_{\pi_1} + \mathcal{I}_{\pi_2}$, $beg(\pi) = beg(\pi_1)$ and $end(\pi) = end(\pi_1)$.

(P2) Let $\mathcal{A}$ be a finite-state automaton and $\pi = t_1 \cdots t_k$ be a path from q to q' with image $\mathcal{I}_\pi$. The following statements hold true:

(I) $\mathcal{A}_{\mathcal{I}_\pi}$ is connected.

(II) If $q = q'$, i.e. π is a cycle, then for each state q'', the number of transitions entering in q'' is equal to the number of transitions going out of q''. In symbols, for every $q'' \in Q$, we get the satisfaction of the following equation.

$$\sum_{t \in \delta \text{ s.t. } end(t) = q''} \mathcal{I}_\pi(t) \; - \sum_{t \in \delta \text{ s.t. } beg(t) = q''} \mathcal{I}_\pi(t) = 0.$$

(III) If $q \neq q'$, then for every $q'' \in Q \setminus \{q, q'\}$, we are in the position as in (II). However, the number of transitions entering in q is one less than the number of transitions going out of q. Similarly, the number of transitions entering in q' is one more than the number of transitions going out of q'. In symbols, we get the satisfaction of the following equations.

1. for every $q'' \in Q \setminus \{q, q'\}$,
$$\sum_{t\in\delta \text{ s.t. } end(t)=q''} \mathcal{I}_\pi(t) \; - \sum_{t\in\delta \text{ s.t. } beg(t)=q''} \mathcal{I}_\pi(t) = 0.$$
2. $\sum_{t\in\delta \text{ s.t. } end(t)=q} \mathcal{I}_\pi(t) \; - \sum_{t\in\delta \text{ s.t. } beg(t)=q} \mathcal{I}_\pi(t) = -1.$
3. $\sum_{t\in\delta \text{ s.t. } end(t)=q'} \mathcal{I}_\pi(t) \; - \sum_{t\in\delta \text{ s.t. } beg(t)=q'} \mathcal{I}_\pi(t) = 1.$

Now, we can show the property below that is a variant of the characterization for the existence of Eulerian paths in a directed graph. Let $\mathcal{A}$ be a finite-state automaton and $\mathcal{I} : \delta \to \mathbb{N}$ be a map. Based on the properties (P1) and (P2) and on a further analysis that is omitted (see [Reu90] for further details), we can show the property (♯) below:

(♯) $\mathcal{I}$ is the image of some path iff there are q and q' in Q such that (I)–(III) hold true (by replacing $\mathcal{I}_\pi$ by $\mathcal{I}$).

For each letter $a \in \Sigma$, we introduce the variable x_a. Similarly, for each transition $t \in \delta$, we introduce the variable $\mathsf{x}_{t'}$. Say $\delta = \{t_1, \ldots t_{k'}\}$.

The Presburger formula $\varphi_{\mathcal{A}}(\mathsf{x}_{a_1}, \ldots, \mathsf{x}_{a_k})$ is of the form below:

$$\exists \mathsf{x}_{t_1} \cdots \mathsf{x}_{t_{k'}} (\bigwedge_{i=1}^{k} \mathsf{x}_{a_i} = \sum_{\Sigma(t)=a_i} \mathsf{x}_t) \wedge$$

$$(\bigvee_{q_0\in Q_0, q_f\in F} \; \bigvee_{\text{connected } (Q',\delta'),\ q_0,q_f\in Q'} \varphi_{(Q',q_0,q_f,\delta')} \wedge (\bigwedge_{t\in\delta'} \mathsf{x}_t > 0) \wedge (\bigwedge_{t\in(\delta\setminus\delta')} \mathsf{x}_t = 0))$$

In the generalized disjunction over the connected direct labelled graphs (Q', δ'), we assume that $Q' \subseteq Q$ and $\delta' \subseteq \delta \cap Q' \times \Sigma \times Q'$. It remains to explain how the formula $\varphi_{(Q,q_0,q_f,\delta')}$ is defined based on the previous properties.

- ⋆ If $q_0 = q_f$, then $\varphi_{(Q',q_0,q_f,\delta')}$ takes the following value (see (II)):
$$\bigwedge_{q''\in Q'} (\sum_{t\in\delta' \text{ s.t. } end(t)=q''} \mathsf{x}_t \; - \sum_{t\in\delta' \text{ s.t. } beg(t)=q''} \mathsf{x}_t = 0.)$$
- ⋆ If $q_0 \neq q_f$, then $\varphi_{(Q',q_0,q_f,\delta')}$ is the conjunction $\varphi_1 \wedge \varphi_2 \wedge \varphi_3$ (see (III)(1.–3.)):
  - ∗ $\varphi_1 = \bigwedge_{q''\in Q'\setminus\{q_0,q_f\}} (\sum_{t\in\delta' \text{ s.t. } end(t)=q''} \mathsf{x}_t \; - \sum_{t\in\delta' \text{ s.t. } beg(t)=q''} \mathsf{x}_t = 0).$
  - ∗ $\varphi_2 = \sum_{t\in\delta' \text{ s.t. } end(t)=q_0} \mathsf{x}_t \; - \sum_{t\in\delta' \text{ s.t. } beg(t)=q_0} \mathsf{x}_t = -1.$
  - ∗ $\varphi_3 = \sum_{t\in\delta' \text{ s.t. } end(t)=q_f} \mathsf{x}_t \; - \sum_{t\in\delta' \text{ s.t. } beg(t)=q_f} \mathsf{x}_t = 1.$

Condition (I) for connectivity is taking care by a case analysis; (Q', δ') has to be connected and there is an exponential amount of such restrictions. The formula $(\bigwedge_{t\in\delta'} \mathsf{x}_t > 0) \wedge (\bigwedge_{t\in(\delta\setminus\delta')} \mathsf{x}_t = 0)$ plays also a central role since it guarantees that exactly the transitions in δ' have been considered. QED

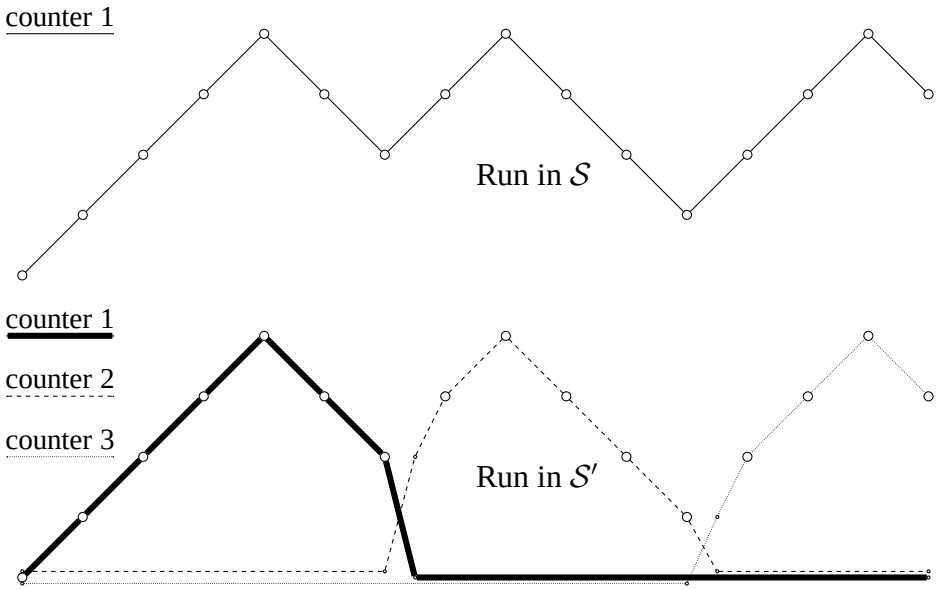


Figure 4.3: Counter 1 in $\mathcal{S}$ and counters 1, 2 and 3 in $\mathcal{S}'$

4.2.3 1-reversal-bounded counter automata

In this section, we show that in order to show that reachability sets [resp. reachability relation] for [resp. uniformly] reversal-bounded counter automata are Presburger-definable, it is sufficient to show that the reachability relation for uniformly 1-reversal-bounded counter automata is effectively Presburger-definable.

Let $(\mathcal{S}, (q_0, \vec{x}_0))$ be an initialized counter automaton that is r-reversal-bounded for some $r \geq 0$ with $\mathcal{S} = (Q, n, \delta)$. Before going any further, let us introduce a bit of vocabulary. A *phase* for a counter $i \in [1, n]$ in a finite run is a finite sequence of instructions dealing with counter i and extracted from a larger sequence of instructions obtained by erasing instructions dealing with other counters, such that the sequence is either in $\mathrm{zero}(i)^* \cdot \mathrm{inc}(i)^*$ (*increasing* phase) or in $\mathrm{dec}(i)^* \cdot \mathrm{zero}(i)^*$ (*decreasing* phase). A *biphase* is defined as a sequence in $\mathrm{zero}(i)^* \cdot \mathrm{inc}(i)^* \cdot \mathrm{dec}(i)^* \cdot \mathrm{zero}(i)^*$ obtained by concatenating an increasing phase with a decreasing phase. A biphase is *complete* when it is in $\mathrm{zero}(i)^* \cdot \mathrm{inc}(i)^+ \cdot \mathrm{dec}(i)^+ \cdot \mathrm{zero}(i)^*$, that is there is at least an increment followed by a decrement. In Figure 4.3, the counter 1 has three complete biphases in the run of $\mathcal{S}$. When the counter i has value different from zero, it is clear that any increasing phase started from this value is in $\mathrm{inc}(i)^*$. It is worth observing that 1-reversal-boundedness implies that counters admit at most a complete biphase on each run starting at the initial configuration.

From $(\mathcal{S}, (q_0, \vec{x}_0))$, we shall build a counter automaton $\mathcal{S}'$ that is uniformly 1-reversal-bounded for which the set of configurations reachable from $(q_0, \vec{x}_0)$ in $\mathcal{S}$, can be characterized as a finite union of reachability sets from $\mathcal{S}'$, possibly by performing some projections since $\mathcal{S}'$ has more components/counters than $\mathcal{S}$. Without any loss of generality, we can assume that r is even and each counter has at most $\frac{r}{2}$ complete biphases for any run starting at $(q_0, \vec{x}_0)$. A *global biphase vector* is an element from $[0, \frac{r}{2}]^n$ indicating for each counter the ordinal of the current biphase. Similarly, a *refined global biphase vector* $\vec{w}$ is an element from $(\{\mathrm{INC}, \mathrm{DEC}\} \times [0, \frac{r}{2}])^n$ that also specifies whether the presence in some biphase is currently either in the increasing phase or in the

decreasing phase. These vectors shall be encoded in the control states of $\mathcal{S}'$ (there is only a finite amount of such vectors) and for each counter and each biphase in $\mathcal{S}$, one specific counter in $\mathcal{S}'$ mimicks the original counter (see an illustration in the bottom part of Figure 4.3). More precisely, when in a run of $\mathcal{S}$, a counter i enters in a new biphase l (typically this occurs when passing from a decreasing phase to an increasing phase), this is mimicked in $\mathcal{S}'$ by introducing a new counter, say i' that depends only on i and on l. To do so, i' is incremented until the counter i'' attached to counter i and biphase $l-1$ reaches the value zero; i'' is the counter in $\mathcal{S}'$ that mimicks the counter i from $\mathcal{S}$ in the previous biphase $l-1$. Then, the counter i' in $\mathcal{S}'$ behaves as i until a new biphase is observed (see Figure 4.3). Hence, the number of counters in $\mathcal{S}'$ is $n' = n \times (1 + \frac{r}{2})$. Depending on the current biphase ordinal, for each counter we shall be able to determine which counters are *active*. Let $c : [0, \frac{r}{2}] \times [1, n] \to [1, n']$ be the map defined by $c(l, i) \stackrel{\text{def}}{=} (i-1)(\frac{r}{2}+1) + (1+l)$ that determines for each counter and each biphase ordinal, the corresponding counter in $\mathcal{S}'$. For instance with $n = 2$ and $\frac{r}{2} = 2$, we have:

$$(\overbrace{c(0,1), c(1,1), c(2,1)}^{\text{3 counters mimicking counter 1}}, \overbrace{c(0,2), c(1,2), c(2,2)}^{\text{3 counters mimicking counter 2}}) = (1,2,3,4,5,6)$$

Given a refined global biphase vector $\vec{w} \in (\{\text{INC}, \text{DEC}\} \times [0, \frac{r}{2}])^n$, we write $NB(\vec{w}) \in [0, \frac{r}{2}]^n$ to denote the corresponding global biphase vector obtained from $\vec{w}$ by omitting the information about the type of the current phases. Similarly, we write $PH(\vec{w}) \in \{\text{INC}, \text{DEC}\}^n$ to denote the restriction of $\vec{w}$ obtained from $\vec{w}$ by omitting the biphase ordinals.

Given a configuration $((q, \vec{w}), \vec{x})$ of $\mathcal{S}'$, we write $Act(((q, \vec{w}), \vec{x}))$ to denote the corresponding counter values in $\mathbb{N}^n$ by selecting only values corresponding to active counters (there are exactly n): for $i \in [1, n]$, we have $Act(((q, \vec{w}), \vec{x}))(i) \stackrel{\text{def}}{=} \vec{x}(c(NB(\vec{w})(i), i))$. In the sequel, we write $\mathcal{S}'[\vec{w}, i]$ instead of $c(NB(\vec{w})(i), i)$ to denote the counter in $\mathcal{S}'$ that behaves as the counter i in its $NB(\vec{w})(i)$th biphase in $\mathcal{S}$. So, the value of counter i in $\mathcal{S}$ when the run is currently in the refined global biphase vector $\vec{w}$ is taken care by the counter $\mathcal{S}'[\vec{w}, i]$ in $\mathcal{S}'$.

We are now in position to define the counter automaton $\mathcal{S}' = (Q', n', \delta')$ such that $Q \times (\{\text{INC}, \text{DEC}\} \times [0, \frac{r}{2}])^n \subseteq Q'$; the unspecified additional control states will appear to be auxiliary. It remains to define the transition relation δ'.

- ⋆ For all $q \xrightarrow{\text{inc}(i)} q'$ and $\vec{w} \in (\{\text{INC}, \text{DEC}\} \times [0, \frac{r}{2}])^n$ such that $PH(\vec{w})(i) = \text{INC}$, we have $(q, \vec{w}) \xrightarrow{\text{inc}(\mathcal{S}'[\vec{w},i])} (q', \vec{w}) \in \delta'$.
- ⋆ For all $q \xrightarrow{\text{dec}(i)} q'$ and $\vec{w} \in (\{\text{INC}, \text{DEC}\} \times [0, \frac{r}{2}])^n$ such that $PH(\vec{w})(i) = \text{DEC}$, we have $(q, \vec{w}) \xrightarrow{\text{dec}(\mathcal{S}'[\vec{w},i])} (q', \vec{w}) \in \delta'$.
- ⋆ For all $q \xrightarrow{\text{zero}(i)} q'$ and $\vec{w} \in (\{\text{INC}, \text{DEC}\} \times [0, \frac{r}{2}])^n$, we have $(q, \vec{w}) \xrightarrow{\text{zero}(\mathcal{S}'[\vec{w},i])} (q', \vec{w}) \in \delta'$.
- ⋆ For all $q \xrightarrow{\text{dec}(i)} q'$ and $\vec{w} \in (\{\text{INC}, \text{DEC}\} \times [0, \frac{r}{2}])^n$ such that $PH(\vec{w})(i) = \text{INC}$, we have $(q, \vec{w}) \xrightarrow{\text{dec}(\mathcal{S}'[\vec{w},i])} (q', \vec{w'}) \in \delta'$ where $\vec{w'}(j) = \vec{w}(j)$ for $j \neq i$ and $\vec{w'}(i) = (\text{DEC}, l)$ with $\vec{w}(i) = (\text{INC}, l)$.
- ⋆ The remaining case in this definition is the more complex one since it corresponds to a biphase change. For all $q \xrightarrow{\text{inc}(i)} q'$ and $\vec{w} \in (\{\text{INC}, \text{DEC}\} \times [0, \frac{r}{2}])^n$ such that $\vec{w}(i) =$

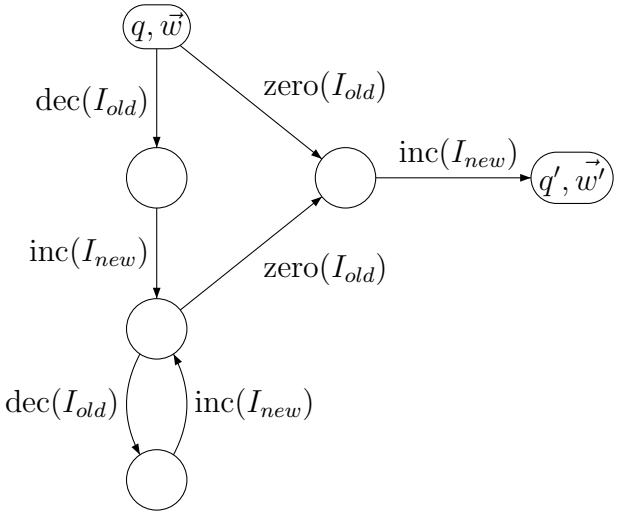


Figure 4.4: Completing δ'

(DEC, l), we add the transitions present in Figure 4.4 (there are four new auxiliary control states).

We put $\vec{w'} \in (\{\mathrm{INC}, \mathrm{DEC}\} \times [0, \frac{r}{2}])^n$ with $\vec{w'}(j) = \vec{w}(j)$ for $j \neq i$ and $\vec{w'}(i) = (\mathrm{INC}, l+1)$: a new biphase is considered. We also use the shortcuts $I_{old} = \mathcal{S}'[\vec{w}, i]$ and $I_{new} = \mathcal{S}'[\vec{w'}, i]$. Observe that the effect of reaching $(q', \vec{w'})$ from $(q, \vec{w})$ is to transfer the value of counter I_{old} to counter I_{new}.

First, let us state a few properties about $\mathcal{S}'$ and its relationships with $(\mathcal{S}, (q_0, \vec{x_0}))$.

Lemma 4.2.2.

(I) $\mathcal{S}'$ is uniformly 1-reversal-bounded.

(II) Let $\vec{x'_0} \in \mathbb{N}^{n'}$ such that

1. for $i \in [1, n]$, $\vec{x'_0}(\mathcal{S}'[(\vec{\mathrm{INC}}, 0), i]) = \vec{x_0}(i)$,
2. for $j \in ([1, n'] \setminus \{\mathcal{S}'[(\vec{\mathrm{INC}}, 0), i] : i \in [1, n]\})$, we have $\vec{x'_0}(j) = 0$.

Then, $\{\vec{x} \in \mathbb{N}^n : (q_0, \vec{x}_0) \xrightarrow{*} (q, \vec{x}) \text{ in } \mathcal{S}\} = \{Act(((q, \vec{w}), \vec{x})) \in \mathbb{N}^n : \exists\, \vec{w}\, ((q_0, (\vec{\mathrm{INC}}, 0)), \vec{x'_0}) \xrightarrow{*} ((q, \vec{w}), \vec{x}) \text{ in } \mathcal{S}'\}$ for every $q \in Q$.

(III) Suppose that $((q_0, (\vec{\mathrm{INC}}, 0)), \vec{x'_0}) \xrightarrow{*} ((q, \vec{w}), \vec{x})$. For $j \in ([1, n'] \setminus \{\mathcal{S}'[\vec{w}, i] : i \in [1, n]\})$, we have $\vec{x}(j) = 0$.

Lemma 4.2.2(I) is by simple inspection of the construction of $\mathcal{S}'$. Lemma 4.2.2(III) reflects the property that a counter in $\mathcal{S}'$ is either active or equal to zero (apart from the configurations with auxiliary control states). Based on Lemma 4.2.2, we can show the following lemma.

Lemma 4.2.3.

(I) If for $(q, \vec{w}) \in (\{\mathrm{INC}, \mathrm{DEC}\} \times [0, \frac{r}{2}])^n$, the set

$$\{\vec{x} : ((q_0, (\mathrm{I\vec{N}C}, 0)), \vec{x_0'}) \xrightarrow{*} ((q, \vec{w}), \vec{x}) \text{ in } \mathcal{S}'\}$$

is effectively semilinear, then $\{\vec{x} \in \mathbb{N}^n : (q_0, \vec{x}_0) \xrightarrow{*} (q, \vec{x}) \text{ in } \mathcal{S}\}$ is effectively semilinear too, for every control state q.

(II) If $\mathcal{S}$ is uniformly r-reversal-bounded and the reachability relation for $\mathcal{S}'$ is Presburger-definable, then the reachability relation for $\mathcal{S}$ is Presburger-definable too.

Proof: (I) Suppose one can compute effectively a formula $\varphi_{(q,\vec{w})}(\mathsf{x}_1, \ldots, \mathsf{x}_{n'})$ characterizing the configurations reachable from $((q_0, (\mathrm{I\vec{N}C}, 0)), \vec{x_0'})$ with control state $(q, \vec{w})$. The formula below characterizes the configurations reachable from $(q_0, \vec{x}_0)$ with control state q:

$$\bigvee_{\vec{w} \in [0, \frac{r}{2}]^n} (\exists\, \mathsf{y}_1 \cdots \mathsf{y}_{n'}\ \varphi_{(q,\vec{w})}(\mathsf{y}_1, \ldots, \mathsf{y}_{n'}) \wedge (\bigwedge_{i \in [1,n]} \mathsf{x}_i = \mathsf{y}_{\mathcal{S}'[\vec{w},i]})).$$

(II) By assumption, for $q, q' \in Q'$, there exists a formula $\varphi_{q,q'}(\mathsf{x}_1, \ldots, \mathsf{x}_{n'}, \mathsf{y}_1, \ldots, \mathsf{y}_{n'})$ such that for every valuation $\mathbf{val}$, we have

$$\mathbf{val} \models \varphi_{q,q'} \text{ iff } (q, (\mathbf{val}(\mathsf{x}_1), \ldots, \mathbf{val}(\mathsf{x}_{n'}))) \xrightarrow{*} (q', (\mathbf{val}(\mathsf{y}_1), \ldots, \mathbf{val}(\mathsf{y}_{n'}))) \text{ in } \mathcal{S}'.$$

The Presburger formula $\varphi(\mathsf{z}_1, \ldots, \mathsf{z}_n, \mathsf{z}'_1, \ldots, \mathsf{z}'_n)$ encoding the reachability relation from control state $q \in Q$ to control state $q' \in Q$ is defined as follows:

$$\bigvee_{\vec{w}, \vec{w'} \in [0, \frac{r}{2}]^n} (\exists\, \mathsf{x}_1 \cdots \mathsf{x}_{n'},\ \mathsf{y}_1 \cdots \mathsf{y}_{n'}\ \varphi_{(q,\vec{w}),(q',\vec{w'})}$$

$$\wedge (\bigwedge_{i \in [1,n]} \mathsf{z}'_i = \mathsf{y}_{\mathcal{S}'[\vec{w},i]}) \wedge (\bigwedge_{i \in [1,n]} \mathsf{z}_i = \mathsf{x}_{\mathcal{S}'[(\mathrm{I\vec{N}C},0),i]}) \wedge (\bigwedge_{j \in NA} \mathsf{x}_j = 0)).$$

where $NA = ([1, n'] \setminus \{\mathcal{S}'[(\mathrm{I\vec{N}C}, 0), i] : i \in [1, n]\})$ (set of initial "nonactive" counters). QED

Figure 4.3 presents the behaviour of the counter 1 in $\mathcal{S}$ ($\frac{r}{2} = 3$) and the behaviour of the counters 1, 2 and 3 in $\mathcal{S}'$.

4.2.4 Reachability sets are effectively semilinear

Let $\mathcal{S} = (Q, n, \delta)$ be a counter automaton such that $(\mathcal{S}, (q_0, \vec{x_0}))$ is 1-reversal-bounded. For each finite run from the initial configuration $(q_0, \vec{x_0})$, the restriction to counter i of the sequence of actions corresponds to a biphase of the form $\mathrm{zero}(i)^* \cdot \mathrm{inc}(i)^* \cdot \mathrm{dec}(i)^* \cdot \mathrm{zero}(i)^*$. When $\vec{x_0}(i) > 0$, the biphase can only be an element of $\mathrm{inc}(i)^* \cdot \mathrm{dec}(i)^* \cdot \mathrm{zero}(i)^*$.

Let us define the auxiliary vector $\vec{v_0} \in \{eq(0), neq(0)\}^n$ that essentially records which counters in $\vec{x_0}$ takes the value zero: for $i \in [1, n]$, $\vec{v_0}(i) \stackrel{\text{def}}{=} eq(0)$ if $\vec{x_0}(i) = 0$ otherwise $\vec{v_0}(i) \stackrel{\text{def}}{=} neq(0)$. Now, for $i \in [1, n]$ we write $S_i^{\vec{v_0}}$ to denote either $\{\xrightarrow{0}_1, \nearrow, \searrow, \xrightarrow{0}_2\}$ when $\vec{v_0}(i) = eq(0)$ or $\{\nearrow, \searrow, \xrightarrow{0}_2\}$ when $\vec{v_0}(i) = neq(0)$. Each element in $S_i^{\vec{v_0}}$ corresponds to an action in

$\{\mathrm{zero}(i), \mathrm{inc}(i), \mathrm{dec}(i)\}$ but we wish to possibly distinguish in a biphase the initial zero-tests (represented by the letter $\xrightarrow{0}_1$) from the terminal zero-tests (represented by the letter $\xrightarrow{0}_2$). We write $S^{\vec{v_0}}$ to denote the cartesian product $S_1^{\vec{v_0}} \times \cdots \times S_n^{\vec{v_0}}$. It is worth noting that assuming that $\vec{x_0} = \vec{0}$ would make unecessary the above definitions and would not simplify the main ingredients of the proof below. Nevertheless, this smoothly prepares the treatment for uniform 1-reversal-boundedness.

From $(\mathcal{S}, (q_0, \vec{x_0}))$, we define a finite-state automaton $\mathcal{A}^{(q_0,\vec{v_0})}$ that can be viewed as a product between $\mathcal{S}$ (viewed as a finite-state automaton in which the actions on counters are understood as letters from a finite alphabet) and a finite-state automaton that checks that the sequence of actions for each counter is compatible with 1-reversal-boundedness and with the initial vector $\vec{v_0}$ (see explanations in Section 4.2.1). Hence, counters are removed but at the cost of adding a bit more control in order to preserve 1-reversal-boundedness at the level of symbolic actions. So, the language accepted by $\mathcal{A}^{(q_0,\vec{v_0})}$ overapproximates the sequences of instructions obtained from the runs of $(\mathcal{S}, (q_0, \vec{x_0}))$ but it shall be possible to add constraints to obtain precisely the sequences from $(\mathcal{S}, (q_0, \vec{x_0}))$. The finite-state automaton $\mathcal{A}^{(q_0,\vec{v_0})} = (\Sigma, Q', Q_0, \delta', Q')$ is defined as follows:

- $\Sigma = \{\mathrm{inc}(i), \mathrm{dec}(i), \mathrm{zero}(i) : i \in [1, n]\}$.
- $Q' = Q \times S^{\vec{v_0}}$.
- $Q_0 = \{(q_0, \vec{v_0'})\}$ where for $i \in [1, n]$, $\vec{v_0'}(i) = \xrightarrow{0}_1$ if $\vec{v_0} = eq(0)$, otherwise $\vec{v_0'}(i) = \nearrow$.

It remains to define δ'.

- If $q \xrightarrow{\mathrm{zero}(i)} q' \in \delta$, then $(q, \vec{v}) \xrightarrow{\mathrm{zero}(i)} (q', \vec{v'}) \in \delta'$ with
  1. for $j \neq i$, $\vec{v}(j) = \vec{v'}(j)$,
  2. either $\vec{v}(i) = \vec{v'}(i) \in \{\xrightarrow{0}_1, \xrightarrow{0}_2\}$ or $\vec{v}(i) = \searrow$ and $\vec{v'}(i) = \xrightarrow{0}_2$.
- If $q \xrightarrow{\mathrm{inc}(i)} q' \in \delta$, then $(q, \vec{v}) \xrightarrow{\mathrm{inc}(i)} (q', \vec{v'}) \in \delta'$ with
  1. for $j \neq i$, $\vec{v}(j) = \vec{v'}(j)$,
  2. either $\vec{v}(i) = \vec{v'}(i) = \nearrow$ or $\vec{v}(i) = \xrightarrow{0}_1$ and $\vec{v'}(i) = \nearrow$.
- If $q \xrightarrow{\mathrm{dec}(i)} q' \in \delta$, then $(q, \vec{v}) \xrightarrow{\mathrm{dec}(i)} (q', \vec{v'}) \in \delta'$ with
  1. for $j \neq i$, $\vec{v}(j) = \vec{v'}(j)$,
  2. either $\vec{v}(i) = \vec{v'}(i) = \searrow$ or $\vec{v}(i) = \nearrow$ and $\vec{v'}(i) = \searrow$.

By Theorem 4.2.1, for every $(q, \vec{v}) \in Q'$, one can effectively compute a Presburger formula

$$\varphi_{(q,\vec{v})}^{(q_0,\vec{v_0})}(\mathsf{x}_{inc}^1, \mathsf{x}_{dec}^1, \mathsf{x}_{zero}^1, \ldots, \mathsf{x}_{inc}^n, \mathsf{x}_{dec}^n, \mathsf{x}_{zero}^n)$$

such that for every valuation $\mathbf{val}$, we have $\mathbf{val} \models \varphi_{(q,\vec{v})}^{(q_0,\vec{v_0})}$ iff there is a finite word u in the language $\mathrm{L}((\Sigma, Q', Q_0, \delta', \{(q, \vec{v})\}))$ such that $\Pi(u) = (\mathbf{val}(\mathsf{x}_{inc}^1), \ldots, \mathbf{val}(\mathsf{x}_{zero}^n))$.

Given $q \in Q$, let $\psi_q(\mathsf{y}_1, \ldots, \mathsf{y}_n)$ be the formula below:

$$\bigvee_{\vec{v} \in S^{\vec{v_0}}} \exists \mathsf{x}^1_{inc}, \cdots, \mathsf{x}^n_{zero}\ (\varphi^{(q_0,\vec{v_0})}_{(q,\vec{v})}(\mathsf{x}^1_{inc}, \ldots, \mathsf{x}^n_{zero})$$

$$\wedge\Big(\bigwedge_{i\in[1,n] \text{ s.t. } \vec{v}(i)\in\{\xrightarrow{0}_1,\xrightarrow{0}_2\}} \mathsf{y}_i = 0\Big) \wedge \Big(\bigwedge_{i\in[1,n]} \mathsf{y}_i = \mathsf{x}^i_{inc} + \vec{x_0}(i) - \mathsf{x}^i_{dec}\Big)$$

Lemma 4.2.4. For all $q \in Q$ and all valuations $\mathbf{val}$,

$$\text{we have } (q_0, \vec{x_0}) \xrightarrow{*} (q, (\mathbf{val}(\mathsf{y}_1), \ldots, \mathbf{val}(\mathsf{y}_n))) \text{ iff } \mathbf{val} \models \psi_q(\mathsf{y}_1, \ldots, \mathsf{y}_n).$$

Proof: Let $(q_0, \vec{x_0}) \xrightarrow{a_0} (q_1, \vec{x_1}) \xrightarrow{a_1} \cdots \xrightarrow{a_{k-1}} (q_k, \vec{x_k})$ be a run with $u = a_0 \cdots a_{k-1} \in \Sigma$. It is clear that by construction of $\mathcal{A}^{(q_0,\vec{v_0})}$, there is $\vec{v} \in S^{\vec{v_0}}$ such that $u \in \mathrm{L}((\Sigma, Q', Q_0, \delta', \{(q_k, \vec{v})\}))$. Let $\mathbf{val}$ be the valuation such that for $i \in [1, n]$, $\mathbf{val}(\mathsf{x}^i_{inc})$ [resp. $\mathbf{val}(\mathsf{x}^i_{dec})$, $\mathbf{val}(\mathsf{x}^i_{zero})$] is equal to the number of occurrences of $\mathrm{inc}(i)$ [resp. $\mathrm{dec}(i)$, $\mathrm{zero}(i)$] in u. It is easy to see that $\mathbf{val} \models \psi_{q_k}$ since all the values $\vec{x_0}$, …, $\vec{x_k}$ are in $\mathbb{N}^n$ and whenever $\vec{v}(i) \in \{\xrightarrow{0}_1, \xrightarrow{0}_2\}$, $\vec{x_k}(i) = 0$ by construction of δ'.
The proof in the other direction is analogous. QED

As a consequence, we obtain Theorem 4.1.2.

Now suppose that $\mathcal{S}$ is uniformly 1-reversal-bounded. This means that for every initial configuration $(q_0, \vec{x_0})$, the initialized counter automaton $(\mathcal{S}, (q_0, \vec{x_0}))$ is 1-reversal-bounded. In that case, we can show that the reachability relation is Presburger-definable.

Theorem 4.2.5. Let $\mathcal{S}$ be a uniformly 1-reversal-bounded counter automaton. For all $q, q' \in Q$, one can effectively compute a formula $\varphi_{q,q'}(\mathsf{x}_1, \ldots, \mathsf{x}_n, \mathsf{y}_1, \ldots, \mathsf{y}_n)$ such that for every valuation $\mathbf{val}$, we have

$$\mathbf{val} \models \varphi_{q,q'} \text{ iff } (q, (\mathbf{val}(\mathsf{x}_1), \ldots, \mathbf{val}(\mathsf{x}_n))) \xrightarrow{*} (q', (\mathbf{val}(\mathsf{y}_1), \ldots, \mathbf{val}(\mathsf{y}_n))).$$

Proof: The formula $\varphi_{q,q'}$ is defined below by taking advantage of the construction for reachability sets. Uniform reversal-boundedness ensures that the transition relation is uniformly defined. Nevertheless, the main differences with the previous developments are the following. First, instead of constant values $\vec{x_0}(1)$, …, $\vec{x_0}(n)$, we consider variables x_1, …, x_n and such a replacement in formulae can be done smoothly while respecting the syntax of formulae from Presburger arithmetic. Similarly, the value differences between counter values are precisely the differences between the number increments and the number of decrements, which can be easily expressed by a formula of the form $\mathsf{y}_i = \mathsf{x}_i + \mathsf{x}^i_{inc} - \mathsf{x}^i_{dec}$. Finally, we need to perform a case analysis on the value of the auxiliary vector $\vec{v_0} \in \{eq(0), neq(0)\}^n$, leading to simple constraints on the variables x_1, …, x_n. The formula $\varphi_{q,q'}$ is defined below:

$$\bigvee_{\vec{v_0}\in\{eq(0),neq(0)\}^n} \Big(\Big(\bigwedge_{i\in[1,n] \text{ s.t. } \vec{v_0}(i)=eq(0)} \mathsf{x}_i = 0\Big) \wedge \Big(\bigwedge_{i\in[1,n] \text{ s.t. } \vec{v_0}(i)=neq(0)} \mathsf{x}_i > 0\Big)\Big)\wedge$$

$$\bigvee_{\vec{v}\in S^{\vec{v_0}}} \exists \mathsf{x}^1_{inc}, \cdots, \mathsf{x}^n_{zero}\ (\varphi^{(q,\vec{v_0})}_{(q',\vec{v})}(\mathsf{x}^1_{inc}, \ldots, \mathsf{x}^n_{zero})$$

$$\wedge\Big(\bigwedge_{i\in[1,n] \text{ s.t. } \vec{v}(i)\in\{\xrightarrow{0}_1,\xrightarrow{0}_2\}} \mathsf{y}_i = 0\Big) \wedge \Big(\bigwedge_{i\in[1,n]} \mathsf{y}_i = \mathsf{x}_i + \mathsf{x}^i_{inc} - \mathsf{x}^i_{dec}\Big)$$

QED

As a corollary, we obtain the following result.

Corollary 4.2.6. [Iba78] Let $\mathcal{S} = (Q, n, \delta)$ be a counter automaton, $q, q' \in Q$, $\vec{x} \in \mathbb{N}$ and $r \geq 0$.

(I) If $\mathcal{S}$ is uniformly r-reversal-bounded, then one can effectively compute a formula φ such that for every valuation $\mathbf{val}$, we have $\mathbf{val} \models \varphi$ iff $(q, (\mathbf{val}(\mathsf{x}_1), \ldots, \mathbf{val}(\mathsf{x}_n))) \xrightarrow{*} (q', (\mathbf{val}(\mathsf{y}_1), \ldots, \mathbf{val}(\mathsf{y}_n)))$.

(II) If $(\mathcal{S}, (q, \vec{x}))$ is r-reversal-bounded, then one can effectively compute a Presburger formula φ such that for every valuation $\mathbf{val}$, we have $\mathbf{val} \models \varphi$ iff $(q, \vec{x}) \xrightarrow{*} (q', (\mathbf{val}(\mathsf{x}_1), \ldots, \mathbf{val}(\mathsf{x}_n)))$.

An alternative proof can be found in [LS05] that does not use Parikh's theorem. Complexity characterizations of reachability problems for reversal-bounded counter automata are presented in [GI81, HR87]. Moreover, other classes of counter systems with reachability sets that are effectively semilinear can be found in [HP79, Esp97, CJ98, FS00, LS05, BIL09] (see also Section 5.3).

Finally, the reachability problem with bounded number of reversals when natural numbers are encoded with a binary representation is NEXPTIME-complete [GI81, HR87] (the problem is NP-complete, assuming that all the natural numbers are encoded in binary except the number of reversals). Moreover, decidable reachability problems for parameterized reversal-bounded (init.) counter automata can be found in [ISD+02].

4.2.5 Variants admitting semilinearity too

In this section, we provide two generalizations of reversal-bounded counter automata for which reachability sets are still effectively semilinear.

Adding a free counter

An essential way to relax the notion of reversal-boundedness consists in allowing one counter to be free, i.e. no bounded number of reversals is required for that counter. If one wants to extend the previous results to this new class of counter automata, it is the best we can expect since two free counters already lead to undecidability because such a class would include Minsky machines. In the sequel, we assume that the free counter is the first one. So, an initialized counter automaton $(\mathcal{S}, (q, \vec{x}))$ is *almost reversal-bounded* $\overset{\text{def}}{\Leftrightarrow}$ $(\mathcal{S}, (q, \vec{x}))$ is reversal-bounded with respect to i, for $i \in [2, n]$ (assuming that the dimension of $\mathcal{S}$ is n). Uniform almost reversal-boundedness is defined in the obvious way.

Given that $(\mathcal{S}, (q_0, \vec{x_0}))$ is almost r-reversal-bounded, as done in Section 4.2.3, one can build an almost 1-reversal-bounded counter automaton $\mathcal{S}'$ such that Lemma 4.2.3 can be adapted to almost reversal-boundedness. Indeed, the first counter in $\mathcal{S}$ and $\mathcal{S}'$ behaves identically (no need to introduce additional counters) whereas the counters in $[2, n]$ from $\mathcal{S}$ are treated as in Section 4.2.3. It remains then to show that the reachability relation for uniformly almost 1-reversal-bounded counter automata is Presburger-definable, which can be shown as Theorem 4.2.5 by using that the

Parikh image of languages definable by finite-state automata equipped with a single counter is effectively semilinear. Indeed, such one-counter automata are pushdown automata for which the stack alphabet is simply unary and therefore Parikh's theorem applies [Par66].

Finally, this allows us to extend to almost reversal-bounded counter automata all the nice properties about semilinearity, as stated below.

Theorem 4.2.7. [Iba78] Let $\mathcal{S} = (Q, n, \delta)$ be a counter automaton, $q, q' \in Q$, $\vec{x} \in \mathbb{N}$ and $r \geq 0$.

(I) If $\mathcal{S}$ is uniformly almost r-reversal-bounded, then one can effectively compute a Presburger formula φ such that for every valuation $\mathbf{val}$, we have $\mathbf{val} \models \varphi$ iff $(q, (\mathbf{val}(\mathsf{x}_1), \ldots, \mathbf{val}(\mathsf{x}_n))) \xrightarrow{*} (q', (\mathbf{val}(\mathsf{y}_1), \ldots, \mathbf{val}(\mathsf{y}_n)))$.

(II) If $(\mathcal{S}, (q, \vec{x}))$ is almost r-reversal bounded, then one can effectively compute a Presburger formula φ such that for every valuation $\mathbf{val}$, we have $\mathbf{val} \models \varphi$ iff $(q, \vec{x}) \xrightarrow{*} (q', (\mathbf{val}(\mathsf{x}_1), \ldots, \mathbf{val}(\mathsf{x}_n)))$.

Weak reversal-boundedness

An interesting extension of reversal-boundedness is introduced in [FS08, San08] for which we only count the number of reversals when they occur for a counter value above a given bound B (see Figure 4.5). For instance, finiteness of the reachability set implies reversal-boundedness in the sense of [FS08, San08], which we shall call *weak reversal-boundedness*. Let $\mathcal{S} = (Q, n, \delta)$ be a (standard) counter automaton and a bound $B \in \mathbb{N}$. Instead of defining a counter automaton $\mathcal{S}_{rb}$ as done to characterize (standard) reversal-boundedness, we define directly an infinite directed graph that corresponds to a variant of the transition system of $\mathcal{S}_{rb}$: still, there are n new counters that record the number of reversals but only if they occur above a bound B. That is why, the infinite directed graph TS_B defined below is parameterized by B. $TS_B = (Q \times \{\mathrm{DEC}, \mathrm{INC}\}^n \times \mathbb{N}^{2n}, \rightarrow_B)$ is defined as follows: $(q, \vec{mode}, \vec{x}) \rightarrow_B (q', \vec{mode}', \vec{x}') \overset{\text{def}}{\Leftrightarrow}$ there is a transition $q \xrightarrow{\varphi} q' \in \delta$ such that

- $\star$ if φ does not deal with the jth component, then $\vec{mode}(j) = \vec{mode}'(j)$,
- $\star$ $(q, \vec{x}([1,n])) \xrightarrow{\varphi} (q', \vec{x}'([1,n]))$ in $\mathcal{S}$,
- $\star$ for every $i \in [1,n]$, one of the conditions below is satisfied:
  - $*$ $\varphi = \mathrm{zero}(i)$, $\vec{mode}(i) = \vec{mode}'(i)$, $\vec{x} = \vec{x}'$,
  - $*$ $\varphi = \mathrm{dec}(i)$, $\vec{mode}(i) = \vec{mode}'(i) = \mathrm{DEC}$ and $\vec{x}([n+1, 2n]) = \vec{x}'([n+1, 2n])$,
  - $*$ $\varphi = \mathrm{dec}(i)$, $\vec{mode}(i) = \mathrm{INC}$, $\vec{mode}'(i) = \mathrm{DEC}$, $\vec{x}(i) > B$ and $\vec{x}([n+1, 2n] \setminus \{i\}) = \vec{x}'([n+1, 2n] \setminus \{i\})$, $\vec{x}'(i) = \vec{x}(i) + 1$,
  - $*$ $\varphi = \mathrm{dec}(i)$, $\vec{mode}(i) = \mathrm{INC}$, $\vec{mode}'(i) = \mathrm{DEC}$, $\vec{x}(i) \leq B$ and $\vec{x}([n+1, 2n]) = \vec{x}'([n+1, 2n])$,
  - $*$ $\varphi = \mathrm{inc}(i)$, $\vec{mode}(i) = \mathrm{DEC}$, $\vec{mode}'(i) = \mathrm{INC}$, $\vec{x}(i) > B$ and $\vec{x}([n+1, 2n] \setminus \{i\}) = \vec{x}'([n+1, 2n] \setminus \{i\})$, $\vec{x}'(n+i) = \vec{x}(n+i) + 1$,
  - $*$ $\varphi = \mathrm{inc}(i)$, $\vec{mode}(i) = \mathrm{DEC}$, $\vec{mode}'(i) = \mathrm{INC}$, $\vec{x}(i) \leq B$ and $\vec{x}([n+1, 2n]) = \vec{x}'([n+1, 2n])$.

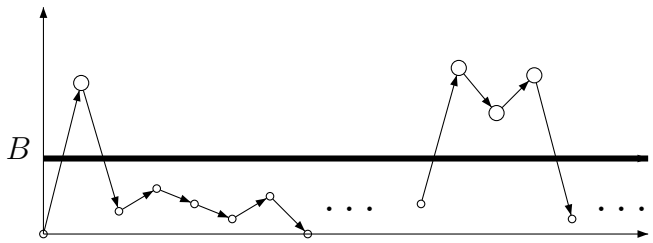


Figure 4.5: A counter satisfying weak reversal-boundedness

Initialized counter automaton $(\mathcal{S}, (q, \vec{x}))$ is *weakly reversal-bounded* [FS08] $\stackrel{\text{def}}{\Leftrightarrow}$ there is some $B \geq 0$ such that for $i \in [n+1, 2n]$, $\{\vec{y}(i) : (q_{rb}, \vec{x}_{rb}) \xrightarrow{*}_B (q', \vec{y}) \text{ in } TS_B\}$ is finite. When $r \geq \max(\{\vec{y}(i) : (q_{rb}, \vec{x}_{rb}) \xrightarrow{*}_B (q', \vec{y}) \text{ in } TS_B\} : i \in [n+1, 2n])$ $\mathcal{S}$ is said to be *r-reversal-B-bounded* from $(q, \vec{x})$. Observe that whenever $(\mathcal{S}, (q, \vec{x}))$ is r-reversal-bounded, $(\mathcal{S}, (q, \vec{x}))$ is r-reversal-0-bounded. Figure 4.5 illustrates weak reversal-boundedness. Reversal-boundedness for counter automata is very appealing because reachability sets are semilinear as stated below.

Theorem 4.2.8. [FS08, San08] Let $(\mathcal{S}, (q, \vec{x}))$ be an initialized counter automaton that is weakly r-reversal-B-bounded for some $r, B \geq 0$. For each control state q', the set $\{\vec{y} \in \mathbb{N}^n : \text{run } (q, \vec{x}) \xrightarrow{*} (q', \vec{y})\}$ is effectively semilinear.

The proof in [Iba78] extends to weak reversal-boundedness [FS08]; whenever a counter value is below B, this information is encoded in the control state which provides a reduction to (standard) reversal-boundedness.

Moreover, a breakthrough has been done in [FS08] by establishing that checking whether a vector addition systems with states is weakly reversal-bounded is decidable. The decidability proof in [FS08] provides a decision procedure that requires nonprimitive recursive time in the worst-case since Karp and Miller tree needs to be built [KM69, VVN81]. A complexity analysis can be found in [Dem10]. Besides, further material about reversal-bounded counter automata can be found in [San08, Chapter 2].

4.3 Decidable repeated reachability problems

In this section, we show how to reduce the control state repeated reachability problem to the reachability problem when reversal-bounded counter automata are involved. Clearly, reversal-boundedness is taken into account and we assume that reversal-bounded counter automata are given with their maximal number of reversals r.

Lemma 4.3.1. [DIP01] Control state repeated reachability problem in the class of reversal-bounded (initialized) counter automata is decidable.

Proof: Let $(\mathcal{S}, (q_0, \vec{x_0}))$ be an initialized counter automaton that is r-reversal-bounded with $\mathcal{S} = (Q, n, \delta)$ and $q_f \in Q$ be the control state to be repeated infinitely often.

We propose an algorithm to answer the following question: is there an infinite run starting at $(q_0, \vec{x_0})$ such that the control state q_f is repeated infinitely often? We reduce it to a reachability

question for a new reversal-bounded counter automaton $\mathcal{S}'$. Furthermore, we know that for each control state, one can effectively compute a Presburger formula that represents the configurations that can reach this control state, leading to decidability since satisfiability problem for Presburger arithmetic is decidable.

Let ($\star$) be the desired property:

($\star$) There is an infinite run from $(q_0, \vec{x_0})$ such that q_f is repeated infinitely often.

Let ($\star\star$) be the property below:

($\star\star$) There exist a finite run $\rho = (q_0, \vec{x_0}) \xrightarrow{t_1} (q_1, \vec{x_1}) \cdots \xrightarrow{t_l} (q_l, \vec{x_l})$, $l' \in [0, l-1]$ and $Z \subseteq [1, n]$ such that

(a) $q_l = q_{l'} = q_f$,
(b) for $i \in Z$ and $j \in [l'+1, l]$, $\vec{x_j}(i) - \vec{x_{j-1}}(i) = 0$,
(c) for $i \in ([1, n] \setminus Z)$, we have $\vec{x_{l'}}(i) \leq \vec{x_l}(i)$,
(d) for $i \in ([1, n] \setminus Z)$ and $j \in [l'+1, l]$, $\vec{x_j}(i) - \vec{x_{j-1}}(i) \geq 0$,
(e) for $i \in ([1, n] \setminus Z)$, $\vec{x_{l'}}(i) \geq 1$.

Observe that (d) implies (c).

Below, we show that ($\star$) and ($\star\star$) are equivalent, which allows us to reduce control state repeated reachability to control state reachability problem. Indeed, checking ($\star\star$) amounts to introduce $\mathrm{card}(\mathcal{P}([1, n]))$ copies of $\mathcal{S}$ (one for each possible set $Z \subseteq [1, n]$).

First, let us show that ($\star$) and ($\star\star$) are equivalent. Suppose ($\star$). There exist an infinite run $\rho = (q_0, \vec{x_0}) \xrightarrow{t_1} (q_1, \vec{x_1}) \xrightarrow{t_2} (q_2, \vec{x_2}) \cdots$ such that q_f is repeated infinitely often. Let $CST(\rho)$ be the subset of $[1, n]$ that contains exactly the counters that are constant in ρ, apart from a finite prefix. Since $(\mathcal{S}, (q_0, \vec{x_0}))$ is reversal-bounded, there exists $I \geq 0$ such that for $k \geq I$, no counter in $[1, n] \setminus Z$ is decremented and its value is greater than 1 and all the counters in $CST(\rho)$ remains constant. Since q_f is repeated infinitely often, there are $I \leq l < l'$ such that $q_l = q_{l'} = q_f$ and (b)-(e) hold true. Now suppose that there exist a finite run $\rho = (q_0, \vec{x_0}) \xrightarrow{t_1} (q_1, \vec{x_1}) \cdots \xrightarrow{t_l} (q_l, \vec{x_l})$, $l' \in [0, l-1]$ and $Z \subseteq [1, n]$ witnessing the satisfaction of ($\star\star$). It is then easy to show that the ω-sequence of transitions $t_1 \cdots t_{l'}(t_{l'+1} \cdots t_l)^\omega$ allows us to define an infinite run ρ' that extends ρ. It is clear that in ρ' the control state q_f is repeated infinitely often. Zero-tests are also successful because of condition (b).

Now, let us build an instance of the reachability problem for reversal-bounded counter automata that allows us to capture the condition ($\star\star$). We construct a reversal-bounded counter automaton $\mathcal{S}' = (Q', n, \delta')$ such that ($\star\star$) iff $(q_0, \vec{x_0}) \xrightarrow{*} (q_{new}, \vec{0})$ in $\mathcal{S}'$. By Theorem 4.1.2, one can effectively build a Presburger formula φ with n free variables such that $\mathrm{REL}(\varphi) = \{\vec{x} : (q_0, \vec{x_0}) \xrightarrow{*} (q_{new}, \vec{x})\}$. By decidability of Presburger arithmetic, we can therefore decide whether there is an infinite run starting at $(q_0, \vec{x_0})$ in which q_f is repeated infinitely often.

It remains to define the counter automaton $\mathcal{S}'$. The counter automaton $\mathcal{S}'$ is made of the original version of $\mathcal{S}$ (called below the *original copy*) augmented with 2^n copies of $\mathcal{S}$; each copy corresponds to a possible set $Z \subseteq [1, n]$ in ($\star\star$). By the *Z-copy*, we mean the restriction of $\mathcal{S}$ such that:

$\star$ no transition in the Z-copy modifies a counter from Z,

$\star$ no transition in the Z-copy decrements a counter in $[1,n] \setminus Z$.

For each $Z \subseteq [1,n]$, the control states of the Z-copy are pairs in $Q \times \{Z\}$. The second component simply indicates to which copy belongs the control state.

In order to simulate the subrun $(q_{l'}, \vec{x_{l'}}) \cdots (q_l, \vec{x_l})$ for the satisfaction of $(\star\star)$ in $\mathcal{S}$, nondeterministically we move from the original copy to some Z-copy in $\mathcal{S}'$ (and therefore we choose which counters remain constants). To do so, for every set $Z \subseteq [1,n]$, we consider in $\mathcal{S}'$ a sequence of transitions from q_f to (q_f, Z) whose task is to check that for $i \in [1,n] \setminus Z$, we have $\mathsf{x}_i \geq 1$ (which can be done by decrementing and incrementing counter i, inducing at most n reversals).

As soon as in the Z-copy, we reach again a control state whose first component is q_f, we may jump to the final control state q_{new}. QED

Even though the problem below is decidable (as shown above), as far as we know its computational complexity is open.

Input: a succinct counter automaton $\mathcal{S}$, a bound $r \in \mathbb{N}$, an initial configuration $(q, \vec{x})$ and a control state q_f.

Question: Is there an infinite run from $(q, \vec{x})$ such that q_f is repeated infinitely often and each counter has at most r reversals?

Lemma 4.3.1 can be extended so that, instead of repeating infinitely often control states, properties on counters definable in Presburger arithmetic are repeated infinitely often. Let us introduce the following problem.

∃-PRESBURGER INFINITELY OFTEN PROBLEM

Input: Initialized counter automaton $(\mathcal{S}, (q, \vec{x}))$ of dimension n that is r-reversal-bounded and a temporal formula of the form $\psi = \mathtt{GF}\varphi(\mathsf{x}_1, \ldots, \mathsf{x}_n)$ where φ is a Presburger formula on counters.

Question: Is there an infinite run from $(q, \vec{x})$ satisfying ψ?

The complement of the above problem is defined as follows. The ∀-PRESBURGER-ALMOST-ALWAYS PROBLEM is defined analogously:

Input: Initialized counter system $(\mathcal{S}, (q, \vec{x}))$ of dimension n that is r-reversal-bounded and a temporal formula of the form $\psi = \mathtt{FG}\varphi(\mathsf{x}_1, \ldots, \mathsf{x}_n)$ where φ is a Presburger formula on counters.

Question: Is it the case that every infinite run from $(q, \vec{x})$ satisfies ψ?

Theorem 4.3.2. [DPK03] The ∃-Presburger infinitely often problem and the ∀-Presburger-almost-always problem are decidable for reversal-bounded (initialized) counter automata.

The proof is indeed a generalization of the proof of Lemma 4.3.1. As will shown in the sequel, the combination of quantifications over runs and positions on runs is essential to get decidability. The design of maximal logical fragments retaining decidability is still open.

4.4 Undecidable reachability problems

Despite the content of the previous sections, many decision problems for reversal-bounded counter automata are undecidable, even if the number of counters is bounded. For instance, a counter system with alphabet is naturally defined as a counter system except that transitions are labelled by letters from a finite alphabet Σ, and sets of initial and final control states are considered. This allows us to define languages from counter systems with alphabet (subsets of Σ^*), as done for finite-state automata (without counters). The *universal problem* consists in checking whether the language defined by a counter system is Σ^*. It is known that the problem is PSPACE-complete for finite-state automata. However, the same problem for 1-reversal-bounded one-counter automata already leads to undecidability.

Theorem 4.4.1. [Iba79] The universal problem for 1-reversal-bounded one-counter automata with alphabet is undecidable.

It is worth noting that one-counter automata with alphabet form a subclass of pushdown systems and therefore accept context-free languages.

In the rest of this section, we shall present two decision problems related to properties definable in temporal logics that are undecidable for reversal-counter automata.

4.4.1 A simple temporal fragment leading to undecidability

In this section, we consider the following problem. The $\exists$-PRESBURGER-ALWAYS PROBLEM is defined as follows:

Input: Initialized counter automaton $(\mathcal{S}, (q, \vec{x}))$ of dimension n that is r-reversal-bounded and a temporal formula of the form $\psi = \mathtt{G}\varphi(\mathsf{x}_1, \ldots, \mathsf{x}_n)$ where φ is a Presburger formula on counters.

Question: Is there an infinite run from $(q, \vec{x})$ satisfying ψ?

Theorem 4.4.2. [DPK03] The $\exists$-Presburger-always problem for reversal-bounded counter automata is undecidable.

As is shown in the proof below, we can even restrict ourselves to 0-reversal-bounded counter automata without zero-tests and with a fixed number of counters (a subclass of VASS).

Proof: The proof is analogous to the undecidability of the reachability problem for reversal-bounded counter automata augmented with guards of the form $\mathsf{x}_i = \mathsf{x}_{i'}$ and $\mathsf{x}_i \neq \mathsf{x}_{i'}$ [ISD$^+$02].

Let $\mathcal{S} = (Q, 2, \delta)$ be a (deterministic) Minsky machine with a unique halting control state q_h without outgoing transitions. We shall build a 0-reversal-bounded counter automaton $\mathcal{S}'$ such that for each counter $i \in \{1, 2\}$ from $\mathcal{S}$, two increasing counters i and $i + 2$ are considered in $\mathcal{S}'$; i records the number of increments and $i + 2$ records the number of decrements. Zero-test for counter i is performed by a simple test $\mathsf{x}_i = \mathsf{x}_{i+2}$. This encoding is indeed the main idea of the proof.

For a technical reason in the proof below, it is helpful to know whether a configuration has been obtained immediately after performing a zero-test on counter $i \in \{1, 2\}$. To do do, we can

slightly modify $\mathcal{S}$ so that performing a zero-test on i can be detected by reaching a control state in the subset Q''_i. In particular, it is not possible to reach a control state in Q''_i if a zero-test on i has not been performed just before. So before defining $\mathcal{S}'$, let us observe that it is easy to define a counter automaton $\mathcal{S}'' = (Q'', 2, \delta'')$ from $\mathcal{S}$, that behaves as $\mathcal{S}$ except that exactly the control states in $Q''_1 \subseteq Q''$ [resp. $Q''_2 \subseteq Q''$] can be reached after a zero-test on counter 1 [resp. on counter 2]. Additionnally, the control states in $Q''_1 \cup Q''_2$ cannot be reached after a decrementation or an incrementation. Hence, each control state in $\mathcal{S}$ may lead to at most three control states in $\mathcal{S}''$. Finally, $\mathcal{S}''$ may have more than one halting control state (and less than four).

Hence, without any loss of generality, we can assume that $\mathcal{S} = (Q, 2, \delta)$ is a deterministic counter automaton with halting control states in $Q_h \subseteq Q$ and for which there are subsets $Q_1, Q_2 \subseteq Q$ containing exactly the control states that are reached after zero-tests. Moreover, from a control state without outgoing transitions that increment some counter, one can either perform a zero-test on some counter i or decrement the counter i. This type of constraints comes from the definition of deterministic Minsky machines.

Let us now build a 0-reversal-bounded counter automaton $\mathcal{S}' = (Q', 5, \delta')$ as follows:

- $\star$ $Q' = Q$.
- $\star$ For $q \xrightarrow{\mathrm{inc}(i)} q' \in \delta$, we have $q \xrightarrow{\mathrm{inc}(i)} q' \in \delta'$.
- $\star$ For $q \xrightarrow{\mathrm{dec}(i)} q' \in \delta$, we have $q \xrightarrow{\mathrm{inc}(i+2)} q' \in \delta'$.
- $\star$ For $q \xrightarrow{\mathrm{zero}(i)} q' \in \delta$, we have $q \xrightarrow{\mathrm{inc}(5)} q' \in \delta'$. The only reason to introduce counter 5 is to perform a dummy instruction that does not involve the four first counters.

Given an initial configuration $(q, \vec{0})$, it is possible to show that no halting control state is reached from $(q, \vec{0})$ in $\mathcal{S}$ iff there is an infinite run from $(q, \vec{0})$ in $\mathcal{S}'$ satisfying the formula φ below:

$$\mathtt{G}(\overbrace{\bigwedge_{i\in\{1,2\}} \bigwedge_{q\in Q_i} (q \Rightarrow \mathtt{x}_i = \mathtt{x}_{i+2})}^{\text{simulation of zero-tests}})\wedge$$

$$\mathtt{G}(\overbrace{\bigwedge_{i\in\{1,2\}} \mathtt{x}_i \geq \mathtt{x}_{i+2}}^{\text{no negative counter values}}) \wedge \mathtt{G}(\overbrace{\bigwedge_{q\in Q_h} \neg q}^{\text{no halting state reached}})$$

It remains to show how we can get rid of atomic formulae made of control states. Suppose that $Q = \{q_1, \ldots, q_N\}$. We update the definition of $\mathcal{S}'$ by adding 4 counters such that the atomic formula q_j above can be replaced by the Presburger formula $(\mathtt{x}_7 - \mathtt{x}_6 = j \wedge \mathtt{x}_9 - \mathtt{x}_8 = j)$, which is then of the required form to get undecidability of the $\exists$-Presburger-always problem. The new set of control states Q' includes Q with auxiliary control states that are described below. The definition of the transitions in δ' is updated as follows.

- $\star$ For $q_j \xrightarrow{\mathrm{inc}(i)} q_{j'} \in \delta$ with $j \leq j'$, we consider the following sequence of transitions:

$$q_j \xrightarrow{\mathrm{inc}(7)} q_j^1 \xrightarrow{\mathrm{inc}(7)} q_j^2 \cdots q_j^{(j'-j)} \xrightarrow{\mathrm{inc}(i)} q_j^{(j'-j)+1} \overbrace{\xrightarrow{\mathrm{inc}(9)} \cdots \xrightarrow{\mathrm{inc}(9)}}^{(j'-j)\text{ incrementations of counter 9}} q_{j'}$$

All the control states above that are not in Q are auxiliary and are used only for a unique transition from $\mathcal{S}$. Observe that the only control state for which a configuration can satisfy $(\mathsf{x}_7 - \mathsf{x}_6 = j' \wedge \mathsf{x}_9 - \mathsf{x}_8 = j')$ is precisely $q_{j'}$.

- ⋆ For $q_j \xrightarrow{\mathrm{inc}(i)} q_{j'} \in \delta$ with $j > j'$, we consider the following sequence of transitions:

$$q_j \xrightarrow{\mathrm{inc}(6)} q_j^1 \xrightarrow{\mathrm{inc}(6)} q_j^2 \cdots q_j^{(j-j')} \xrightarrow{\mathrm{inc}(i)} q_j^{(j-j')+1} \overbrace{\xrightarrow{\mathrm{inc}(8)} \cdots \xrightarrow{\mathrm{inc}(8)}}^{(j-j') \text{ incrementations of counter } 8} q_{j'}$$

- ⋆ Transitions from δ of the form either $q_j \xrightarrow{\mathrm{dec}(i)} q_{j'}$ or $q_j \xrightarrow{\mathrm{zero}(i)} q_{j'}$ admit a similar treatement.

So, given an initial configuration $(q, \vec{0})$, no halting control state is reached from $(q, \vec{0})$ in $\mathcal{S}$ iff there is an infinite run from $(q, \vec{0})$ in $\mathcal{S}'$ satisfying φ above in which each q_j is replaced by $(\mathsf{x}_7 - \mathsf{x}_6 = j \wedge \mathsf{x}_9 - \mathsf{x}_8 = j)$. If we allow a succinct version of counter automata (updates can be arbitrary integers), we do not need to consider 4 new counters (2 should suffice). Indeed, we do not have to bother about intermediate configurations that have no counterpart for $\mathcal{S}$. QED

4.4.2 Freeze LTL and reversal-bounded VASS

In this section, we show another undecidability result whose proof uses the same idea of encoding one counter by two increasing counters. Moreover, only equality tests are allowed at the level of atomic formulae but we allow a restricted use of the freeze operator (only 1 register is allowed in formulae). By contrast, in the proof of Theorem 4.4.2 constraints of the form either $\mathsf{x}_i \geq \mathsf{x}_{i+2}$ or $\mathsf{x}_7 - \mathsf{x}_6 = \mathsf{x}_9 - \mathsf{x}_8$ are used.

Theorem 4.4.3. [DS10] Model-checking problem $\mathrm{MC}^{\omega}(\mathrm{LTL}^{\downarrow})$ restricted to 1 register and without control states is undecidable for reversal-bounded VASS.

Proof: The proof has similarities with the proof of Theorem 4.4.2 in the way the counters are encoded. Let $\mathcal{S} = (Q, 2, \delta)$ be a deterministic counter automaton with halting control states in $Q_h \subseteq Q$ and for which there are subsets $Q_1, Q_2 \subseteq Q$ containing exactly the control states that are reached after zero-tests. Moreover, from a control state without outgoing transitions that increment some counter, one can either perform a zero-test on some counter i or decrement the counter i.

Again, we shall build a 0-reversal-bounded counter automaton $\mathcal{S}'$ such that each counter i in $\mathcal{S}$ is simulated by two increasing counters i and $i+2$. Moreover, $\mathcal{S}'$ is without zero-test, so it is indeed a VASS. Unlike the proof of Theorem 4.4.2, zero-test for counter i is performed by the formula $\downarrow_1^i \uparrow_1^{i+2}$. We recall that the atomic formula $\uparrow_1^j$ is a shortcut for $\mathsf{y}_1 = \mathsf{x}_j$ and a formula of the form $\downarrow_1^j \chi$ is a shortcut for $\exists\, \mathsf{y}_1\ (\mathsf{y}_1 = \mathsf{x}_j \wedge \chi)$. So, $\downarrow_1^i \uparrow_1^{i+2}$ corresponds literally to $\exists\, \mathsf{y}_1\ (\mathsf{y}_1 = \mathsf{x}_i \wedge \mathsf{y}_1 = \mathsf{x}_{i+2})$, which is logically equivalent to $\mathsf{x}_i = \mathsf{x}_{i+2}$. Let us build a 0-reversal-bounded counter automaton $\mathcal{S}' = (Q', 5, \delta')$ as follows:

- ⋆ $Q' = Q$.
- ⋆ For $q \xrightarrow{\mathrm{inc}(i)} q' \in \delta$, we have $q \xrightarrow{\mathrm{inc}(i)} q' \in \delta'$.

- ⋆ For $q \xrightarrow{\mathrm{dec}(i)} q' \in \delta$, we have $q \xrightarrow{\mathrm{inc}(i+2)} q' \in \delta'$.
- ⋆ For $q \xrightarrow{\mathrm{zero}(i)} q' \in \delta$, we have $q \xrightarrow{\mathrm{inc}(5)} q' \in \delta'$.

Given an initial configuration $(q, \vec{0})$, it is possible to show that no halting control state is reached from $(q, \vec{0})$ in $\mathcal{S}$ iff there is an infinite run from $(q, \vec{0})$ in $\mathcal{S}'$ satisfying the formula φ below:

$$\mathtt{G}(\overbrace{\bigwedge_{i\in\{1,2\}} \bigwedge_{q\in Q_i} (q \Rightarrow \downarrow_1^i \uparrow_1^{i+2})}^{\text{simulation of zero-tests}}) \wedge$$

$$\mathtt{G}(\overbrace{\bigwedge_{q\xrightarrow{\mathrm{dec}(i)}q'\in\delta'} ((q \wedge \mathtt{X}q') \Rightarrow \downarrow_1^i \neg \uparrow_1^{i+2})}^{\text{no negative counter values}}) \wedge \mathtt{G}(\overbrace{\bigwedge_{q\in Q_h} \neg q}^{\text{no halting state reached}})$$

It remains to show how we can get rid of atomic formulae made of control states. Suppose that $Q = \{q_1, \ldots, q_N\}$. We update the definition of $\mathcal{S}'$ by adding $2N$ counters such that the atomic formula q_j above can be replaced by a formula stating that (1) the value of counter $5 + 2j - 1$ (the first 5 counters are already booked for another purpose) is different from the value of counter $5+2j$ and (2) for $j' \neq j$, the value of counter $5+2j'-1$ is equal to value of counter $5+2j'$. Below, we write α_j^+ to denote $5 + 2j - 1$ and α_j^- to denote $5 + 2j$. In the following, we shall enforce that when the value for counter α_j^+ is different from the value for counter α_j^-, then their difference is exactly one, the counter α_j^+ having the greater value. Let ψ_j be defined below expressing (1) and (2):

$$\downarrow_1^{\alpha_j^+} \neg \uparrow_1^{\alpha_j^-} \wedge \bigwedge_{j'\neq j} \downarrow_1^{\alpha_{j'}^+} \uparrow_1^{\alpha_{j'}^-}$$

The new set of control states Q' includes Q plus auxiliary control states that are described below.

- ⋆ For $q_j \xrightarrow{\mathrm{inc}(i)} q_{j'} \in \delta$ we consider the following sequence of transitions:

$$q_j \xrightarrow{\mathrm{inc}(\alpha_j^-)} q_{j,j'}^1 \xrightarrow{\mathrm{inc}(i)} q_{j,j'}^2 \xrightarrow{\mathrm{inc}(\alpha_{j'}^+)} q_{j'}$$

  When $j = j'$, we just need to include $q_j \xrightarrow{\mathrm{inc}(i)} q_j$ in δ'.

- ⋆ For $q_j \xrightarrow{\mathrm{dec}(i)} q_{j'} \in \delta$ we consider the following sequence of transitions:

$$q_j \xrightarrow{\mathrm{inc}(\alpha_j^-)} q_{j,j'}^1 \xrightarrow{\mathrm{inc}(i+2)} q_{j,j'}^2 \xrightarrow{\mathrm{inc}(\alpha_{j'}^+)} q_{j'}$$

  When $j = j'$, we just need to include $q_j \xrightarrow{\mathrm{inc}(i+2)} q_j$ in δ'.

- ⋆ For $q_j \xrightarrow{\mathrm{zero}(i)} q_{j'} \in \delta$ we consider the following sequence of transitions:

$$q_j \xrightarrow{\mathrm{inc}(\alpha_j^-)} q_{j,j'}^1 \xrightarrow{\mathrm{inc}(5)} q_{j,j'}^2 \xrightarrow{\mathrm{inc}(\alpha_{j'}^+)} q_{j'}$$

  When $j = j'$, we just need to include $q_j \xrightarrow{\mathrm{inc}(5)} q_j$ in δ'.

So, given an initial configuration $(q_j, \vec{0})$, no halting control state is reached from $(q, \vec{0})$ in $\mathcal{S}$ iff there is an infinite run from $(q_j, \vec{y_j})$ in $\mathcal{S}'$ satisfying φ above in which each q_j is replaced by ψ_j. The initial counter values in $\vec{y_j}$ are defined as follows:

- $\star$ For $i \in \{1, \ldots, 5\}$, $\vec{y_j}(i) = 0$.
- $\star$ For $j' \in (\{1, \ldots, N\} \setminus \{j\})$, $\vec{y_j}(\alpha^+_{j'}) = \vec{y_j}(\alpha^-_{j'}) = 0$.
- $\star$ $\vec{y_j}(\alpha^+_j) = 1$ and $\vec{y_j}(\alpha^-_j) = 0$.

QED

It is open whether the above proof still works when the number of counters in the VASS is bounded. Indeed, the undecidability proof uses an unbounded number of counters in VASS.

4.5 Exercises

Exercise 4.5.1. Let us consider the reversal-bounded counter automaton in Figure 4.2.

1. Is $(\mathcal{S}, (q_1, \vec{0}))$ reversal-bounded?
2. For which q, every $(\mathcal{S}, (q, \vec{x}))$ is reversal-bounded?
3. Let $\vec{x} \in \mathbb{N}^2$ and φ be the Presburger formula

$$\varphi = (\mathsf{x}_1 \geq 2 \wedge \mathsf{x}_2 \geq 1 + \vec{x}(2) \wedge (\mathsf{x}_2 - \vec{x}(2)) + 1 \geq \mathsf{x}_1) \vee$$

$$(\mathsf{x}_2 \geq 2 \wedge \mathsf{x}_1 \geq 1 + \vec{x}(1) \wedge (\mathsf{x}_1 - \vec{x}(1)) + 1 \geq \mathsf{x}_2)$$

   Show that $\mathrm{REL}(\varphi)$ is equal to $\{\vec{y} \in \mathbb{N}^2 : (q_1, \vec{x}) \xrightarrow{*} (q_9, \vec{y})\}$.
4. Find a Presburger formula φ' such that $\mathrm{REL}(\varphi') = \{\vec{y} \in \mathbb{N}^2 : (q_1, \vec{0}) \xrightarrow{*} (q_6, \vec{y})\}$.
5. Show that for every q, $\{\vec{x} \in \mathbb{N}^2 : (\mathcal{S}, (q, \vec{x}))$ is $\mathrm{RB}\}$ is semilinear.

Exercise 4.5.2. Complete the proof of Theorem 4.2.1 where (♯) appears.

Exercise 4.5.3. Provide the proof of Lemma 4.2.2

Exercise 4.5.4. Show that the statement of Theorem 4.4.3 can be refined by imposing that the freeze operator is used in the strict way, that is each register is associated with a unique counter (see a solution in [DS10]).

Exercise 4.5.5. Prove Theorem 4.2.8

Exercise 4.5.6. Let us extend the class of standard counter automata by allowing transitions labelled by equality tests of the form $\mathsf{x}_i = k$? for some $k \in \mathbb{N}$; this generalizes the usual zero-tests. The notion of r-reversal-boundedness can be defined accordingly. Are the reachability sets of r-reversal-bounded initialized enriched standard counter automata effectively semilinear?

Exercise 4.5.7. A counter automaton $\mathcal{S}$ is uniformly reversal-bounded (Version 2) iff for every initial configuration $(q, \vec{x})$, the initialized counter automaton $(\mathcal{S}, (q, \vec{x}))$ is reversal-bounded. What about the semilinearity of the reachability relation?

Exercise 4.5.8. A set $X \subseteq \mathbb{N}^n$ is upward closed $\stackrel{\text{def}}{\Leftrightarrow}$ for all $\vec{x}, \vec{y} \in \mathbb{N}^n$, $\vec{x} \in X$ and $\vec{x} \preceq \vec{y}$ imply $\vec{y} \in X$.

1. Given a VASS $\mathcal{V} = (Q, n, \delta)$ and $q \in Q$, show that the set $\{\vec{x} \in \mathbb{N}^n : (\mathcal{V}, (q, \vec{x})) \text{ is not RB}\}$ is upward closed.
2. Show that the set $\{\vec{x} \in \mathbb{N}^n : (\mathcal{V}, (q, \vec{x})) \text{ is RB}\}$ is semilinear.
3. Suggest a proof strategy to show that the above set is effectively semilinear, i.e. one can effectively compute a Presburger formula φ from $\mathcal{V}$ and q such that $\mathrm{REL}(\varphi) = \{\vec{x} \in \mathbb{N}^n : (\mathcal{V}, (q, \vec{x})) \text{ is RB}\}$.

Exercise 4.5.9.

Question 4.5.9.1 Given $B \geq 0$ and $\vec{x} \in \mathbb{N}^n$, we define the *B-truncation* of $\vec{x}$, written $\mathtt{trunc}_B(\vec{x})$, as a tuple in $\mathbb{N}^n$ such that for $i \in [1, n]$, we have $\mathtt{trunc}_B(\vec{x})(i) \stackrel{\text{def}}{=} min(\vec{x}(i), B)$. A set $X \subseteq \mathbb{N}^n$ is said to be *simple* $\stackrel{\text{def}}{\Leftrightarrow}$ there are $B \geq 0$ and $Y \subseteq [0, B]^n$ such that for every $\vec{x} \in \mathbb{N}^n$, $\vec{x} \in X$ iff $\mathtt{trunc}_B(\vec{x}) \in Y$. A *simple guard* φ is defined as a Presburger formula respecting the grammar below:

$$\mathsf{x}_i \geq k \mid \mathsf{x}_i \leq k \mid \varphi_1 \wedge \varphi_2 \mid \top$$

with $k \in \mathbb{N}$, x_i is a variable interpreted by a natural number in $\mathbb{N}$ and $\top$ is the truth constant. Let φ be a simple guard with free variables among $\{\mathsf{x}_1, \ldots, \mathsf{x}_n\}$. Show that $\mathrm{REL}(\varphi)$ is a simple set, i.e. φ can be associated with a pair (B, Y) encoding $\mathrm{REL}(\varphi)$.

Question 4.5.9.2 An *extended counter automata* $\mathcal{S}$ of dimension n is a counter system of dimension n in which the transitions are represented in the following way:

$$t = q \xrightarrow{(\varphi(\mathsf{x}_1, \ldots, \mathsf{x}_n), \vec{b})} q'$$

where $\varphi(\mathsf{x}_1, \ldots, \mathsf{x}_n)$ is a simple guard with free variables among $\{\mathsf{x}_1, \ldots, \mathsf{x}_n\}$ and $\vec{b} \in \mathbb{Z}^n$ (update vector). Given configurations $(q, \vec{a}), (q', \vec{a'}) \in Q \times \mathbb{N}^n$, by definition $(q, \vec{a}) \xrightarrow{t} (q', \vec{a'}) \stackrel{\text{def}}{\Leftrightarrow} \vec{a} \models \varphi(\mathsf{x}_1, \ldots, \mathsf{x}_n)$ and $\vec{a'} = \vec{a} + \vec{b}$. Reversal-boundedness for extended counter automata is defined as for standard counter automata: initialized extended counter automaton $(\mathcal{S}, (q, \vec{x}))$ is r-reversal-bounded $\stackrel{\text{def}}{\Leftrightarrow}$ for every run from $(q, \vec{x})$, every counter performs at most r reversals.

Let $(\mathcal{S}, (q_0, \vec{x_0}))$ be a reversal-bounded extended counter automata and B_{max} be the maximal bound from all the bounds B associated to simple guards in $\mathcal{S}$. Let $(q_0, \vec{x_0}), (q_1, \vec{x_1}), \ldots$ be an infinite run for the extended counter automaton $\mathcal{S}$ such that the control state q_f is repeated infinitely often. Show that there are positions $l' < l$ and a set of counters $Z \subseteq [1, n]$ such that:

(a) $q_l = q_{l'} = q_f$,
(b) for $i \in Z$ and $j \in [l'+1, l]$, $\vec{x_j}(i) - \vec{x_{j-1}}(i) = 0$,
(c) for $i \in ([1,n] \setminus Z)$, we have $\vec{x_{l'}}(i) \leq \vec{x_l}(i)$,
(d) for $i \in ([1,n] \setminus Z)$ and $j \in [l'+1, l]$, $\vec{x_j}(i) - \vec{x_{j-1}}(i) \geq 0$,
(e) for $i \in ([1,n] \setminus Z)$, $\vec{x_{l'}}(i) \geq B_{max}$.

Observe that (d) implies (c).

Question 4.5.9.3 Show that there is an infinite run from $(q_0, \vec{x_0})$ with control state q_f repeated infinitely often iff there are a finite run $(q_0, \vec{x_0}), (q_1, \vec{x_1}), \ldots, (q_l, \vec{x_l})$, $l' < l$ and $Z \subseteq [1,n]$ such that (a)–(e) hold true.

Question 4.5.9.4 Define a reversal-bounded extended counter automaton $\mathcal{S}'$ such that there is an infinite run from $(q_0, \vec{x_0})$ with q_f repeated infinitely often in $\mathcal{S}$ iff $(q_0, \vec{x_0}) \xrightarrow{*} (q_{new}, \vec{0})$ in $\mathcal{S}'$ (q_{new} is a new control state occuring in $\mathcal{S}'$ but not in $\mathcal{S}$).

Chapter 5

Model-Checking Counter Systems

In this chapter, we mainly focus on repeated control-state reachability problem (i.e. existing of infinite runs with Büchi acceptance condition) and LTL model-checking when the atomic formulae are simply control states. So, in this chapter, plain LTL is understood as the fragment of $\mathrm{LTL}^{\mathrm{CS}}(\mathrm{PrA})$ in which the atomic formulae are restricted to control states (so there is no need to use first-order quantification), i.e., $\mathrm{LTL} \approx \mathrm{LTL}(\mathtt{Q})$. The control state repeated reachability problem is mainly considered for VASS, reversal-bounded counter automata, and lossy counter automata; sometimes we summarize developments from previous chapters. In the second part of this chapter, we show that existential model-checking for $\mathrm{LTL}^{\mathrm{CS}}(\mathrm{PrA})$ for admissible counter systems is decidable by reduction to the satisfiability problem for Presburger arithmetic. We provide an almost complete proof for this result since we believe the proof technique is interesting for its own sake. Admissible counter systems are defined as a subclass of counter systems with affine updates with a condition imposing that the effect of loops is effectively semilinear.

5.1 When LTL model-checking is equivalent to repeated control state reachability

Let $\mathcal{S}$ be a counter system, $(q_0, \vec{x_0})$ be an initial configuration and φ be an LTL formula; the atomic formulae are restricted to control states. If the run ρ starting by $(q_0, \vec{x_0})$ satisfies $\rho, 0 \models \varphi$, then one can easily show that $proj_Q(\rho), 0 \models \varphi$, where $proj_Q(\rho) \in Q^\omega$ is obtained from ρ by erasing the counter values. Consequently, by construction of the Büchi automaton $\mathcal{A}_\varphi$ with alphabet made of singleton subsets of Q, there is a successful run of the form $\rho' = X_0 \xrightarrow{proj_Q(\rho)(0)} X_1 \xrightarrow{proj_Q(\rho)(1)} X_2 \xrightarrow{proj_Q(\rho)(2)} X_3 \cdots$. The satisfaction of $\rho, 0 \models \varphi$ and $proj_Q(\rho), 0 \models \varphi$ can be represented by the two synchronized sequences below:

$$\begin{array}{ccccccccccc} (q_0, \vec{x_0}) & \rightarrow & (q_1, \vec{x_1}) & \rightarrow & (q_2, \vec{x_2}) & \rightarrow & (q_3, \vec{x_3}) & \rightarrow & \cdots & \models \varphi \\ X_0 & \xrightarrow{q_0} & X_1 & \xrightarrow{q_1} & X_2 & \xrightarrow{q_2} & X_3 & \xrightarrow{q_3} & & \models \varphi \end{array}$$

The definition of synchronized product below is motivated by the design of a unique counter system synchronizing $\mathcal{S}$ and $\mathcal{A}_\varphi$ with control states of the form (q_i, X_i) and updating the counter values according to the transitions from $\mathcal{S}$. Acceptance condition from $\mathcal{A}_\varphi$ shall be naturally

expressed by infinite repetition of control states whose second component is a final state from $\mathcal{A}_\varphi$.

Definition 5.1.1. [Synchronized product] Let $\mathcal{S} = (Q, n, \delta)$ be a counter system and $\mathcal{A} = (\Sigma, Q', Q'_0, \delta', F)$ be a Büchi automaton with alphabet $\Sigma = Q$. The *synchronized product* $\mathcal{S} \otimes \mathcal{A}$ is a counter system (Q'', n'', δ'') such that

- $Q'' = Q \times Q'$,
- $n'' = n$,
- $(q_0, q'_0) \xrightarrow{\varphi} (q_1, q'_1) \overset{\text{def}}{\Leftrightarrow} q_0 \xrightarrow{\varphi} q_1 \in \delta$ and $q'_0 \xrightarrow{q_0} q'_1 \in \delta'$.

∇

Lemma 5.1.1. Let $\mathcal{S} = (Q, n, \delta)$ be a counter system, $(q, \vec{x})$ be a configuration and φ be an LTL formula built over the control states in $\mathcal{S}$. Let $\mathcal{A}_\varphi = (\Sigma, Q', Q'_0, \delta', F)$ be the Büchi automaton such that Models(φ) $= \mathrm{L}(\mathcal{A}_\varphi)$ and $\Sigma = Q$. The propositions below are equivalent:

(I) There is an infinite run ρ with initial configuration $(q, \vec{x})$ such that $\rho, 0 \models \varphi$.

(II) For some $q_i \in Q'_0$ and $(q'', q_f) \in Q \times F$, there is an infinite run in $\mathcal{S} \otimes \mathcal{A}_\varphi$ with initial configuration $((q, q_i), \vec{x})$ such that the control state (q'', q_f) is repeated infinitely often.

Proof: Left as an exercise. QED

Consequently, an instance of the LTL model-checking problem can be solved by checking several instances of the control state repeated reachability problem.

Theorem 5.1.2. Let C be a class of counter systems such that

1. the control state repeated reachability problem is decidable,
2. C is closed under synchronized products (with Büchi automata).

Then, the LTL model-checking problem restricted to counter systems in C is decidable.

In terms of computational complexity, in the worst-case we may observe an exponential blow-up since the number of control states in $\mathcal{S} \otimes \mathcal{A}_\varphi$ can be exponential in the size of $\mathcal{S}$ and φ. Complexity results for LTL model-checking problems for infinite-state systems can be also found in [TL10].

Proof: Let $\mathcal{S}$, $(q, \vec{x})$ and φ be an instance the the LTL model-checking problem. The Büchi automaton $\mathcal{A}_\varphi$ can be effectively computed from φ and there is an infinite run ρ with initial configuration $(q, \vec{x})$ such that $\rho, 0 \models \varphi$ iff for some $q_i \in Q'_0$ and $(q'', q_f) \in Q \times F$, there is an infinite run in $\mathcal{S} \otimes \mathcal{A}_\varphi$ with initial configuration $((q, q_i), \vec{x})$ such that (q'', q_f) is repeated infinitely often (see Lemma 5.1.1). Since both Q'_0 and $Q \times F$ are finite sets, the existence of a finite run ρ such that $\rho, 0 \models \varphi$ can be verified by checking at most $\mathrm{card}(Q'_0) \times \mathrm{card}(Q \times F)$ instances of the control state repeated reachability problem on the system $\mathcal{S} \otimes \mathcal{A}_\varphi$. By (2.), such a system belongs also to C and the target problem is decidable by (1.). QED

5.2 Control State Repeated Reachability Problem

In this section, we review the decidability status of the control state repeated reachability problems for several classes of counter systems, mainly those introduced earlier.

5.2.1 VASS

Control state repeated reachability problem restricted to VASS has been shown in [Jan90] whereas the exponential space upper bound has been established in [Hab97] by adapting Rackoff's proof for solving the boundedness problem for VASS in exponential space.

Lemma 5.2.1. [Hab97] Control state repeated reachability problem restricted to VASS can be solved in exponential space.

The proof is by adapting Rackoff's technique, it is necessary to establish a property with a witness path: there is an infinite run with initial configuration $(q, \vec{x})$ such that the control state q_f is repeated infinitely often iff there is a finite run $(q_0, \vec{x_0}), \ldots, (q_k, \vec{x_k})$ such that

- $(q_0, \vec{x_0}) = (q, \vec{x})$,
- there is $k' < k$ such that $\vec{x_{k'}} \preceq \vec{x_k}$,
- $q_k = q_{k'} = q_f$.

This is obtained by using Dickson's Lemma [Dic13]: for any ω-sequence $\vec{x_0}, \vec{x_1}, \ldots$ of tuples in $\mathbb{N}^n$, there are $i < j$ such that $\vec{x_i} \preceq \vec{x_j}$. The key argument to get the EXPSPACE upper bound is to show that k can be at most double-exponential in the size of the instance $\mathcal{S}$, $(q, \vec{x})$, q'.

This allows to show the following result (the proof technique developped in [Rac78] has been also used to establish the result below).

Theorem 5.2.2. [Hab97] LTL model-checking problem for VASS is EXPSPACE-complete.

Let us conclude this section by presenting a fragment of $\mathrm{LTL}^{\mathrm{CS}}(\mathrm{PrA})$ introduced in [Jan90] such that the atomic formulae are either control states or atomic formulae of the form $\mathsf{x}_i \geq c$ or $\neg(\mathsf{x}_i \geq c)$ with $c \in \mathbb{N}$. Atomic formulae are therefore richer than those of LTL but as will be presented, temporal and Boolean operators are restricted.

The temporal logic with fairness TLF is defined as a logic on VASS for which formulae are defined by the grammar below:

$$q \mid \mathsf{x}_i \geq c \mid \neg(\mathsf{x}_i \geq c) \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \mathtt{GF}\varphi$$

where $q \in \mathtt{Q}$ and $c \in \mathbb{N}$. Observe that TLF formulae are not closed under negations and the temporal properties are intersection or union of fairness conditions.

Theorem 5.2.3. [Jan90] Existential model-checking problem fo TLF restricted to VASS is decidable.

In [Jan90], decidability is shown by reduction into the reachability problem for VASS. The proof is quite difficult and one of its interests is to reduce the existence of an infinite run to a reachability question. Fairness conditions on VASS can be also found in [GS92].

Moreover, it is worth noting that the operator F cannot be expressed in TLF, otherwise undecidability would hold. Indeed, in [HR89] a linear-time temporal logic (on Petri nets) is shown undecidable with the temporal operator F, Boolean connectives and atomic formulae of the form $\mathsf{x}_i \geq c$ and "transition t is the next one in the run". Other decidability and undecidability results for linear-time temporal logic on Petri nets can be found in [Esp94]; for instance linear μ-calculus with propositions $\mathsf{x}_i = 0$ is undecidable.

5.2.2 Reversal-bounded counter automata

We have seen in Chapter 4 that the control state repeated reachability problem is decidable for reversal-bounded (initialized) counter automata. Consequently, we obtain the following result since the class of reversal-bounded counter automata is closed under synchronized product with Büchi automata.

Theorem 5.2.4. LTL model-checking problem for reversal-bounded (initialized) counter automata is decidable.

It is worth observing that a stronger result is shown in [DIP01] since Presburger-definable atomic properties can be included while preserving decidability.

5.2.3 Imperfect counter automata

In this section, we shall consider variants of counter automata in which counter values can be decremented without notification (a loss) or counter values can be incremented without notification (a gain) – but not the two possibilities in the same model. A similar model is the class of reset VASS that has been defined in Section 1.4.3.

Theorem 5.2.5. Control state repeated reachability for reset VASS is undecidable.

By contrast, control state reachability problem for reset VASS is decidable as a consequence of the decidability of the covering problem for reset Petri nets [DFS98]. However, the problem has a nonprimitive recursive complexity, as a consequence of [Sch02]. Many results on reset VASS can be found in [DFS98, DJS99], see also the recent survey [Sch10b].

Lossy counter automata

A *lossy counter automaton* is standard counter automaton such that for $q \in Q$ and $i \in [1, n]$, $q \xrightarrow{\mathrm{dec}(i)} q$ (which allows us to simulate losses).

Theorem 5.2.6. The control state reachability problem for lossy counter automata is decidable.

It is worth noting that lossy counter automata form a subclass of lossy channel systems, see e.g. [Sch02] and the reachability problem for lossy channel systems is decidable [AJ96, FS01]. For instance, they can be used to model lossy channel systems for which the ordering of the messages is not relevant. In that case, each counter can store how many messages of a given type are present in the channel. Lossy counter automata have been introduced in [May03] and a survey paper on recent developments can be found in [Sch10b]. Besides, a logic with temporal operator EF for lossy VASS has been shown to admit a decidable model-checking problem in [BM99].

Gainy counter automata

Let us shift to the model with gains. A *gainy counter automaton* is a standard counter automaton (Q, n, δ) such that for $q \in Q$ and $i \in [1, n]$, $q \xrightarrow{\mathrm{inc}(i)} q \in \delta$ (which allows us to simulate gains). In the sequel, we shall not represent these transitions. Instead, we consider that the one-step derivation relation is modified as follows: $(q, \vec{x}) \xrightarrow{t}_g (q', \vec{x'})$ iff there are $\vec{y}$ and $\vec{y'}$ in $\mathbb{N}^n$ such that $\vec{x} \preceq \vec{y}$, $(q, \vec{y}) \xrightarrow{t} (q', \vec{y'})$ (exact step) and $\vec{y'} \preceq \vec{x'}$.

From a gainy counter automaton $\mathcal{S}$, one can effectively compute in logarithmic space a reset VASS $\mathcal{S}'$ such that runs of $\mathcal{S}$ are precisely reverse runs in $\mathcal{S}'$, whence the control state reachability problem for gainy counter automata can be reduced to the analogous problem for reset VASS.

Corollary 5.2.7. The control state reachability problem for gainy counter automata is decidable.

Theorem 5.2.8. [Sch02, Sch10c] (see also [Urq99][1]) The control state reachability problem for gainy counter automata is nonprimitive recursive.

Lemma 5.2.9. [DL06] The control state repeated reachability problem restricted to gainy counter automata is undecidable.

The proof is by adapting the proof for undecidability of the recurrence problem for Insertion Channel Machines with Emptiness-Testing (ICMET) [OW06]. Non-reachability of a control state q in Minsky machine $\mathcal{S}$ can be reduced to control state repeated reachability problem for gainy counter automata.

Proof: Let $\mathcal{S} = (Q, 2, \delta)$ be a (deterministic) Minsky machine with a special control state q_h from which no transition goes out (and this is the only dead-end control state). We have seen that it is undecidable whether there is a run reaching a configuration with control state q_h from the initial configuration $(q_i, \vec{0})$.

First, we build a counter automaton $\mathcal{S}' = (Q', 3, \delta')$ that behaves exactly as $\mathcal{S}$ as far as the counters 1 and 2 are concerned. However, the counter 3 is incremented after each instruction of $\mathcal{S}$. Consequently, the control state q_h cannot be reached in $\mathcal{S}$ iff for the unique run of $\mathcal{S}'$, the counter 3 has no bounded value.

Second, we build a gainy counter automaton $\mathcal{S}''$ with 6 counters:

- ⋆ The counters 1, 2 and 3 roughly behave as the three respective counters in $\mathcal{S}'$.
- ⋆ The counter 4 is the global budget that is progressively incremented.

[1] Thanks to M. Praveen (IMSc, Chennai) for pointing me to this work.

- ⋆ The counter 5 is the current budget that records how many increments on one of the counters 1, 2 or 3 can be still performed. For instance, an increment of counter 3 is followed by a decrement of counter 5.

- ⋆ The counter 6 is an auxiliary counter that is mainly instrumental to perform a copy from counter 4 to counter 5.

Figure 5.1 contains the schematic construction of $\mathcal{S}''$. The instruction $\mathrm{copy}(4,5)$ that copies the content of counter 4 into counter 5 (with possible gains) can be performed thanks to the gadget described on the left of Figure 5.2. Similarly, the instruction $\mathrm{transfer}(1+2+3,5)$ that transfers the content of the counters 1, 2 and 3 to the counter 5 (with possible gains) can be performed thanks to the gadget described on the right of Figure 5.2. We have used shortcuts in some places but it is easy to see that $\mathcal{S}''$ can be defined as a gainy counter automaton.

It remains to explain the part of the simulation of $\mathcal{S}'$ (see middle of Figure 5.1). Below $i \in \{1,2,3\}$.

- ⋆ A transition $q \xrightarrow{\mathrm{dec}(i)} q'$ is simulated by two transitions $q \xrightarrow{\mathrm{dec}(i)} \circ \xrightarrow{\mathrm{inc}(5)} q'$. The location $\circ$ is an arbitrary new location only used to simulate this transition.

- ⋆ A transition $q \xrightarrow{\mathrm{zero}(i)} q'$ is simulated by itself.

- ⋆ A transition $q \xrightarrow{\mathrm{inc}(i)} q'$ is simulated by three transitions: $q \xrightarrow{\mathrm{inc}(i)} \circ$, $\circ \xrightarrow{\mathrm{dec}(5)} q'$ and $\circ \xrightarrow{\mathrm{zero}(5)} \mathrm{MO}$ (memory overflow). This last transition is represented in Figure 5.1 by a transition entering in the control state MO.

One can show that Minsky machine $\mathcal{S}$ cannot reach q_h iff $(\mathcal{S}'', \vec{0})$ has a run that visits infinitely often the control state (1). The Minsky machine $\mathcal{S}$ cannot reach q_h iff the counter automaton $\mathcal{S}'$ cannot reach q_h. If $\mathcal{S}'$ cannot reach q_h, then an error-free run of $\mathcal{S}''$ visits infinitely often (1). For the converse direction we use the following facts:

- ⋆ In (A), the only way to decrement counter 5 is to simulate exactly $\mathcal{S}'$.

- ⋆ In order to reach (1), in the part between q_i and (A), counter 5 is decremented regularly.

- ⋆ If $\mathcal{S}''$ visits infinitely often (1) and $\mathcal{S}'$ can reach $(q_h, \vec{x})$, then at some point an error-free simulation of $\mathcal{S}'$ shall be done with value for counter 5 greater than $\vec{x}(1)+\vec{x}(2)+\vec{x}(3)$, a contradiction.

QED

As a corollary (using Lemma 5.1.1) the get the following negative result:

Corollary 5.2.10. LTL model-checking problem restricted to gainy counter automata is undecidable.

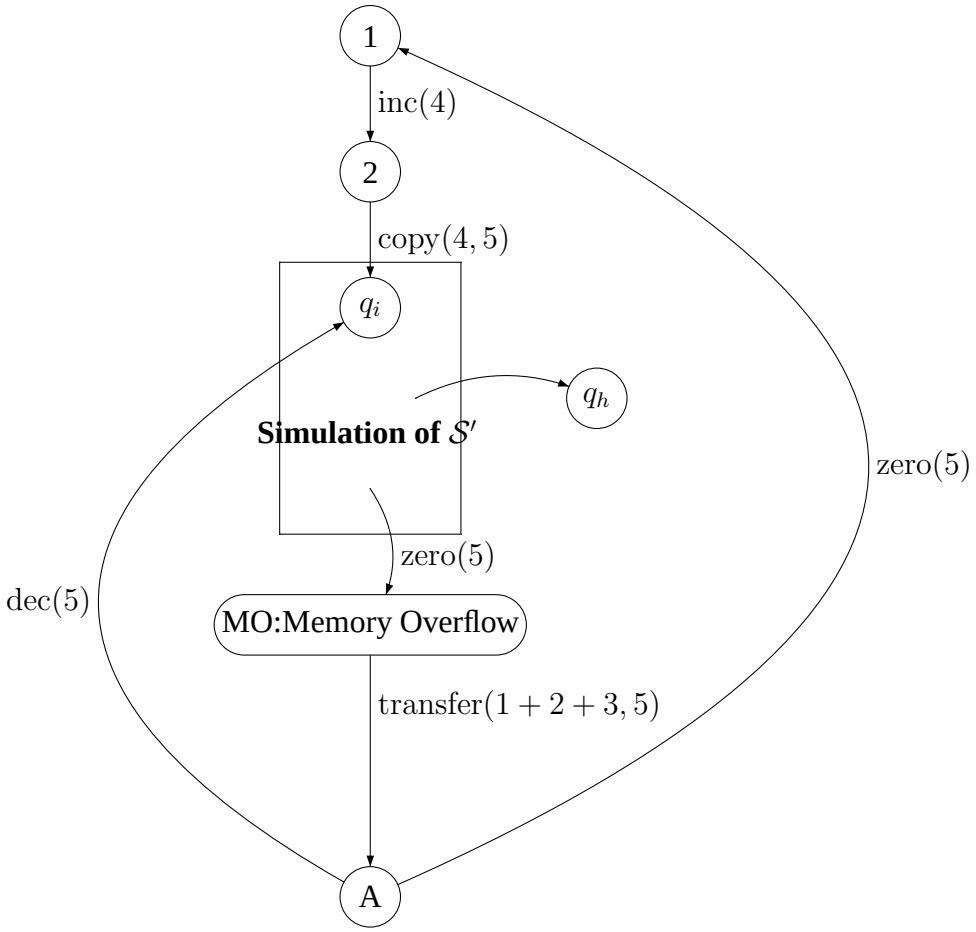
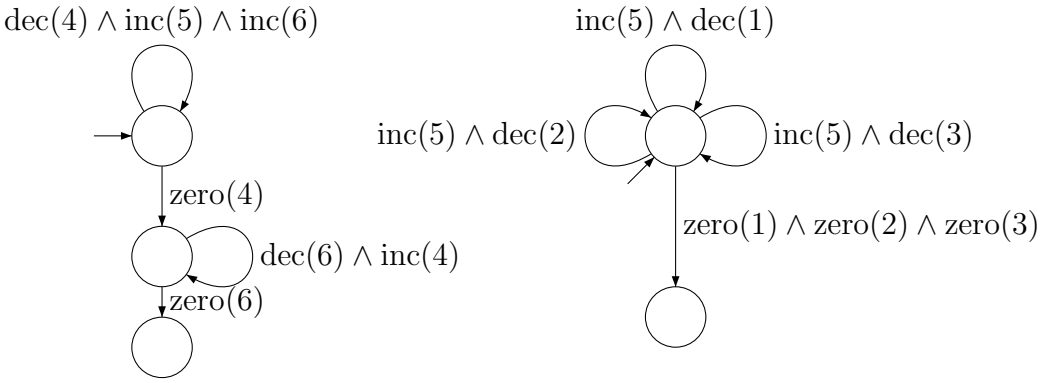


Figure 5.1: Gainy counter automaton $\mathcal{S}''$



A gadget to transfer the counters 1, 2 and 3 into counter 5

A gadget to copy counter 4 into counter 5

Figure 5.2: Gadgets

5.3 Admissible Counter Systems

In this section, we introduce another class of counter systems for which we show that the accessibility relation is effectively semilinear (we have already seen a detailed proof for the class of reversal-bounded –initialized– counter automata in Chapter 4). This class is not comparable with the class of flat relational counter systems for which semilinearity is a consequence of [CJ98] and we provide below a proof for effective semilinearity based on [FL02, Ler03]. Moreover, not only this implies that the reachability problem for admissible counter systems is decidable but we shall show that the model-checking problem for $\mathrm{LTL}^{\mathrm{CS}}(\mathrm{PrA})$ restricted to admissible counter systems is decidable too. This is obtained by reduction to satisfiability for Presburger arithmetic. The class of admissible counter systems is defined by restricting both the control graph (flatness) and the class of Presburger formulae labelling transitions (those defining affine functions).

5.3.1 Affine counter systems

In this section, we shall define the class of *affine counter systems* that slightly generalizes the class of succinct counter automata (roughly speaking, a counter value can be multiplied by a factor different from 1). To do so, we start by proposing a few definitions.

A *binary relation of dimension* n is a relation $R \subseteq \mathbb{N}^{2n}$. R is *Presburger definable* $\stackrel{\text{def}}{\Leftrightarrow}$ there is a Presburger formula $\varphi(\mathsf{x}_1, \ldots, \mathsf{x}_n, \mathsf{x}'_1, \ldots, \mathsf{x}'_n)$ with $2n$ free variables such that $R = \mathrm{REL}(\varphi)$. A partial function f from $\mathbb{N}^n$ to $\mathbb{N}^n$ is *affine* $\stackrel{\text{def}}{\Leftrightarrow}$ there exist a matrix $A \in \mathbb{Z}^{n\times n}$ and $\vec{b} \in \mathbb{Z}^n$ such that for every $\vec{a} \in \mathrm{dom}(f)$, we have $f(\vec{a}) = A\vec{a} + \vec{b}$. f is *Presburger definable* $\stackrel{\text{def}}{\Leftrightarrow}$ the graph of f is a Presburger definable relation.

A counter system $\mathcal{S} = (Q, n, \delta)$ is *affine* when for every transition $q \xrightarrow{\varphi} q' \in \delta$, $\mathrm{REL}(\varphi)$ is affine. In the sequel, each formula φ labelling a transition in an affine counter system is encoded by a triple $(A, \vec{b}, \psi)$ such that

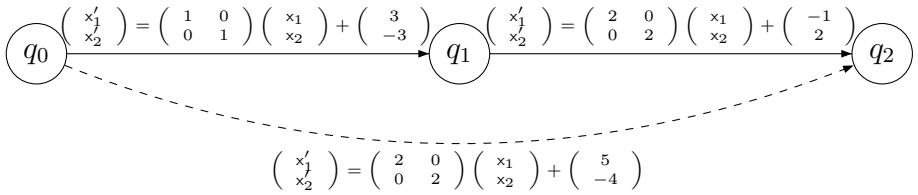
1. $A \in \mathbb{Z}^{n \times n}$,
2. $\vec{b} \in \mathbb{Z}^n$,
3. ψ has free variables $\mathsf{x}_1, \ldots, \mathsf{x}_n$,
4. $\mathrm{REL}(\varphi) = \{(\vec{x}, \vec{x'}) \in \mathbb{N}^{2n} : \vec{x'} = A\vec{x} + \vec{b} \text{ and } \vec{x} \in \mathrm{REL}(\psi)\}$.

The formula ψ can be viewed as the guard of the transition and the pair $(A, \vec{b})$ as the (deterministic) update function. Such a triple $(A, \vec{b}, \psi)$ is called an *affine update* and we also write $\mathrm{REL}((A, \vec{b}, \psi))$ to denote $\mathrm{REL}(\varphi)$. Observe that one can decidable whether a Presburger formula φ satisfies that $\mathrm{REL}(\varphi)$ is affine [DFGvD06, DFGvD11]. Furthermore, succinct counter automata are affine counter systems in which the matrices are always equal to the identity matrix. Moreover, in succinct counter automata the guards are reduced to the truth constant or to a zero-test. This class of counter systems has been introduced in [FL02].

Observe that assuming the transition $t = q \xrightarrow{(A,\vec{b},\psi)} q'$, there is a Presburger formula $\chi(\vec{\mathsf{x}}, \vec{\mathsf{x}'})$ such that for every $\mathbf{val}$, we have $\mathbf{val} \models \chi$ iff $(q, (\mathbf{val}(\mathsf{x}_1), \ldots, \mathbf{val}(\mathsf{x}_n))) \xrightarrow{t} (q', (\mathbf{val}(\mathsf{x}'_1), \ldots, \mathbf{val}(\mathsf{x}'_n)))$. Here is the witness formula that encodes the one-step relation:

$$\psi(\vec{\mathsf{x}}) \wedge \bigwedge_{i \in [1,n]} (\mathsf{x}'_i = \sum_j A(i,j)\mathsf{x}_j + \vec{b}(i))$$

Affine updates are closed under composition as illustrated below:



Lemma 5.3.1 roughly states that the composition of affine updates is still an affine update, which shall be helpful to show that the accessibility relation for admissible counter systems is Presburger definable.

Lemma 5.3.1. Let $(A_1, \vec{b_1}, \psi_1)$ and $(A_2, \vec{b_2}, \psi_2)$ be two affine updates. There exists an affine update $(A, \vec{b}, \psi)$ such that

$$\mathrm{REL}((A, \vec{b}, \psi)) =$$
$$\{(\vec{x}, \vec{x'}) \in \mathbb{N}^{2n} : \exists \vec{y} \in \mathbb{N}^n \ (\vec{x}, \vec{y}) \in \mathrm{REL}((A_1, \vec{b_1}, \psi_1)) \text{ and } (\vec{y}, \vec{x'}) \in \mathrm{REL}((A_2, \vec{b_2}, \psi_2))\}$$

Proof: Consider the partial map $f_i : \mathbb{N}^n \to \mathbb{N}^n$ such that

$$\{(\vec{x}, \vec{x'}) \in \mathbb{N}^{2n} : \vec{x} \in \mathrm{REL}(\psi_i),\ \vec{x'} = A_i\vec{x} + \vec{b_i}\}$$

We have that $\mathrm{REL}((A, \vec{b}, \psi))$ is equal to

$$\{(\vec{x}, \vec{x'}) \in \mathbb{N}^{2n} : \exists \vec{y} \in \mathbb{N}^n\ f_1(\vec{x}) = \vec{y},\ \vec{x} \in \mathrm{dom}(f_1),\ f_2(\vec{y}) = \vec{x'},\ \vec{y} \in \mathrm{dom}(f_2)\}$$

Now, the condition "$\exists \vec{y} \in \mathbb{N}^n\ f_1(\vec{x}) = \vec{y},\ \vec{x} \in \mathrm{dom}(f_1),\ f_2(\vec{y}) = \vec{x'},\ \vec{y} \in \mathrm{dom}(f_2)$" is equivalent to the conditions:

1. $\vec{x'} = A_2A_1\vec{x} + A_2\vec{b_1} + \vec{b_2}$,
2. $\vec{x} \in \mathrm{REL}(\psi_1)$,
3. $A_1\vec{x} + \vec{b_1} \in \mathrm{REL}(\psi_2)$.

So, it is easy to see that the triple $(A, \vec{b}, \psi)$ below satisfies the requirements:

- $\star$ $A = A_2A_1$,
- $\star$ $\vec{b} = A_2\vec{b_1} + \vec{b_2}$,
- $\star$ $\psi = \exists\, \vec{\mathsf{y}}\ \psi_1(\vec{\mathsf{x}}) \wedge \vec{\mathsf{y}} = A_1\vec{\mathsf{x}} + \vec{b_1} \wedge \psi_2(\vec{\mathsf{y}})$, where $\vec{\mathsf{x}} = (\mathsf{x}_1, \ldots, \mathsf{x}_n)$, $\vec{\mathsf{y}} = (\mathsf{y}_1, \ldots, \mathsf{y}_n)$ and $\vec{\mathsf{y}} = A_1\vec{\mathsf{x}} + \vec{b_1}$ is a shortcut for a conjunction made of n conjuncts. Indeed, assuming that $A_1 = (a_{i,j})_{(i,j)\in[1,n]^2}$, each conjunct is of the form $\mathsf{y}_i = \sum_j a_{i,j}\mathsf{x}_j + \vec{b_1}(i)$.

QED

5.3.2 Loop effects

In the forthcoming class of admissible counter systems, we shall assume that the control graph is flat. Hence, it becomes essential to represent symbolically the effect of loops on counter values. Anyhow, this sounds as a necessary condition to establish that a reachability relation is semilinear. We already know by Lemma 5.3.1 that transitions in affine counter systems are closed under bounded compositions.

Let R be a binary relation of dimension n. The *reflexive and transitive closure* of R, written R^*, is a subset of $\mathbb{N}^{2n}$ such that $(\vec{y}, \vec{y'}) \in R^*$ iff there are $\vec{x_1}, \ldots \vec{x_k} \in \mathbb{N}^n$ such that

- $\star$ $\vec{x_1} = \vec{y}$,
- $\star$ $\vec{x_k} = \vec{y'}$,
- $\star$ for $i \in [1, k-1]$, we have $(\vec{x_i}, \vec{x_{i+1}}) \in R$.

If R is Presburger definable, then this does not imply that R^* is Presburger definable too. For instance, if $R = \{(\alpha, 2\alpha) \in \mathbb{N}^2 : \alpha \in \mathbb{N}\}$ then $R^* = \{(\alpha, 2^\beta\alpha) \in \mathbb{N}^2 : \alpha, \beta \in \mathbb{N}\}$ is not Presburger definable. By contrast, if $S = \{(\alpha, \alpha+1) \in \mathbb{N}^2 : \alpha \in \mathbb{N}\}$ then $S^* = \{(\alpha, \beta) \in \mathbb{N}^2 : \alpha < \beta,\ \alpha, \beta \in \mathbb{N}\}$ is Presburger definable. The question of deciding whether the reflexive and transitive closure of a Presburger definable binary relation is Presburger definable is intimately related to the fact that accessibility relations from counter systems are Presburger definable, which leads to decidability when effectiveness is guaranteed too.

Indeed, consider the following loop with $q_1 = q_k$:

$$q_1 \xrightarrow{\varphi_1(\mathsf{x}_1,\ldots,\mathsf{x}'_n)} q_2 \xrightarrow{\varphi_2(\mathsf{x}_1,\ldots,\mathsf{x}'_n)} \cdots \xrightarrow{\varphi_{k-1}(\mathsf{x}_1,\ldots,\mathsf{x}'_n)} q_{k-1} \xrightarrow{\varphi_k(\mathsf{x}_1,\ldots,\mathsf{x}'_n)} q_k.$$

The effect of the loop can be represented by the Presburger formula below:

$$\psi(\vec{\mathsf{x}_1}, \vec{\mathsf{x}'}) \stackrel{\text{def}}{=} \exists\, \vec{\mathsf{y}_1}, \ldots, \vec{\mathsf{y}_k}\ \varphi_1(\vec{\mathsf{x}}, \vec{\mathsf{y}_1}) \wedge \varphi_2(\vec{\mathsf{y}_1}, \vec{\mathsf{y}_2}) \wedge \cdots \wedge \varphi_k(\vec{\mathsf{y}_k}, \vec{\mathsf{x}'})$$

In order to decide the reachability problem on the loop, it is essential to represent symbolically the set $\{(\vec{x}, \vec{x'}) \in \mathbb{N}^{2n} : (q_1, \vec{x}) \to (q_2, \vec{y_1}) \cdots \to (q_k, \vec{x'})\}$. The best we can hope for is that this set is Presburger definable. This motivates the definition below.

Definition 5.3.1. Given a binary relation $R \subseteq \mathbb{N}^{2n}$, we define the *counting iteration of* R as the relation $R_{\mathbf{CI}} \subseteq \mathbb{N}^n \times \mathbb{N} \times \mathbb{N}^n$ such that $(\vec{a}, i, \vec{b}) \in R_{\mathbf{CI}} \overset{\text{def}}{\Leftrightarrow} (\vec{a}, \vec{b}) \in R^i$. R has a ***Presburger counting iteration*** if its counting iteration is Presburger definable. ∇

If R has a Presburger counting iteration, then there exists a Presburger formula $\chi(\vec{\mathsf{x}}, \mathsf{z}, \vec{\mathsf{y}})$ such that $\mathrm{REL}(\chi) = R_{\mathbf{CI}}$. Consequently, the relation R^* is Presburger definable since $\mathrm{REL}(\exists\, \mathsf{z}\ \chi) = R^*$. Definition 5.3.1 is precisely the concept we need to show that the model-checking for $\mathrm{LTL}^{\mathrm{CS}}(\mathrm{PrA})$ is decidable for admissible counter systems (see Theorem 5.3.5). Observe that $\{(\alpha, \alpha+1) \in \mathbb{N}^2 : \alpha \in \mathbb{N}\}$ has a Presburger counter iteration witnessed by a Presburger formula of the form $\mathsf{x}' = \mathsf{x} + \mathsf{y}$.

Given $A \in \mathbb{Z}^{n \times n}$, we write A^* to denote the monoid generated from A with $A^* = \{A^i : i \in \mathbb{N}\}$. The identity element is naturally the identity matrix $A^0 = I$. Given a matrix $A \in \mathbb{Z}^{n \times n}$, checking whether the monoid generated by A is finite, is decidable [MS77].

By way of example, with $A = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$, we have

$$A^2 = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix}\ A^3 = \begin{pmatrix} 1 & 0 \\ 3 & 1 \end{pmatrix}\ \ldots A^m = \begin{pmatrix} 1 & 0 \\ m & 1 \end{pmatrix}$$

So A does not have the finite monoid property. Finiteness of the monoid generated from A is interesting because of the lemma below.

Lemma 5.3.2. [Boi98, FL02] Let R be a binary relation of dimension n defined by the triple $(A, \vec{b}, \psi)$ such that $R = \{(\vec{x}, \vec{x'}) \in \mathbb{N}^{2n} : \vec{x'} = A\vec{x} + \vec{b} \text{ and } \vec{x} \in \mathrm{REL}(\psi)\}$. If A^* is finite, then R has a Presburger counting iteration.

It is worth adding that one can also effectively compute the Presburger formula encoding the relation R^*, which is exactly what is done in the proof below.

Proof: Let R be a binary relation of dimension n defined by the triple $(A, \vec{b}, \psi)$. We write g to denote the total map from $\mathbb{Z}^n$ to $\mathbb{Z}^n$ such that $g(\vec{a}) = A\vec{a} + \vec{b}$.

Since A^* is finite, there are $\alpha, \beta \in \mathbb{N}$ such that $A^{\alpha+\beta} = A^\alpha$. By [MS77], α and β can be effectively computed from A and below these values are therefore constants. The following equalities are easy to show ($k \geq 1$):

- $g^k(\vec{a}) = A^k\vec{a} + A^{k-1}\vec{b} + \cdots + \vec{b}$ (shown by an easy induction on k).
- $g^k(\vec{0}) = A^{k-1}\vec{b} + \cdots + \vec{b}$.

Before going any further, let us fix some notations about tuples of terms from Presburger arithmetic. We recall that terms in Presburger arithmetic are defined by the grammar $t ::= 0 \mid 1 \mid \mathsf{x} \mid t + t$ where $\mathsf{x} \in \mathrm{VAR}$ and 0 and 1 are distinguished constants (see also Section 1.3). Given an n-tuple $\vec{t}$ of terms and $k \geq 1$, we write $g^k(\vec{t})$ to denote the n-tuple obtained from the expression $A^k\vec{t} + A^{k-1}\vec{b} + \cdots + \vec{b}$. Similarly, we write $\psi(\vec{t})$ to denote the Presburger formula $\exists \mathsf{x}_1, \ldots, \mathsf{x}_n\ \psi(\mathsf{x}_1, \ldots, \mathsf{x}_n) \wedge (\bigwedge_{i \in [1,n]} \mathsf{x}_i = \vec{t}(i))$. In the case $\vec{t}$ contains negative constants or negative factors, we replace $\mathsf{x}_i = \vec{t}(i)$ by its variant in which negative terms are moved to the left of

the equality. In this way, we completely respect our initial syntax for Presburger arithmetic. For instance, if $\vec{t}$ takes the value below

$$\vec{t} = \begin{pmatrix} 2 & -2 \\ -3 & 7 \end{pmatrix} \begin{pmatrix} \mathsf{x} \\ \mathsf{y} \end{pmatrix} + \begin{pmatrix} 1 \\ -2 \end{pmatrix} = \begin{pmatrix} 2\mathsf{x} - 2\mathsf{y} + 1 \\ -3\mathsf{x} + 7\mathsf{y} - 2 \end{pmatrix}$$

then $\psi(\vec{t})$ is equal to

$$\exists \mathsf{x}_1, \ldots, \mathsf{x}_n \; \psi(\mathsf{x}_1, \ldots, \mathsf{x}_n) \wedge \mathsf{x}_1 + 2\mathsf{y} = 2\mathsf{x} + 1 \wedge \mathsf{x}_2 + 3\mathsf{x} + 2 = 7\mathsf{y}.$$

Remember that $(\vec{x}, \vec{x'}) \in R^*$ iff there is $i \geq 0$ such that $\vec{x'} = g^i(\vec{x})$ and for every $0 \leq j < i$, $g^j(\vec{x}) \models \psi$. So, the Presburger formula defining R^* could look like

$$\exists\, i \; (\vec{\mathsf{x}'} = g^i(\vec{\mathsf{x}})) \wedge \bigwedge_{j<i} \psi(g^j(\vec{\mathsf{x}})).$$

Unfortunately, $g^i(\vec{\mathsf{x}})$ is a shortcut for $A^i\vec{\mathsf{x}} + A^{i-1}\vec{b} + \cdots + \vec{b}$ and the generalized conjunction has exactly i conjuncts, which disqualifies this expression as a formula from Presburger arithmetic. Instead, the proof below uses $A^{\alpha+\beta} = A^\alpha$ in order to replace i applications of the map g by expressions in which i may appear as a variable that is multiplied by a constant factor. To do so, we shall show that for $q \geq 1$, we have $g^{\alpha+q\beta}(\vec{a}) = g^\alpha(\vec{a}) + qA^\alpha g^\beta(\vec{0})$; observe that q as an exponent is transformed into a factor and $A^\alpha g^\beta(\vec{0})$ is in $\mathbb{Z}^n$. First, let us consider the following identities:

$$\begin{aligned} g^{\alpha+\beta}(\vec{a}) &= A^{\alpha+\beta}\vec{a} + A^{\alpha+\beta-1}\vec{b} + \cdots + \vec{b}. \\ &= A^{\alpha+\beta}\vec{a} + A^\alpha(A^{\beta-1}\vec{b} + \cdots + \vec{b}) + (A^{\alpha-1}\vec{b} + \cdots + \vec{b}) \\ &= A^\alpha\vec{a} + A^\alpha g^\beta(\vec{0}) + (A^{\alpha-1}\vec{b} + \cdots + \vec{b}) \\ &= g^\alpha(\vec{a}) + A^\alpha g^\beta(\vec{0}). \end{aligned}$$

Now, let us show that for $q \geq 1$, we have $g^{\alpha+q\beta}(\vec{a}) = g^\alpha(\vec{a}) + qA^\alpha g^\beta(\vec{0})$. The case $q = 1$ is treated above. For the induction step, let us compute the value of $g^{\alpha+(q+1)\beta}(\vec{a})$. By the induction hypothesis, we have $g^{\alpha+(q+1)\beta}(\vec{a}) = g^\alpha(g^\beta(\vec{a})) + qA^\alpha g^\beta(\vec{0})$. Using the argument from the base case, we get $g^{\alpha+(q+1)\beta}(\vec{a}) = g^\alpha(\vec{a}) + A^\alpha g^\beta(\vec{0}) + qA^\alpha g^\beta(\vec{0})$, which entails $g^{\alpha+(q+1)\beta}(\vec{a}) = g^\alpha(\vec{a}) + (q+1)A^\alpha g^\beta(\vec{0})$.

For each $i \geq 0$, it is easy to define a Presburger formula $\mathtt{R}[i]$ such that $\mathrm{REL}(\mathtt{R}[i]) = \{(\vec{y}, \vec{y'}) \in \mathbb{N}^{2n} : \vec{y}R^i\vec{y'}\}$. For instance, $\mathtt{R}[0]$ is equal to $\bigwedge_{j\in[1,n]} \mathsf{x}_j = \mathsf{x}'_j$ and $\mathtt{R}[i+1]$ is equal to $\exists\, \vec{\mathsf{y}}\; \psi(\vec{\mathsf{y}}) \wedge \mathtt{R}[i](\vec{\mathsf{x}}, \vec{\mathsf{y}}) \wedge \vec{\mathsf{x}'} = A\vec{\mathsf{y}} + \vec{b}$, where $\vec{\mathsf{x}'} = A\vec{\mathsf{y}} + \vec{b}$ is again understood as a conjunction with n conjuncts.

In order to show that R has a Presburger counting iteration, we define below a formula $\chi(\vec{\mathsf{x}}, \mathsf{z}, \vec{\mathsf{x}'})$ such that $R_{\mathbf{CI}} = \mathrm{REL}(\chi(\vec{\mathsf{x}}, \mathsf{z}, \vec{\mathsf{x}'}))$. To do so, we use the fact that whenever $(\vec{y}, \vec{y'}) \in R^i$, either $(\vec{y}, \vec{y'}) \in R^i$ and $i < \alpha$ or $(\vec{y}, \vec{y'}) \in R^i$, $i \geq \alpha$, there are $(r, q) \in [0, \beta - 1] \times \mathbb{N}$ such that $i - \alpha = r + q\beta$, $\vec{y'} = g^\alpha(\vec{y}) + qA^\alpha g^\beta(\vec{0})$ and for $0 \leq i' < i$, $g^{i'}(\vec{y})$ satisfies ψ. Here is the formula $\chi(\vec{\mathsf{x}}, \mathsf{z}, \vec{\mathsf{x}'})$:

$$((\mathsf{z} = 0 \wedge \mathtt{R}[0]) \vee \cdots \vee (\mathsf{z} = \alpha - 1 \wedge \mathtt{R}[\alpha - 1])) \vee (\mathsf{z} \geq \alpha \wedge \exists q \; \underbrace{(\chi_{q,0} \vee \cdots \vee \chi_{q,\beta-1})}_{\text{one formula per remainder } r})$$

where for $r \in [0, \beta - 1]$, the formula $\chi_{q,r}$ is defined as follows:

$$(\mathsf{z} = \alpha + r + \beta \times q) \wedge \underbrace{(\exists \vec{\mathsf{y}'}\ \underbrace{(\vec{\mathsf{y}'} = A^\alpha \vec{\mathsf{x}} + qA^\alpha(A^{\beta-1}\vec{b} + \cdots + \vec{b}))}_{\vec{y'} = g^{\mathsf{z}-r}(\vec{\mathsf{x}})} \wedge (\vec{\mathsf{x}'} = g^r(\vec{\mathsf{y}'})))}_{\vec{x'} = g^{\mathsf{z}}(\vec{\mathsf{x}})} \wedge \chi^{\mathrm{guard}}(\mathsf{z}, \vec{\mathsf{x}})$$

The formula $\chi^{\mathrm{guard}}(\mathsf{z}, \vec{\mathsf{x}})$ checks that the guard is satisfied for all the intermediate configurations:

$$\chi^{\mathrm{guard}}(\mathsf{z}, \vec{\mathsf{x}}) \stackrel{\text{def}}{=} (\bigwedge_{i \in [1,\alpha]} \exists\, \vec{\mathsf{y}}\, \mathtt{R}[i](\vec{\mathsf{x}}, \vec{\mathsf{y}})) \wedge \forall\, \mathsf{z}'\ \alpha \leq \mathsf{z}' < \mathsf{z} \Rightarrow$$

$$\bigvee_{r' \in [1,\beta-1]} \exists\, q'\ (\mathsf{z}' = \alpha + r' + q'\beta \wedge (\exists \vec{\mathsf{y}'}\ \underbrace{(\vec{\mathsf{y}'} = A^\alpha \vec{\mathsf{x}} + q' A^\alpha (A^{\beta-1}\vec{b} + \cdots + \vec{b}))}_{\vec{y'} = g^{\mathsf{z}'-r'}(\vec{\mathsf{x}})}$$

$$\wedge\ \overbrace{\psi(\underbrace{g^{r'}(\vec{\mathsf{y}'})}_{= g^{\mathsf{z}'}(\vec{\mathsf{x}})})}^{\text{guard satisfaction}}\ )))$$

It is now easy to check that $\chi(\vec{\mathsf{x}}, \mathsf{z}, \vec{\mathsf{x}'})$ belong to Presburger arithmetic and in particular no multiplication between variables is present in it. QED

A recent work unifying [CJ98, FL02, BGI09, BIL09] by considering all the families of formulae labelling transitions from these works can be found in [BIK10].

A loop in an affine counter system has the *finite monoid property* $\stackrel{\text{def}}{\Leftrightarrow}$ its corresponding affine update $(A, \vec{b}, \psi)$, possibly obtained by composition of several affine updates, satisfies that A^* is finite.

5.3.3 Admissible counter systems

Let us introduce below the class of admissible counter systems.

Definition 5.3.2. A counter system $\mathcal{S}$ is *admissible* iff

1. $\mathcal{S}$ is an affine counter system,
2. there is at most one transition between two control states,
3. its control graph is flat (see Section 1.4.3),
4. each loop has the finite monoid property.

∇

Uniqueness of the transitions between two control states is a consequence of flatness. Definition 5.3.2 can be generalized as done in [DFGvD06] by requiring flatness, (effective) Presburger counting iteration for every loop and functionality, which are all properties satisfied by admissible counter systems with Definition 5.3.2. The restriction to admissible counter systems mainly takes advantage of Lemma 5.3.2 as shown below.

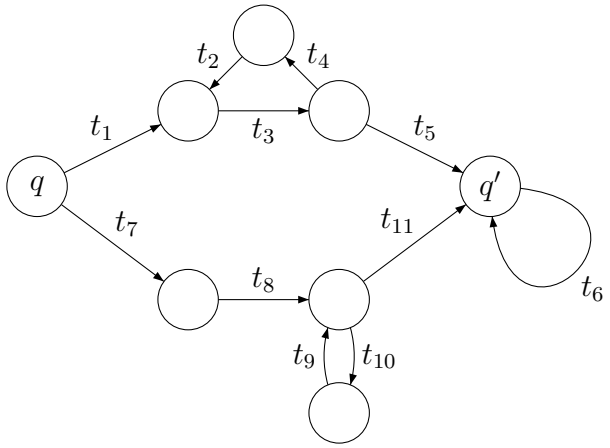


Figure 5.3: An admissible counter system

Theorem 5.3.3. [FL02, Ler03] Let $\mathcal{S}$ be an admissible counter system and $q, q' \in Q$. One can effectively compute a Presburger formula φ such that for every valuation $\mathbf{val}$, we have $\mathbf{val} \models \varphi$ iff $(q, (\mathbf{val}(\mathsf{x}_1), \ldots, \mathbf{val}(\mathsf{x}_n))) \xrightarrow{*} (q', (\mathbf{val}(\mathsf{x}'_1), \ldots, \mathbf{val}(\mathsf{x}'_n)))$.

If we give up the assumption on the finite monoid property, the reachability problem is undecidable for flat affine counter systems [Cor02]. However, Theorem 5.3.3 still holds true if we relax a bit the notion of admissibility for instance by allowing that between two control states for which no transition belongs to a cycle, more than one transitions are allowed.

Proof: Let $\mathcal{S} = (Q, n, \delta)$ be an admissible counter system and $q, q' \in Q$. We define below a finite-state automaton that overapproximates the language of transitions between q and q' (constraints on counters are simply ignored). By way of example, let us consider the admissible counter system from Figure 5.3.3. The language of transitions between q and q' can be approximated by the union below:

$$t_1 t_3 (t_4 t_2 t_3)^* t_5 t_6^* \cup t_7 t_8 (t_{10} t_9)^* t_{11} t_6^*$$

Let $\mathcal{A} = (\Sigma, Q, Q_0, \delta', F)$ be a finite-state automaton such that

- $\Sigma = \delta$,
- $Q_0 = \{q\}$, $F = \{q'\}$,
- $q_1 \xrightarrow{t} q_2 \in \delta' \overset{\text{def}}{\Leftrightarrow} t$ is of the form $q_1 \xrightarrow{(A, \vec{b}, \psi)} q_2$.

Since $\mathcal{S}$ is flat, $\mathrm{L}(\mathcal{A})$ is a finite union of bounded languages of the form

$$\mathrm{L} = u_1 (v_1)^* u_2 (v_2)^* \cdots (v_k)^* u_{k+1}$$

with $u_i \in \Sigma^*$ and $v_i \in \Sigma^+$. Moreover, by flatness, for each u_i occurring between $(v_i)^*$ and $(v_{i+1})^*$, we have $u_i \in \Sigma^+$.

By Lemma 5.3.2, there is a Presburger formula that encodes the effect of applying a finite number of times the sequence of transitions v_i. Similarly, by Lemma 5.3.1, there is a Presburger

formula that encodes the effect of applying once the sequence of transitions u_i. Hence, one can effectively compute the effect of applying a sequence of transitions in the language L; it is sufficient to use existential quantification for intermediate positions.

Let us be a bit more precise by considering the sequence below:

$$u_1(v_1)^*u_2(v_2)^*\cdots(v_k)^*u_{k+1}$$

- ⋆ By closure under composition, for $i \in [1, k+1]$, there is a Presburger formula $\psi^i_{seg}(\vec{\mathsf{x}}, \vec{\mathsf{x}'})$ that encodes the effect of segments of transitions u_i.
- ⋆ By previous theorem, for $i \in [1, k]$, there is a Presburger formula $\psi^i_{loop}(\vec{\mathsf{x}}, \mathsf{z}, \vec{\mathsf{x}'})$ that encodes the effect of the loop v_i.
- ⋆ Presburger formula encoding the effect of the above sequence is the following (free variables in $\vec{\mathsf{x}}$, $\vec{\mathsf{x}'}$):

$$\exists\, \mathsf{z}_1, \ldots, \mathsf{z}_k,\ \vec{\mathsf{y}'_1}, \vec{\mathsf{y}_2}, \vec{\mathsf{y}'_2}, \ldots, \vec{\mathsf{y}_{k+1}}$$
$$\psi^1_{seg}(\vec{\mathsf{x}}, \vec{\mathsf{y}'_1}) \wedge \psi^1_{loop}(\vec{\mathsf{y}'_1}, \mathsf{z}_1, \vec{\mathsf{y}_2}) \wedge \psi^2_{seg}(\vec{\mathsf{y}_2}, \mathsf{z}_1, \vec{\mathsf{y}'_2}) \wedge \psi^2_{loop}(\vec{\mathsf{y}'_2}, \mathsf{z}_2, \vec{\mathsf{y}_3}) \wedge \cdots$$
$$\cdots \wedge \psi^k_{loop}(\vec{\mathsf{y}'_k}, \mathsf{z}_k, \vec{\mathsf{y}_{k+1}}) \wedge \psi^{k+1}_{seg}(\vec{\mathsf{y}_{k+1}}, \vec{\mathsf{x}'})$$

Since $\mathrm{L}(\mathcal{A})$ is made of a finite union of bounded languages and Presburger arithmetic has obviously disjunction, one can effectively compute a Presburger formula $\varphi(\vec{\mathsf{x}}, \vec{\mathsf{x}'})$ such that for every valuation $\mathbf{val}$, we have $\mathbf{val} \models \varphi$ iff $(q, (\mathbf{val}(\mathsf{x}_1), \ldots, \mathbf{val}(\mathsf{x}_n))) \xrightarrow{*} (q', (\mathbf{val}(\mathsf{x}'_1), \ldots, \mathbf{val}(\mathsf{x}'_n)))$. QED

As observed in [CJ98, FL02, Ler03, BIL09], flatness is very often essential to get effective semilinear reachability sets (but of course this is not a necessary condition, see e.g. [Par66, HP79]). However, flat counter systems are seldom natural in real-life applications. That is why, a relaxed version of flatness has been considered in [LS05, DFGvD06] so that an initialized counter system $(\mathcal{S}, (q, \vec{x}))$ is *flattable* whenever there is a partial unfolding of $(\mathcal{S}, (q, \vec{x}))$ that is flat and has the same reachability set as $(\mathcal{S}, (q, \vec{x}))$. In that way, reachability questions on $(\mathcal{S}, (q, \vec{x}))$ can still be decided even in the absence of flatness. For the sake of completeness, let us provide below basic definitions about flattable counter systems.

Let L be a finite union of bounded languages of the form

$$u_1(v_1)^*u_2(v_2)^*\cdots(v_k)^*u_{k+1},$$

where $u_i \in \Sigma^*$, $v_i \in \Sigma^+$, $\Sigma = \delta$ is the set of transitions from $\mathcal{S}$ such that in the expression

$$u_1(v_1)^*u_2(v_2)^*\cdots(v_k)^*u_{k+1},$$

two consecutive transitions share an intermediate control state (as in the proof of Theorem 5.3.3). So, $(\mathcal{S}, (q, \vec{x}))$ is *initially flattable* [LS05] iff there is some language L of the above form such that the configurations reachable from $(q, \vec{x})$ are those reachable by firing the sequences of transitions from L (not every such sequence leads to a run). So, there is some language L of the above form such that

$$\{(q', \vec{x'}) : (q, \vec{x}) \xrightarrow{*} (q', \vec{x'})\} = \{(q', \vec{x'}) : (q, \vec{x}) \xrightarrow{u} (q', \vec{x'}), u \in \mathrm{L}\}$$

For instance, the initialized relational counter system $(\mathcal{S}, (q_1, \vec{0}))$ in Figure 1.4.3 is initially flattable. Similarly, $\mathcal{S}$ is *uniformly flattable* [LS05] iff there is some language L of the above form such that

$$\xrightarrow{*} = \{((q, \vec{x}), (q', \vec{x'})) : (q, \vec{x}) \xrightarrow{u} (q', \vec{x'}), u \in \mathrm{L}\}$$

Surprisingly, standard classes of counter automata contain already flattable counter systems.

Theorem 5.3.4. [LS05]

(I) Uniformly reversal-bounded counter automata are uniformly flattable and reversal-bounded initialized counter automata are initially flattable.

(II) Initialized gainy counter automata are initially flattable.

Theorem 5.3.4(I) can be refined since the language L, finite union of bounded languages, can be effectively computed from a uniformly reversal-bounded counter automaton, which provides an alternative proof for the effective semilinearity of the reachability relation. Indeed, an initialized counter automaton and a finite union of bounded languages can be simulated by an admissible counter system.

5.3.4 $\mathrm{LTL}^{\mathrm{CS}}(\mathrm{PrA})$ model-checking for admissible counter systems

A consequence of Theorem 5.3.3, the reachability problem for admissible counter systems is decidable. However, this result can be improved by showing decidability of $\mathrm{LTL}^{\mathrm{CS}}(\mathrm{PrA})$ model-checking thanks to an encoding of runs in Presburger arithmetic.

Theorem 5.3.5. [DFGvD06] Existential model-checking problem for $\mathrm{LTL}^{\mathrm{CS}}(\mathrm{PrA})$ restricted to admissible counter systems is decidable.

In [DFGvD06], Theorem 5.3.5 is extended to a branching-time variant of $\mathrm{LTL}^{\mathrm{CS}}(\mathrm{PrA})$.

Proof: Let $\mathcal{S} = (Q, n, \delta)$ be an admissible counter system, $(q, \vec{x})$ be a configuration and φ be an $\mathrm{LTL}^{\mathrm{CS}}(\mathrm{PrA})$ formula. Without any loss of generality, we can assume that φ has no control states as atomic formulae; otherwise we can add one counter that behaves as the control states. We wish to check whether there is an infinite run ρ starting from $(q, \vec{x})$ such that $\rho, 0 \models \varphi$. To do so, we shall build a Presburger formula ψ (with free variables $\mathsf{x}_1, \ldots, \mathsf{x}_n$) such that for every valuation $\mathbf{val}$, we have $\mathbf{val} \models \psi$ iff there is an infinite run ρ starting from $(q, (\mathbf{val}(\mathsf{x}_1), \ldots, \mathbf{val}(\mathsf{x}_n)))$ such that $\rho, 0 \models \varphi$. Then, the existence of the infinite run from $(q, \vec{x})$ satisfying φ is equivalent to the satisfaction of $\psi \wedge (\bigwedge_{i \in [1,n]} \mathsf{x}_i = \vec{x}(i))$.

As in the proof of Theorem 5.3.3, we consider sequences of transitions and we introduce a Büchi automaton that accepts ω-sequences of transitions starting from the control state q. Let $\mathcal{A} = (\Sigma, Q, Q_0, \delta', F)$ be the Büchi automaton such that

- $\Sigma = \delta$, $Q_0 = \{q\}$, $F = Q$,
- $q_1 \xrightarrow{t} q_2 \in \delta' \stackrel{\text{def}}{\Leftrightarrow} t$ is of the form $q_1 \xrightarrow{(A, \vec{b}, \psi)} q_2$.

Since $\mathcal{S}$ is flat, $\mathrm{L}(\mathcal{A})$ is a finite union of languages of the form

$$\mathrm{L} = u_1(v_1)^* u_2(v_2)^* \cdots (v_k)^\omega$$

with $u_i \in \Sigma^*$ and $v_i \in \Sigma^+$. Such an expression is called below a *run schema* and it may correspond to an infinite number of runs starting at $(q, \vec{x})$ depending how many times each sequence v_i is visited. Alternatively, it may not correspond to any run from $(q, \vec{x})$, for instance if the sequence of transitions v_k can be taken only a finite amount of times. For instance, with the admissible counter system described in Figure 5.3.3, we obtain the following run schemata:

$$t_1 t_3 (t_4 t_2 t_3)^* t_5 t_6^\omega, t_1 t_3 (t_4 t_2 t_3)^\omega, t_7 t_8 (t_{10} t_9)^* t_{11} t_6^\omega, t_7 t_8 (t_{10} t_9)^\omega.$$

Observe that the number of run schemata is at most exponential in the size of $\mathcal{S}$ and the run schemata can be effectively computed.

A run schema $\mathrm{L} = u_1(v_1)^* u_2(v_2)^* \cdots (v_k)^\omega$ and natural numbers $m_1, \ldots, m_{k-1}$ specify a unique sequence $u_1(v_1)^{m_1} u_2(v_2)^{m_2} \cdots (v_k)^\omega$ that may correspond to an infinite run from $(q, \vec{x})$ (or not). However, note that once L and $m_1, \ldots, m_{k-1}$ are fixed, there is at most one infinite run from $(q, \vec{x})$ obtained from the sequence of transitions $u_1(v_1)^{m_1} u_2(v_2)^{m_2} \cdots (v_k)^\omega$. Indeed, the update functions in affine counter systems (and *a fortiori* in admissible counter systems) are also deterministic.

By using Lemma 5.3.1 and Theorem 5.3.3, one can effectively build Presburger formulae $\chi^{\exists}_{\mathrm{L}}(\mathsf{z}_1, \ldots, \mathsf{z}_{k-1}, \vec{\mathsf{x}})$ and $\chi^{steps}_{\mathrm{L}}(\mathsf{z}_1, \ldots, \mathsf{z}_{k-1}, \vec{\mathsf{x}}, \mathsf{z}, \vec{\mathsf{x}'})$ (see details of the contruction in [DFGvD06]) such that for every valuation $\mathbf{val}$,

- $\star$ $\mathbf{val} \models \chi^{\exists}_{\mathrm{L}}(\mathsf{z}_1, \ldots, \mathsf{z}_{k-1}, \vec{\mathsf{x}})$ iff there is an infinite run starting from $(q, (\mathbf{val}(\mathsf{x}_1), \ldots, \mathbf{val}(\mathsf{x}_n)))$ obtained from the sequence of transitions $u_1(v_1)^{\mathbf{val}(\mathsf{z}_1)} u_2(v_2)^{\mathbf{val}(\mathsf{z}_2)} \cdots (v_k)^\omega$.
- $\star$ $\mathbf{val} \models \chi^{steps}_{\mathrm{L}}(\mathsf{z}_1, \ldots, \mathsf{z}_{k-1}, \vec{\mathsf{x}}, \mathsf{z}, \vec{\mathsf{x}'})$ iff $\mathbf{val} \models \chi^{\exists}_{\mathrm{L}}(\mathsf{z}_1, \ldots, \mathsf{z}_{k-1}, \vec{\mathsf{x}})$ and the $\mathbf{val}(\mathsf{z})$th tuple of counter values in the infinite run is $(\mathbf{val}(\mathsf{x}'_1), \ldots, \mathbf{val}(\mathsf{x}'_n))$.

In order to define such formulae, let us just mention that one needs to introduce variables for the counter values obtained after applying the transitions for the following prefixes:

$$u_1, u_1(v_1)^{\mathbf{val}(\mathsf{z}_1)}, u_1(v_1)^{\mathbf{val}(\mathsf{z}_1)} u_2, u_1(v_1)^{\mathbf{val}(\mathsf{z}_1)} u_2(v_2)^{\mathbf{val}(\mathsf{z}_2)}, \ldots, u_1(v_1)^{\mathbf{val}(\mathsf{z}_1)} \cdots (v_{k-1})^{\mathbf{val}(\mathsf{z}_{k-1})}$$

The formula ψ can be now defined as a finite disjunction (each disjunct is parameterized by a run schema) and the values m_1, …, m_{k-1} are encoded by existential quantifications, which allows us to quantify over all possible runs in Presburger arithmetic. The translation map t_{L}, parametrized by the run schema L, simply mimicks $\mathrm{LTL}^{\mathrm{CS}}(\mathrm{PrA})$ semantics in first-order logic, i.e. in Presburger arithmetic. Here is the definition of ψ:

$$\bigvee_{\mathrm{L}=u_1(v_1)^* u_2(v_2)^* \cdots (v_k)^\omega} (\exists \mathsf{z}_1, \ldots, \mathsf{z}_{k-1}, \mathsf{z}_0 \; \chi^{\exists}_{\mathrm{L}}(\mathsf{z}_1, \ldots, \mathsf{z}_{k-1}, \vec{\mathsf{x}}) \wedge \mathsf{z}_0 = 0 \wedge t_{\mathrm{L}}(\mathsf{z}_0, \varphi))$$

The translation map t_{L} is defined as follows:

- $\star$ t_{L} is homomorphic for Boolean connectives.

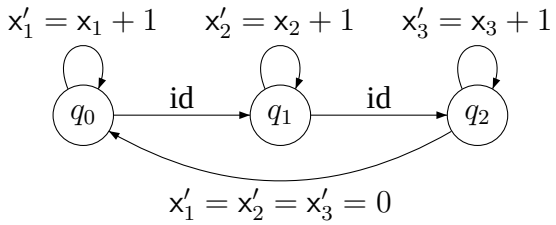


Figure 5.4: An almost admissible counter system

- $t_{\mathrm{L}}(\mathsf{z}, \psi(\vec{\mathsf{y}}, \vec{\mathsf{x}})) \stackrel{\text{def}}{=} \forall\, \vec{\mathsf{x}'}\ (\chi_{\mathrm{L}}^{steps}(\mathsf{z}_1, \ldots, \mathsf{z}_{k-1}, \vec{\mathsf{x}}, \mathsf{z}, \vec{\mathsf{x}'}) \Rightarrow \psi(\vec{\mathsf{y}}, \vec{\mathsf{x}'}))$ where $\psi(\vec{\mathsf{y}}, \vec{\mathsf{x}})$ is an atomic formula with a tuple $\vec{\mathsf{y}}$ of variables from $\mathrm{VAR}^{\mathrm{p}}$.
- $t_{\mathrm{L}}(\mathsf{z}, \mathtt{X}\psi) \stackrel{\text{def}}{=} \exists\, \mathsf{z}'\ (\mathsf{z}' = \mathsf{z} + 1) \wedge t_{\mathrm{L}}(\mathsf{z}', \psi)$.
- The definition of $t_{\mathrm{L}}(\mathsf{z}, \psi_1 \mathtt{U} \psi_2)$ is analogous.
- $t_{\mathrm{L}}(\mathsf{z}, \forall\, \mathsf{y}\ \psi) \stackrel{\text{def}}{=} \forall\, \mathsf{y}\ t_{\mathrm{L}}(\mathsf{z}, \psi)$.

QED

It is worth noting that we have established decidability but the characterization of the computational complexity is still open for the model-checking problem for $\mathrm{LTL}^{\mathrm{CS}}(\mathrm{PrA})$ restricted to admissible counter systems. Moreover, it is open whether the above decidability results still hold with the linear μ-calculus extension.

Moreover, in the above proof, flatness is essential and Figure 5.4 presents a affine counter system $\mathcal{S}_u$ of dimension 3 such that between two control states there is at most one transition and each transition defines a functional relations (id refers to a transition that do not change the counter values). However, $\mathcal{S}_u$ is not flat because of the existence of the transition between q_2 and q_1.

Theorem 5.3.6. Existential model-checking problem for $\mathrm{LTL}^{\mathrm{CS}}(\mathrm{PrA})$ restricted to the affine counter system $\mathcal{S}_u$ is undecidable.

Proof: The proof is by reducing the recurrence problem for nondeterministic Minsky machines that is shown Σ_1^1-hard in [AH94]. A nondeterministic Minsky machine M consists of two counters C_1 and C_2, and a sequence of $n \geq 1$ instructions. The lth instruction is written as one of the following:

l : $\mathsf{C}_i := \mathsf{C}_i + 1$; goto l' or goto l''.

l : if $\mathsf{C}_i = 0$ then goto l' else $\mathsf{C}_i := \mathsf{C}_i - 1$; goto l''_0 or goto l''_1.

We represent the configurations of M by triples (c_1, c_2, l) where $1 \leq l \leq n$, $c_1 \geq 0$ and $c_2 \geq 0$. A computation of M is a finite sequence of related configurations, starting with the initial configuration $(0, 0, 1)$ (location encoded as last element). The recurrence problem can be stated as the existence of an infinite execution that passes through the instruction 1 infinitely often. We shall build a formula φ of $\mathrm{LTL}^{\mathrm{CS}}(\mathrm{PrA})$ such that M visits 1 infinitely often iff there is an infinite run ρ starting at $(q_2, (0, 0, 1))$ such that $\rho, 0 \models \varphi$. The formula φ is of the form

$$\mathtt{GF}(\mathsf{x}_3 = 1 \wedge \mathtt{X}q_0) \wedge \bigwedge_{1 \leq l \leq n} \mathtt{G}\psi_l,$$

where ψ_l encodes the l-th instruction. For instance, the l-th instruction "$\mathsf{C}_1 := \mathsf{C}_1 + 1$; goto l_0'' or goto l_1'' is encoded by

$$\forall \mathsf{y}, \mathsf{z}\, (\mathsf{x}_1 = \mathsf{y} \ \wedge\ \mathsf{x}_2 = \mathsf{z} \ \wedge\ \mathsf{x}_3 = l \ \wedge\ \mathtt{X} q_0) \Rightarrow$$

$$\mathtt{X}(\neg(\mathtt{X} q_0)\ \mathtt{U}\ (\mathtt{X} q_0 \ \wedge\ \overbrace{\mathsf{x}_1 = \mathsf{y} + 1}^{\text{increase } \mathsf{C}_1} \wedge \mathsf{x}_2 = \mathsf{z} \ \wedge\ (\mathsf{x}_3 = l_0'' \vee \mathsf{x}_3 = l_1''))).$$

Other instructions can be encoded similarly. QED

Here are a few open problems related to the second part of this chapter:

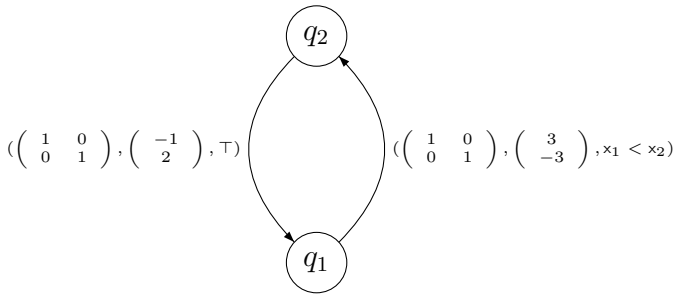
- ⋆ Computational complexity of the model-checking problem for $\mathrm{LTL}^{\mathrm{CS}}(\mathrm{PrA})$ restricted to admissible counter systems is still open.
- ⋆ Decidability extends to a CTL* extension of $\mathrm{LTL}^{\mathrm{CS}}(\mathrm{PrA})$. What about the linear μ-calculus extension?
- ⋆ Which conditions in the definition of admissible counter systems can be relaxed so that the model-checking problem for $\mathrm{LTL}^{\mathrm{CS}}(\mathrm{PrA})$ remains decidable?

5.4 Exercises

Exercise 5.4.1. Show that control state reachability problem for gainy counter automata is equivalent to control state reachability problem for reset VASS.

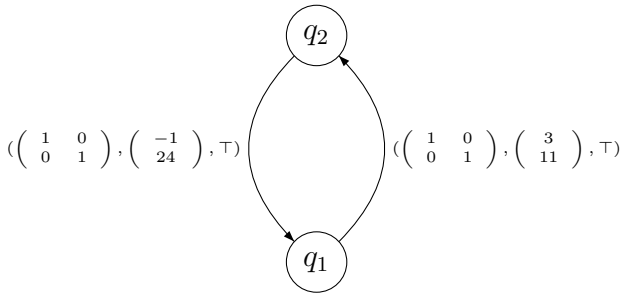
Exercise 5.4.2. Prove Lemma 5.1.1.

Exercise 5.4.3.



1. Compute $\varphi(\mathsf{x}_1, \mathsf{x}_2, \mathsf{x}_1', \mathsf{x}_2')$ such that for every $\mathbf{val}$, we have $\mathbf{val} \models \varphi$ iff $(q_1, \mathbf{val}(\mathsf{x}_1), \mathbf{val}(\mathsf{x}_2)) \xrightarrow{*} (q_1, \mathbf{val}(\mathsf{x}_1'), \mathbf{val}(\mathsf{x}_2'))$.
2. Same question when $\top$ is replaced by $\neg(\mathsf{x}_1 \equiv_{15} \mathsf{x}_2)$.

Exercise 5.4.4.



1. Compute $\varphi(\mathsf{x}_1, \mathsf{x}_2, \mathsf{z}, \mathsf{x}'_1, \mathsf{x}'_2)$ such that for every $\mathbf{val}$, we have $\mathbf{val} \models \varphi$ iff on the unique run starting at $(q_1, \mathbf{val}(\mathsf{x}_1), \mathbf{val}(\mathsf{x}_2))$, the $\mathbf{val}(\mathsf{z})$th configuration has counter values $(\mathbf{val}(\mathsf{x}'_1), \mathbf{val}(\mathsf{x}'_2))$.

2. Given a Presburger formula $\psi(\mathsf{y}_1, \mathsf{y}_2)$ viewed as a constraint on counter values, compute $\varphi'(\mathsf{x}_1, \mathsf{x}_2)$ such that for every $\mathbf{val}$, we have $\mathbf{val} \models \varphi'$ iff on the unique run starting at $(q_1, \mathbf{val}(\mathsf{x}_1), \mathbf{val}(\mathsf{x}_2))$, the number of configurations with counter values satisfying $\psi(\mathsf{y}_1, \mathsf{y}_2)$ is infinite.

Exercise 5.4.5. Define the formulae $\chi_{\mathrm{L}}^{\exists}(\mathsf{z}_1, \ldots, \mathsf{z}_{k-1}, \vec{\mathsf{x}})$ and $\chi_{\mathrm{L}}^{steps}(\mathsf{z}_1, \ldots, \mathsf{z}_{k-1}, \vec{\mathsf{x}}, \mathsf{z}, \vec{\mathsf{x}'})$ in the proof of Theorem 5.3.5.

Exercise 5.4.6. Prove Theorem 5.3.4(II). Hint: use the fact that any upward closed subset of $\mathbb{N}^n$ (for the ordering $\preceq$) has a finite set of minimal elements.

Exercise 5.4.7. Complete the proof of Theorem 5.3.5.

Exercise 5.4.8. Consider the extension of $\mathrm{LTL}^{\mathrm{CS}}(\mathrm{PrA})$ with existential and universal quantifications over infinite runs as in the branching-time temporal logic CTL*. Show that the model-checking problem for this extension over admissible counter systems is decidable.

Exercise 5.4.9. Consider the variant of $\mathrm{LTL}^{\mathrm{CS}}(\mathrm{PrA})$ with existential and universal quantifications over infinite runs as in the branching-time temporal logic CTL* in which the atomic formulae are reduced to control states. Show that the model-checking problem for this extension over VASS is undecidable.

Exercise 5.4.10. Let LTL$^+$ be the fragment of the logic $\mathrm{LTL}^{\mathrm{CS}}(\mathrm{PrA})$

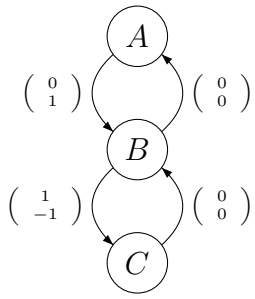
- $\star$ with temporal operators $\mathtt{X}$, $\mathtt{U}$ and standard Boolean connectives,
- $\star$ atomic formulae are restricted to control states or zero-tests of the form $\mathsf{x}_j = 0$.
- $\star$ without first-order quantification.

EXISTENTIAL MODEL-CHECKING PROBLEM FOR LTL$^+$ RESTRICTED TO VASS is defined as follows:

input: VASS $\mathcal{V}$, configuration $(q, \vec{x})$ and a formula φ built over the control states and counters from $\mathcal{V}$.

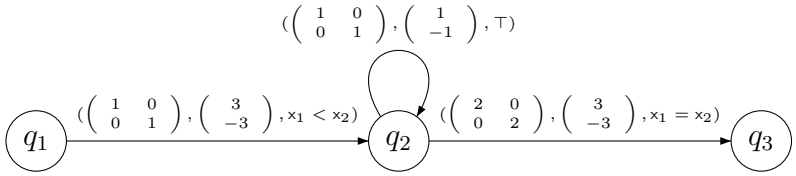
question: is there an infinite run ρ starting at $(q, \vec{x})$ satisfying φ (written $\rho, 0 \models \varphi$)?

Let us consider the VASS below:



1. For each formula below, determine whether there is an infinite run starting at $(A, \vec{0})$ such that $\rho, 0 \models \varphi$.
   (a) $\varphi = \texttt{GF}\ A$,
   (b) $\varphi = \texttt{GF}\ (\mathsf{x}_2 = 0)$,
   (c) $\varphi = \texttt{GF}\ (\mathsf{x}_1 = 0) \wedge \texttt{GF}\ C$,
   (d) $\varphi = \texttt{G}(C \Rightarrow \texttt{XG}\neg(\mathsf{x}_1 = 0))$,
   (e) $(\texttt{GF}\ A) \wedge (\texttt{GF}\ B) \wedge (\texttt{GF}\ C) \wedge (\texttt{GF}\ \mathsf{x}_2 = 0) \wedge (\texttt{GF}\ \neg(\mathsf{x}_1 = 0))$.
2. What are the formulae among (a)-(e) such that all the infinite runs starting at $(A, \vec{0})$, we have $\rho, 0 \models \varphi$?
3. Show that the existential model-checking for LTL$^+$ restricted to VASS is undecidable.

Exercise 5.4.11. Let us consider the affine counter system below:



Design a Presburger formula $\varphi(\mathsf{x}_1, \mathsf{x}_2, \mathsf{y}_1, \mathsf{y}_2)$ such that for every valuation $\mathbf{val}$, we have $\mathbf{val} \models \varphi$ iff $(q_1, \mathbf{val}(\mathsf{x}_1), \mathbf{val}(\mathsf{x}_2)) \xrightarrow{*} (q_3, \mathbf{val}(\mathsf{y}_1), \mathbf{val}(\mathsf{y}_2))$, i.e. $(q_3, \mathbf{val}(\mathsf{y}_1), \mathbf{val}(\mathsf{y}_2))$ is reachable from $(q_1, \mathbf{val}(\mathsf{x}_1), \mathbf{val}(\mathsf{x}_2))$.

Chapter 6

Concluding Remarks

In order to conclude these notes, other related topics are enumerated below. By no means, this is intended to be exhaustive and the main intention is to provide pointers for further reading.

Other topics related to these notes could have been added. Here some examples below:

- ⋆ Theory of well-structured transition systems, see e.g., [AJ96, FS01, FMP04].
- ⋆ The decidability proof for the reachability problem for VASS, see e.g., [Reu90, Lam92, Mog01, Ler11].
- ⋆ Other decidability and computational complexity issues for reachability and model-checking problems, see e.g., [FS00, LS05, AH09, HKOW09].

Finally, the following research trends generate an increasing interest:

- ⋆ Transition closures of integer relations, see e.g. [BIK10].
- ⋆ The branching extension of VASS, leading to BVASS [VGL05, DJLL09, Laz10, Sch10a].
- ⋆ SMT solvers for model-checking infinite-state systems, see e.g., [GNRZ07, BFM$^+$10].
- ⋆ Relationships between counter automata and data logics [BMS$^+$06, DL06, DDG07, BL10].

Bibliography

[ABM99] C. Areces, P. Blackburn, and M. Marx. A road-map on complexity for hybrid logics. In J. Flum and M. Rodríguez-Artalejo, editors, *CSL'99*, volume 1683 of *Lecture Notes in Computer Science*, pages 307–321. Springer, 1999. (Cited on page 38)

[ABM01] C. Areces, P. Blackburn, and M. Marx. Hybrid logics: characterization, interpolation and complexity. *The Journal of Symbolic Logic*, 66(3):977–1010, 2001. (Cited on page 38)

[AD94] R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994. (Cited on page 5)

[AH94] R. Alur and Th. Henzinger. A really temporal logic. *Journal of the Association for Computing Machinery*, 41(1):181–204, 1994. (Cited on pages 10, 38, 110)

[AH09] M. Faouzi Atig and P. Habermehl. On Yen's path logic for Petri nets. In *RP'09*, volume 5797 of *Lecture Notes in Computer Science*, pages 51–63. Springer, 2009. (Cited on pages 54, 60, 115)

[AJ96] P. Abdulla and B. Jonsson. Verifying programs with unreliable channels. *Information and Computation*, 127(2):91–101, 1996. (Cited on pages 97, 115)

[Are00] C. Areces. *Logic Engineering: The Case of Description and Hybrid Logics*. PhD thesis, University of Amsterdam, 2000. (Cited on page 38)

[BB74] B. Baker and R. Book. Reversal-bounded multipushdown machines. *Journal of Computer and System Sciences*, 8:315–332, 1974. (Cited on pages 66, 70)

[BBF+01] B. Bérard, M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci, and Ph. Schnoebelen. *Systems and Software Verification, Model-Checking Techniques and Tools*. Springer-Verlag, 2001. (Cited on page 5)

[BBH+06] A. Bouajjani, M. Bozga, P. Habermehl, R. Iosif, and P. Moro nd T. Vojnar. Programs with lists are counter automata. In *CAV'06*, volume 4144 of *Lecture Notes in Computer Science*, pages 517–531. Springer, 2006. (Cited on pages 5, 18)

[BDM+06] M. Bojańczyk, C. David, A. Muscholl, Th. Schwentick, and L. Segoufin. Two-variable logic on data trees and XML reasoning. In *PODS'06*, pages 10–19. ACM, 2006. (Cited on page 45)

[BDR03] V. Bruyère, E. Dall'Olio, and J.F. Raskin. Durations, parametric model-checking in timed automata with presburger arithmetic. In *STACS'03*, volume 2607 of *Lecture Notes in Computer Science*, pages 687–698. Springer, 2003. (Cited on page 34)

[BEH95] A. Bouajjani, R. Echahed, and P. Habermehl. On the verification problem of nonregular properties for nonregular processes. In *LICS'95*, pages 123–133, 1995. (Cited on pages 13, 34)

[Ber80] L. Berman. The complexity of logical theories. *Theoretical Computer Science*, 11:71–78, 1980. (Cited on page 13)

[BFL+08] P. Bouyer, U. Fahrenberg, K. Larsen, N. Markey, and J. Srba. Infinite runs in weighted timed automata with energy constraints. In *FORMATS'08*, volume 5215 of *Lecture Notes in Computer Science*, pages 33–47. Springer, 2008. (Cited on page 18)

[BFLS06] S. Bardin, A. Finkel, E. Lozes, and A. Sangnier. From pointer systems to counter systems using shape analysis. *Proceedings of the 5th International Workshop on Automated Verification of Infinite-State Systems (AVIS'06)*, 2006. (Cited on page 5)

[BFM+10] M. Bersani, A. Frigeri, A. Morzenti, M. Pradella, M. Rossi, and P. San Pietro. Bounded reachability for temporal logic over constraint systems. In *TIME'10*. IEEE, 2010. To appear. (Cited on pages 34, 115)

[BFN04] S. Bardin, A. Finkel, and D. Nowak. Toward symbolic verification of programs handling pointers. In *Proceedings of the 3rd International Workshop on Automated Verification of Infinite-State Systems (AVIS'04)*, 2004. (Cited on page 18)

[BGI09] M. Bozga, C. Girlea, and R. Iosif. Iterating octagons. In *TACAS'09*, volume 5505 of *Lecture Notes in Computer Science*, pages 337–351. springer, 2009. (Cited on page 105)

[BGP97] T. Bultan, R. Gerber, and W. Pugh. Symbolic model checking of infinite state systems using Presburger arithmetic. In *CAV'97*, volume 1254 of *Lecture Notes in Computer Science*, pages 400–411. Springer, 1997. (Cited on page 34)

[BH91] F. Baader and P. Hanschke. A scheme for integrating concrete domains into concept languages. In *IJCAI'91*, pages 452–457, 1991. (Cited on page 32)

[BIK10] M. Bozga, R. Iosif, and F. Konečný. Fast acceleration of ultimately periodic relations. In *CAV'10*, volume 6174 of *Lecture Notes in Computer Science*, pages 227–242. springer, 2010. (Cited on pages 19, 21, 105, 115)

[BIL09] M. Bozga, R. Iosif, and Y. Lakhnech. Flat parametric counter automata. *Fundamenta Informaticae*, 91(2):275–303, 2009. (Cited on pages 20, 81, 105, 107)

[BJW01] B. Boigelot, S. Jodogne, and P. Wolper. On the use of weak automata for deciding linear arithmetic with integer and real variables. In *IJCAR'01*, volume 2083 of *Lecture Notes in Artificial Intelligence*, pages 611–625. Springer, 2001. (Cited on page 13)

[BKR96] M. Biehl, N. Klarlund, and T. Rauhe. Mona: Decidable arithmetic in practice. In *FTRTFT'96*, volume 1135 of *Lecture Notes in Computer Science*, pages 459–462. Springer, 1996. (Cited on page 13)

[BL10] M. Bojańczyk and S. Lasota. An extension of data automata that captures XPath. In *Proc. LICS'10*, 2010. to appear. (Cited on pages 45, 115)

[BM99] A. Bouajjani and R. Mayr. Model checking lossy vector addition systems. In *STACS'99*, volume 1563 of *Lecture Notes in Computer Science*, pages 323–333, 1999. (Cited on page 97)

[BMS+06] M. Bojańczyk, A. Muscholl, Th. Schwentick, L. Segoufin, and C. David. Two-variable logic on words with data. In *LICS'06*, pages 7–16. IEEE, 2006. (Cited on pages 5, 39, 45, 46, 61, 115)

[BMSS09] M. Bojańczyk, A. Muscholl, Th. Schwentick, and L. Segoufin. Two-variable logic on data trees and XML reasoning. *Journal of the Association for Computing Machinery*, 56(3), 2009. (Cited on page 46)

[Boi98] B. Boigelot. *Symbolic methods for exploring infinite state spaces*. PhD thesis, Université de Liège, 1998. (Cited on pages 5, 103)

[Bou02] P. Bouyer. A logical characterization of data languages. *Information Processing Letters*, 84(2):75–85, 2002. (Cited on pages 39, 45)

[BP08] F. Baader and R. Penaloza. Automata-based axiom pinpointing. In *IJCAR'08*, volume 5195 of *Lecture Notes in Computer Science*, pages 226–241. Springer, 2008. (Cited on page 28)

[BPT03] P. Bouyer, A. Petit, and D. Thérien. An algebraic approach to data languages and timed languages. *Information and Computation*, 182(2):137–162, 2003. (Cited on page 39)

[BSST08] C. Barrett, R. Sebastiani, S. Seshia, and C. Tinelli. *Satisfiability Modulo Theories*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, chapter 26, pages 825–885. IOS Press, 2008. (Cited on page 32)

[BT76] I. Borosh and L. Treybig. Bounds on positive integral solutions of linear diophantine equations. 55:299–304, 1976. (Cited on page 13)

[Büc62] R. Büchi. On a decision method in restricted second-order arithmetic. In *International Congress on Logic, Method and Philosophical Science'60*, pages 1–11, 1962. (Cited on pages 28, 29)

[CC00] H. Comon and V. Cortier. Flatness is not a weakness. In *CSL'00*, volume 1862 of *Lecture Notes in Computer Science*, pages 262–276. Springer, 2000. (Cited on pages 7, 13, 34, 37)

[Čer94] K. Čerāns. Deciding properties of integral relational automata. In *ICALP'94*, volume 820 of *Lecture Notes in Computer Science*, pages 35–46. Springer, 1994. (Cited on page 34)

[CGP00] E. Clarke, O. Grumberg, and D. Peled. *Model checking*. The MIT Press Books, 2000. (Cited on page 5)

[CJ98] H. Comon and Y. Jurski. Multiple counters automata, safety analysis and Presburger arithmetic. In *CAV'98*, volume 1427 of *Lecture Notes in Computer Science*, pages 268–279. Springer, 1998. (Cited on pages 5, 7, 19, 20, 21, 81, 100, 105, 107)

[CLM76] E. Cardoza, R.J. Lipton, and A.R. Meyer. Exponential space complete problems for Petri nets and Commutative Semigroups: Preliminary report. In *STOC'76*, pages 50–54. ACM, 1976. (Cited on pages 23, 60)

[Coo72] D. Cooper. Theorem proving in arithmetic without multiplication. *Machine Learning*, 7:91–99, 1972. (Cited on page 13)

[Cor02] V. Cortier. About the decision of reachability for register machines. *Theoretical Informatics and Applications*, 36(4):341–358, 2002. (Cited on page 106)

[CR04] C. Chitic and D. Rosu. On validation of XML streams using finite state machines. Technical Report CSRG-489, University of Toronto, 2004. (Cited on page 5)

[Dav09] C. David. *Analyse de XML avec données non-bornées*. PhD thesis, LIAFA, Université Paris VII, 2009. (Cited on pages 46, 61)

[dBvE01] F. de Boer and R. van Eijk. Decidable navigation logics for object structures. In L. Fribourg, editor, *CSL'01*, volume 2142 of *Lecture Notes in Computer Science*, pages 324–338. Springer, 2001. (Cited on page 38)

[DD07] S. Demri and D. D'Souza. An automata-theoretic approach to constraint LTL. *Information and Computation*, 205(3):380–415, 2007. (Cited on page 38)

[DDG07] S. Demri, D. D'Souza, and R. Gascon. Decidable temporal logic with repeating values. In *LFCS'07*, volume 4514 of *Lecture Notes in Computer Science*, pages 180–194. Springer, 2007. (Cited on pages 38, 115)

[Dem10] S. Demri. On Selective Unboundedness of VASS. In *Proceedings of the 12th International Workshop on Verification of Infinite State Systems (INFINITY'10)*, volume 39 of *Electronic Proceedings in Theoretical Computer Science*, pages 1–15, 2010. (Cited on pages 54, 68, 70, 83)

[DFGvD06] S. Demri, A. Finkel, V. Goranko, and G. van Drimmelen. Towards a model-checker for counter systems. In *ATVA'06*, volume 4218 of *Lecture Notes in Computer Science*, pages 493–507. Springer, 2006. (Cited on pages 101, 105, 107, 108, 109)

[DFGvD11] S. Demri, A. Finkel, V. Goranko, and G. van Drimmelen. Model-checking CTL* over flat presburger counter systems. *Journal of Non-Classical Logics*, 2011. Accepted for publication. (Cited on page 101)

[DFS98] C. Dufourd, A. Finkel, and Ph. Schnoebelen. Reset nets between decidability and undecidability. In *ICALP'98*, volume 1443 of *Lecture Notes in Computer Science*, pages 103–115. Springer, 1998. (Cited on pages 23, 96)

[DG08] S. Demri and R. Gascon. Verification of qualitative Z constraints. *Theoretical Computer Science*, 409(1):24–40, 2008. (Cited on pages 13, 38)

[dGGS04] Ph. de Groote, B. Guillaume, and S. Salvati. Vector addition tree automata. In *LICS'04*, pages 64–73. IEEE, 2004. (Cited on page 61)

[Dic13] L.E. Dickson. Finiteness of the odd perfect and primitive abundant numbers with n distinct prime factors. *Amer. Journal Math.*, pages 413–422, 1913. (Cited on pages 11, 24, 95)

[DIP01] Z. Dang, O. Ibarra, and P. San Pietro. Liveness verification of reversal-bounded multicounter machines with a free counter. In *FSTTCS'01*, volume 2245 of *Lecture Notes in Computer Science*, pages 132–143. Springer, 2001. (Cited on pages 83, 96)

[DJLL09] S. Demri, M. Jurdziński, O. Lachish, and R. Lazić. The covering and boundedness problems for branching vector addition systems. In *FST&TCS'09, Kanpur*. LZI, 2009. (Cited on pages 61, 115)

[DJS99] C. Dufourd, P. Jančar, and Ph. Schnoebelen. Boundedness of reset P/T nets. In *ICALP'99*, volume 1644 of *Lecture Notes in Computer Science*, pages 301–310. Springer, 1999. (Cited on page 96)

[DL06] S. Demri and R. Lazić. LTL with the freeze quantifier and register automata. In *LICS'06*, pages 17–26. IEEE, 2006. (Cited on pages 45, 97, 115)

[DL09] S. Demri and R. Lazić. LTL with the freeze quantifier and register automata. *ACM Transactions on Computational Logic*, 10(3), 2009. (Cited on pages 40, 48)

[DL10] S. Demri and D. Lugiez. Complexity of Modal Logics with Presburger Constraints. *Journal of Applied Logic*, 2010. To appear. (Cited on page 13)

[DLN05] S. Demri, R. Lazić, and D. Nowak. On the freeze quantifier in constraint LTL: decidability and complexity. In *TIME'05*, pages 113–121. IEEE, 2005. (Cited on page 39)

[DLS10] S. Demri, R. Lazić, and A. Sangnier. Model checking memoryful linear-time logics over one-counter automata. *Theoretical Computer Science*, 411(22–24):2298–2316, 2010. (Cited on page 42)

[DPK03] Z. Dang, P. San Pietro, and R. Kemmerer. Presburger liveness verification of discrete timed automata. *Theoretical Computer Science*, 299:413–438, 2003. (Cited on pages 18, 85, 86)

[DS10] S. Demri and A. Sangnier. When model checking freeze LTL over counter machines becomes decidable. In *FOSSACS'10*, volume 6014 of *Lecture Notes in Computer Science*, pages 176–190. Springer, 2010. (Cited on pages 42, 88, 90)

[DSV04] A. Deutsch, L. Sui, and V. Vianu. Specification and verification of data-driven web services. In *PODS'04*, pages 71–82, 2004. (Cited on page 35)

[DSZ10] G. Delzanno, A. Sangnier, and G. Zavattaro. Parameterized verification of ad hoc networks. In *CONCUR'10*, volume 6269 of *Lecture Notes in Computer Science*. Springer, 2010. (Cited on page 23)

[EF06] C. Eisner and D. Fisman. *A Practical Introduction to PSL*. Springer, 2006. (Cited on page 27)

[EFM99] J. Esparza, A. Finkel, and R. Mayr. On the verification of broadcast protocols. In *LICS'99*, pages 352–359, 1999. (Cited on page 18)

[Esp94] J. Esparza. On the decidability of model checking for several μ-calculi and Petri nets. In *CAAP'94*, volume 787 of *Lecture Notes in Computer Science*, pages 115–129. Springer, 1994. (Cited on page 96)

[Esp97] J. Esparza. Petri nets, commutative context-free grammars, and basic parallel processes. *Fundamenta Informaticae*, 31(13):13–26, 1997. (Cited on pages 21, 72, 81)

[Esp98] J. Esparza. Decidability and complexity of Petri net problems — an introduction. In *Advances in Petri Nets 1998*, volume 1491 of *Lecture Notes in Computer Science*, pages 374–428. Springer, Berlin, 1998. (Cited on pages 23, 54, 60)

[Fig09] D. Figueira. Satisfiability of downward XPath with data equality tests. In *PODS'09*, pages 197–206. ACM Press, 2009. (Cited on page 45)

[Fig10] D. Figueira. Forward-XPath and extended register automata on data-trees. In *ICDT'10*, pages 231–241. ACM Press, 2010. (Cited on page 45)

[Fin72] K. Fine. In so many possible worlds. *Notre Dame Journal of Formal Logic*, 13(4):516–520, 1972. (Cited on page 13)

[Fit02] M. Fitting. Modal logic between propositional and first-order. *Journal of Logic and Computation*, 12(6):1017–1026, 2002. (Cited on page 39)

[FKV04] E. Friedgut, O. Kupferman, and M. Vardi. Büchi complementation made tighter. In *ATVA'04*, volume 3299 of *Lecture Notes in Computer Science*, pages 64–78. Springer, 2004. (Cited on page 29)

[FL02] A. Finkel and J. Leroux. How to compose Presburger accelerations: Applications to broadcast protocols. In *FST&TCS'02*, volume 2256 of *Lecture Notes in Computer Science*, pages 145–156. Springer, Berlin, 2002. (Cited on pages 5, 21, 100, 101, 103, 105, 106, 107)

[FLS09] A. Finkel, E. Lozes, and A. Sangnier. Towards model-checking programs with lists. In *Infinity in Logic and Computation*, volume 5489 of *Lecture Notes in Artificial Intelligence*, pages 56–82. Springer, 2009. (Cited on page 18)

[FMP04] A. Finkel, P. McKenzie, and C. Picaronny. A well-structured framework for analysing Petri net extensions. *Information and Computation*, 195(1–2):1–29, 2004. (Cited on page 115)

[FR74] M. Fischer and M. Rabin. Super-exponential complexity of Presburger arithmetic. In *Complexity of Computation*, volume 7 of *SIAM-AMS proceedings*, pages 27–42. American Mathematical Society, 1974. (Cited on page 13)

[FR79] J. Ferrante and C. Rackoff. *The Computational Complexity of Logical Theories*, volume 718 of *Lecture Notes in Mathematics*. Springer, 1979. (Cited on page 13)

[FS00] A. Finkel and G. Sutre. Decidability of reachability problems for classes of two counters automata. In *STACS'00*, volume 2256 of *Lecture Notes in Computer Science*, pages 346–357. Springer, 2000. (Cited on pages 21, 81, 115)

[FS01] A. Finkel and Ph. Schnoebelen. Well-structured transition systems everywhere! *Theoretical Computer Science*, 256(1–2):63–92, 2001. (Cited on pages 5, 23, 97, 115)

[FS08] A. Finkel and A. Sangnier. Reversal-bounded counter machines revisited. In *MFCS'08*, volume 5162 of *Lecture Notes in Computer Science*, pages 323–334. Springer, 2008. (Cited on pages 21, 65, 68, 70, 82, 83)

[FS09] D. Figueira and L. Segoufin. Future-looking logics on data words and trees. In *MFCS'09*, volume 5734 of *Lecture Notes in Computer Science*, pages 331–343. Springer, 2009. (Cited on pages 40, 45)

[GaAR09] P. Ganty and R. Majumdar abd A. Rybalchenko. Verifying liveness for asynchronous programs. In *POPL'09*, pages 102–113. ACM, 2009. (Cited on page 23)

[GI81] E. Gurari and O. Ibarra. The complexity of decision problems for finite-turn multicounter machines. In *ICALP'81*, volume 115 of *Lecture Notes in Computer Science*, pages 495–505. Springer, 1981. (Cited on page 81)

[GNRZ07] S. Ghilardi, E. Nicolini, S. Ranise, and D. Zucchelli. Combination Methods for Satisfiability and Model-Checking of Infinite-State Systems. In *CAV'07*, volume 4603 of *Lecture Notes in Computer Science*, pages 362–378. Springer, 2007. (Cited on pages 32, 115)

[Gor94] V. Goranko. Temporal logic with reference pointers. In *1st International Conference on Temporal Logic (ICTL), Bonn*, volume 827 of *Lecture Notes in Artificial Intelligence*, pages 133–148. Springer, 1994. (Cited on page 38)

[Gor96] V. Goranko. Hierarchies of modal and temporal logics with references pointers. *Journal of Logic, Language, and Information*, 5:1–24, 1996. (Cited on page 38)

[GPSS80] D. Gabbay, A. Pnueli, S. Shelah, and J. Stavi. On the temporal analysis of fairness. In *7th Annual ACM Symposium on Principles of Programming Languages*, pages 163–173. ACM Press, 1980. (Cited on page 26)

[Grä88] E. Grädel. Subclasses of Presburger arithmetic and the polynomial-time hierarchy. *Theoretical Computer Science*, 56:289–301, 1988. (Cited on page 13)

[GS66] S. Ginsburg and E. Spanier. Semigroups, Presburger formulas and languages. *Pacific Journal of Mathematics*, 16(2):285–296, 1966. (Cited on page 14)

[GS92] S. German and P. Sistla. Reasoning about systems with many processes. *Journal of the Association for Computing Machinery*, 39(3):675–735, 1992. (Cited on page 96)

[GS09] A. Gaiser and S. Schwoon. Comparison of algorithms for checking emptiness on büchi automata. Technical report, arXiv:cs, October 2009. (Cited on page 28)

[Hab97] P. Habermehl. On the complexity of the linear-time mu-calculus for Petri nets. volume 1248 of *Lecture Notes in Computer Science*, pages 102–116. Springer, 1997. (Cited on page 95)

[Had01] S. Haddad. Décidabilité et complexité des problèmes de réseaux de Petri. In M. Diaz, editor, *Les réseaux de Petri — Modèles fondamentaux*, chapter 4, pages 119–158. Hermès, 2001. (Cited on page 60)

[HB91] B. Hollunder and F. Baader. Qualifying number restrictions in concept languages. In *KR'91*, pages 335–346, 1991. (Cited on page 13)

[Hen90] Th. Henzinger. Half-order modal logic: how to prove real-time properties. In *PODC'90*, pages 281–296. ACM, 1990. (Cited on pages 38, 39)

[HKOW09] C. Haase, S. Kreutzer, J. Ouaknine, and J. Worrell. Reachability in succinct and parametric one-counter automata. In *CONCUR'09*, volume 5710 of *Lecture Notes in Computer Science*, pages 369–383. Springer, 2009. (Cited on page 115)

[HLP90] E. Harel, O. Lichtenstein, and A. Pnueli. Explicit clock temporal logic. In *LICS'90*, pages 400–413. IEEE, 1990. (Cited on page 38)

[Hol97] G. Holzmann. The model checker SPIN. *IEEE Transactions on Software Engineering*, 23(5):279–295, 1997. (Cited on page 27)

[HP79] J. Hopcroft and J.J. Pansiot. On the reachability problem for 5-dimensional vector addition systems. *Theoretical Computer Science*, 8:135–159, 1979. (Cited on pages 21, 52, 60, 81, 107)

[HR87] R. Howell and L. Rosier. An analysis of the nonemptiness problem for classes of reversal-bounded multicounter machines. *Journal of Computer and System Sciences*, 34(1):55–74, 1987. (Cited on page 81)

[HR89] R. Howell and L.E. Rosier. Problems concerning fairness and temporal logic for conflict-free Petri nets. *Theoretical Computer Science*, 64:305–329, 1989. (Cited on page 96)

[Iba78] O. Ibarra. Reversal-bounded multicounter machines and their decision problems. *Journal of the Association for Computing Machinery*, 25(1):116–133, 1978. (Cited on pages 5, 21, 65, 66, 67, 68, 81, 82, 83)

[Iba79] O. Ibarra. Restricted one-counter machines with undecidable universe problems. *Mathematical Systems Theory*, 13(181):181–186, 1979. (Cited on page 86)

[ISD+00] O. Ibarra, J. Su, Z. Dang, T. Bultan, and A. Kemmerer. Counter machines: Decidable properties and applications to verification problems. In *MFCS'00*, volume 1893 of *Lecture Notes in Computer Science*, pages 426–435. Springer, 2000. (Cited on page 5)

[ISD+02] O. Ibarra, J. Su, Z. Dang, T. Bultan, and R. Kemmerer. Counter machines and verification problems. *Theoretical Computer Science*, 289(1):165–189, 2002. (Cited on pages 81, 86)

[Jan90] P. Jančar. Decidability of a temporal logic problem for Petri nets. *Theoretical Computer Science*, 74(1):71–93, 1990. (Cited on pages 95, 96)

[JL07] M. Jurdziński and R. Lazić. Alternation-free modal mu-calculus for data trees. In *LICS'07*, pages 131–140. IEEE, 2007. (Cited on page 45)

[Kam68] J. Kamp. *Tense Logic and the theory of linear order*. PhD thesis, UCLA, USA, 1968. (Cited on pages 26, 32)

[KKW10] A. Kaiser, D. Kroening, and Th. Wahl. Dynamic cutoff detection in parameterized concurrent programs. In *CAV'10*, volume 6174 of *Lecture Notes in Computer Science*, pages 645–659. Springer, 2010. (Cited on page 23)

[KM69] R. M. Karp and R. E. Miller. Parallel program schemata. *Journal of Computer and System Sciences*, 3(2):147–195, 1969. (Cited on pages 22, 23, 52, 55, 83)

[Kos82] R. Kosaraju. Decidability of reachability in vector addition systems. In *STOC'82*, pages 267–281, 1982. (Cited on pages 23, 60)

[Koz97] D. Kozen. *Automata and Computability*. Springer, 1997. (Cited on page 72)

[Kra02] M. Kracht. A new proof of a theorem by Ginsburg and Spanier. Published in "The Mathematics of Language", 2002. Manuscript, UCLA. (Cited on page 14)

[Lam92] J.L. Lambert. A structure to decide reachability in Petri nets. *Theoretical Computer Science*, 99:79–104, 1992. (Cited on pages 23, 60, 115)

[Laz06] R. Lazić. Safely freezing LTL. In *FSTTCS'06*, volume 4337, pages 381–392. LNCS, 2006. (Cited on page 40)

[Laz10] R. Lazić. The reachability problem for branching vector addition systems requires doubly-exponential space. *Information Processing Letters*, 110(17):740–745, 2010. (Cited on pages 61, 115)

[Ler03] J. Leroux. *Algorithmique de la vérification des systèmes à compteurs. Approximation et accélération. Implémentation de l'outil FAST.* PhD thesis, ENS de Cachan, France, 2003. (Cited on pages 13, 21, 100, 106, 107)

[Ler09] J. Leroux. The general vector addition system reachability problem by presburger inductive invariants. In *LICS'09*, pages 4–13. IEEE, 2009. (Cited on page 60)

[Ler11] J. Leroux. Vector Addition System Reachability Problem (a Short Self-contained Proof). In *POPL 2011*, 2011. To appear. (Cited on pages 60, 115)

[Lip76] R. J. Lipton. The reachability problem requires exponential space. Technical Report 62, Department of Computer Science, Yale University, 1976. (Cited on pages 23, 54, 60)

[LLT05] P. Lafourcade, D. Lugiez, and R. Treinen. Intruder deduction for AC-like equational theories with homomorphisms. In J. Giesl, editor, *RTA'05*, volume 3467 of *Lecture Notes in Computer Science*, pages 308–322. Springer, 2005. (Cited on page 5)

[LMP10] F. Laroussinie, A. Meyer, and E. Petonnet. Counting LTL. In *TIME'10*. IEEE, 2010. To appear. (Cited on page 34)

[LMS02] F. Laroussinie, N. Markey, and Ph. Schnoebelen. Temporal logic with forgettable past. In *LICS'02*, pages 383–392. IEEE, 2002. (Cited on page 38)

[LMSS92] P. Lincoln, J. Mitchell, A. Scedrov, and N. Shankar. Decision problems for propositional linear logic. *Annals of Pure and Applied Logic*, 56:239–311, 1992. (Cited on page 61)

[LP05] A. Lisitsa and I. Potapov. Temporal logic with predicate λ-abstraction. In *TIME'05*, pages 147–155. IEEE, 2005. (Cited on page 39)

[LP09] J. Leroux and G. Point. TaPAS: The Talence Presburger Arithmetic Suite. In *TACAS'09*, volume 5505 of *Lecture Notes in Computer Science*, pages 182–185. springer, 2009. (Cited on page 13)

[LS05] J. Leroux and G. Sutre. Flat counter systems are everywhere! In *ATVA'05*, volume 3707 of *Lecture Notes in Computer Science*, pages 489–503. Springer, 2005. (Cited on pages 21, 81, 107, 108, 115)

[Lut03] C. Lutz. Description logics with concrete domains—a survey. In *Advances in Modal Logics Volume 4*, pages 265–296. King's College Publications, 2003. (Cited on page 32)

[Lut04] C. Lutz. NEXPTIME-complete description logics with concrete domains. *ACM Transactions on Computational Logic*, 5(4):669–705, 2004. (Cited on page 32)

[May84] E.W. Mayr. An algorithm for the general Petri net reachability problem. *SIAM Journal of Computing*, 13(3):441–460, 1984. (Cited on pages 23, 60)

[May03] R. Mayr. Undecidable problems in unreliable computations. *Theoretical Computer Science*, 297(1–3):337–354, 2003. (Cited on page 97)

[McM93] K. McMillan. *Symbolic Model Checking*. Kluwer Academic Publishers, 1993. (Cited on pages 5, 27)

[Min67] M. Minsky. *Computation: Finite and Infinite Machines*. Prentice Hall, Englewood Cliffs, NJ, 1967. (Cited on pages 9, 10)

[Mog01] V. Mogbil. *Sémantiques des phases: réseaux de preuves et divers problèmes de décision en logique linéaire*. Thèse de doctorat, Université de la Méditerranée - Aix-Marseille II, 2001. (Cited on pages 60, 115)

[MP79] Z. Manna and A. Pnueli. The modal logic of programs. In *ICALP'79*, volume 71 of *Lecture Notes in Computer Science*, pages 385–409. Springer, 1979. (Cited on page 26)

[MP92] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification*. Springer-Verlag, New York, 1992. (Cited on page 39)

[MP95] Z. Manna and A. Pnueli. *Temporal verification of reative systems: safety*. Springer-Verlag, New York, 1995. (Cited on page 32)

[MS77] A. Mandel and I. Simon. On finite semigroups of matrices. *Theoretical Computer Science*, 5(2):101–111, 1977. (Cited on page 103)

[Muk09] M. Mukund. *Modern Applications of Automata Theory*, chapter Finite-state automata on infinite inputs. IIsc Research Monographs. World Scientific, 2009. To appear. (Cited on page 29)

[Opp78] D. Oppen. A $2^{2^{2^{pn}}}$ upper bound on the complexity of Presburger arithmetic. *Journal of Computer and System Sciences*, 16(3):323–332, 1978. (Cited on page 13)

[OW06] J. Ouaknine and J. Worrell. On Metric temporal logic and faulty Turing machines. In *FOSSACS*, volume 3921 of *Lecture Notes in Computer Science*, pages 217–230. Springer, 2006. (Cited on page 97)

[Pap81] Chr. Papadimitriou. On the complexity of integer programming. *Journal of the Association for Computing Machinery*, 28(4):765–768, 1981. (Cited on page 13)

[Par66] R. Parikh. On context-free languages. *Journal of the Association for Computing Machinery*, 13(4):570–581, 1966. (Cited on pages 14, 70, 72, 82, 107)

[Pet81] J.L. Peterson. *Petri Net Theory and the Modelling of Systems*. Prentice-Hall, 1981. (Cited on page 48)

[PL09] M. Praveen and K. Lodaya. Modelchecking counting properties of 1-safe nets with buffers in paraPSPACE. In *FST&TCS'09, Kanpur*, pages 347–358. LZI, 2009. (Cited on page 61)

[Pnu77] A. Pnueli. The temporal logic of programs. In *FOCS'77*, pages 46–57. IEEE Computer Society Press, 1977. (Cited on page 26)

[Pnu79] A. Pnueli. The temporal semantics of concurrent programs. In *International Symposium on Semantics of Concurrent Computation 1979*, volume 70 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 1979. (Cited on page 26)

[Pre29] M. Presburger. Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. *Comptes Rendus du premier congrès de mathématiciens des Pays Slaves, Warszawa*, pages 92–101, 1929. (Cited on pages 11, 13)

[Rac78] C. Rackoff. The covering and boundedness problems for vector addition systems. *Theoretical Computer Science*, 6(2):223–231, 1978. (Cited on pages 23, 54, 55, 60, 61, 95)

[Reu90] C. Reutenauer. *The mathematics of Petri nets*. Masson and Prentice, 1990. (Cited on pages 5, 23, 60, 74, 115)

[RGL01] M. Roger and J. Goubault-Larrecq. Log auditing through model-checking. In *CSFW'01*, pages 220–236. IEEE, 2001. (Cited on page 34)

[Rog67] H. Rogers. *Theory of recursive functions and effective computability*. McGraw-Hill, 1967. (Cited on page 10)

[RR98] W. Reisig and G. Rozenberg, editors. *Lectures on Petri Nets I: Basic Models*, volume 1491 of *Lecture Notes in Computer Science*. Springer, 1998. (Cited on page 48)

[San08] A. Sangnier. *Vérification de systèmes avec compteurs et pointeurs*. Thèse de doctorat, LSV, ENS Cachan, France, 2008. (Cited on pages 82, 83)

[Sav70] W.J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4(2):177–192, 1970. (Cited on pages 55, 57)

[SC85] P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logics. *Journal of the Association for Computing Machinery*, 32(3):733–749, 1985. (Cited on page 32)

[Sch02] Ph. Schnoebelen. Verifying lossy channel systems has nonprimitive recursive complexity. *Information Processing Letters*, 83(5):251–261, 2002. (Cited on pages 96, 97)

[Sch07] T. Schuele. *Verification of Infinite State Systems Using Presburger Arithmetic*. PhD thesis, Department of Computer Science, University of Kaiserslautern, Germany, Kaiserslautern, Germany, March 2007. (Cited on page 13)

[Sch10a] S. Schmitz. On the Computational Complexity of Dominance Links in Grammatical Formalisms. In *ACL'10*, pages 514–524. Association for Computational Linguistics, 2010. (Cited on pages 61, 115)

[Sch10b] Ph. Schnoebelen. Lossy counter machines undecidability cheat sheet. In *RP'10*, volume 6227 of *Lecture Notes in Computer Science*. Springer, 2010. To appear. (Cited on pages 96, 97)

[Sch10c] Ph. Schnoebelen. Revisiting Ackermann-hardness for lossy counter machines and reset Petri nets. In *MFCS'10*, Lecture Notes in Computer Science. Springer, 2010. To appear. (Cited on page 97)

[Seg06] L. Segoufin. Automata and logics for words and trees over an infinite alphabet. In *CSL'06*, volume 4207 of *Lecture Notes in Computer Science*, pages 41–57. Springer, 2006. (Cited on page 45)

[SSM07] H. Seidl, Th. Schwentick, and A. Muscholl. Counting in trees. *Texts in Logic and Games*, 2:575–612, 2007. (Cited on page 13)

[SSMH04] H. Seidl, Th. Schwentick, A. Muscholl, and P. Habermehl. Counting in trees for free. In *ICALP'04*, volume 3142 of *Lecture Notes in Computer Science*, pages 1136–1149. Springer, 2004. (Cited on page 14)

[Str94] H. Straubing. *Finite Automata, Formal Logic, and Circuit Complexity*. Progress in Theoretical Computer Science. Birkhäuser, 1994. (Cited on page 45)

[SVW87] A. Sistla, M. Vardi, and P. Wolper. The complementation problem for Büchi automata with applications to temporal logic. *Theoretical Computer Science*, 49:217–237, 1987. (Cited on page 30)

[Tar53] A. Tarski. *Undecidable Theories*. Studies in Logic and the Foundations of Mathematics. North-Holland, 1953. In collaboration with A. Mostowski. and R. Robinson. (Cited on page 13)

[Tho99] W. Thomas. Complementation of Büchi automata revisited. In *Jewels are Forever, Contributions on Theoretical Computer Science in Honor of Arto Salomaa*, pages 109–122. Springer, 1999. (Cited on page 29)

[TL10] A. To and L. Libkin. Algorithmic metatheorems for decidable LTL model checking over infinite systems. In *FOSSACS'10*, volume 6014 of *Lecture Notes in Computer Science*, pages 221–236. Springer, 2010. (Cited on page 94)

[Tur36] A. Turing. On computable numbers, with an application to the entscheidungsproblem. In *Proceedings of the London Mathematical Society*, number 42 in 2, pages 230–265, 1936. (Cited on page 10)

[Urq99] A. Urquhart. The Complexity of Decision Procedures in Relevance Logic II. *The Journal of Symbolic Logic*, 64(4):1774–1802, 1999. (Cited on pages 61, 97)

[Var88] M. Vardi. A temporal fixpoint calculus. In *15th Annual ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages, San Diego*, pages 250–259. ACM, 1988. (Cited on page 29)

[VGL05] K. N. Verma and Jean Goubault-Larrecq. Karp-Miller Trees for a Branching Extension of VASS. *Discrete Mathematics and Theoretical Computer Science*, 7:217–230, 2005. (Cited on pages 61, 115)

[VVN81] R. Valk and G. Vidal-Naquet. Petri nets and regular languages. *Journal of Computer and System Sciences*, 23:299–325, 1981. (Cited on pages 53, 83)

[VW94] M. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115:1–37, 1994. (Cited on page 30)

[Wol83] P. Wolper. Temporal logic can be more expressive. *Information and Control*, 56:72–99, 1983. (Cited on page 29)

[WZ00] F. Wolter and M. Zakharyaschev. Spatio-temporal representation and reasoning based on RCC-8. In *KR'00*, pages 3–14, 2000. (Cited on page 38)

[Yen92] H.-C. Yen. A unified approach for deciding the existence of certain net paths. *Information and Computation*, 96:119–137, 1992. (Cited on pages 54, 60)

[ZL03] S. Dal Zilio and D. Lugiez. XML schema, tree logic and sheaves automata. In *RTA'03*, volume 2706 of *Lecture Notes in Computer Science*, pages 246–263. Springer, 2003. (Cited on page 13)