UNIVERSITÉ DE BORDEAUX

ÉCOLE DOCTORALE DE MATHÉMATIQUES ET INFORMATIQUE DE BORDEAUX

Habilitation à Diriger des Recherches

Jérôme Leroux

PRESBURGER COUNTER MACHINES

Préparée au LaBRI, équipe MF

Jury :

Rapporteurs :

Javier Esparza	-	Professeur Technische Universität München, Allemagne
Denis Lugiez	-	Professeur Université de Provence / LIF
Jean François Raskin	-	Professeur Université Libre de Bruxelles, Belgique

Examinateurs :

Hubert Comon	-	Professeur ENS Cachan / LSV
Nicolas Halbwachs	-	Directeur de Recherche CNRS / Verimag
Anca Muscholl	-	Professeur Université de Bordeaux-1 / LaBRI
Jean Éric Pin	-	Directeur de Recherches CNRS / LIAFA
Philippe Schnoebelen	-	Directeur de Recherche CNRS / LSV

Acknowledgments

Last thing to do :-)

Contents

Chapter 1

Introduction

Critical, embedded or real-time systems have many applications from the control (automotive and avionic industry), to signal processing (audio or video data, GPS navigation systems) and communication (cell phones, internet). Usually these systems are *infinite* since they are based on potentially *unbounded variables*: integer variables (program counters, number of processes connected to a remote server) or real numbers (clocks modeling elapsing time), communication channels, stacks, memory heap, and so on. Whereas the verification of finite state systems is algorithmically decidable (and there exist efficient tools), the verification of infinite state systems is not a sinecure. In fact, even the easiest properties are undecidable in general. For instance, Minsky proved the undecidability of the reachability problem for machines with just two counters that can be incremented, decremented or tested to zero[Minsky 1967].

A result of complexity or decidability is very useful to understand the theoretical limits of a problem. However for important problems with human or economic implications, even a result of undecidability cannot be the last word of the solution. In practice the considered instances of the general problem are not the most difficult ones. For these practical problems we are interested in the most efficient algorithms.

In this document we present results on the *automatic verification of infinite state systems* with the main objective to overcome undecidable results with tools based on strong theoretical foundations. In the sequel, we provide a brief summary of our contributions in the following research directions:

- In Section 1.2 : The Presburger arithmetic, a decidable formalism for denoting complex properties on the configurations of systems manipulating counters.

- In Section 1.3 : Tools and theoretical frameworks for the verification of infinite states systems.

- In Section 1.4 : The reachability problem for vector addition systems, a class of systems equivalent to the Petri nets.

These sections correspond roughly to the outline of this document. In these sections research results related to our work are briefly presented. A more detailed related work is presented in each chapter. In Section 1.1 we present the most significant results.

1.1 Most Significant Results

From the most significant results presented in this document, we cite the polynomial time characterization of the arithmetic automata encoding the solutions of Presburger formulas, the separation of unreachable configurations of Petri nets by inductive invariants definable in the Presburger arithmetic, and the tools PRESTAF and BATOF distributed in the Talence Presburger Arithmetic Suite TAPAS, and the model-checker FAST. These tools are available on the website `http://altarica.labri.fr`.

Arithmetic automata and Presburger formulas

Finite state automata provide a simple way for deciding the satisfiability of a Presburger formula. Since [Büchi 1962] we know that for the positional notation of natural numbers and vectors of natural numbers, the solutions of Presburger formulas are encoded by regular languages that can be effectively represented by finite state automata. Whereas there exist efficient algorithms computing arithmetic automata from Presburger formulas [Wolper 2000], the inverse problem of decoding arithmetic automata to Presburger formulas is much more difficult. Since in general an arithmetic automaton may encode a set that is not definable in the Presburger arithmetic (for instance the power of the basis of decomposition), the decoding problem is linked to deciding if an arithmetic automaton represents a Presburger set. The decidability of this problem was first established by Muchnik in 1991 with a 4-exponential time algorithm[Muchnik 1991, Muchnik 2003]. In the book [Allouche 2003] published in 2003, Jean-Paul Allouche and Jeffrey Shallit conjectured that the problem in dimension one should be solvable in polynomial time. In 2004, Louis Latour and Denis Lugiez provides exponential time and double-exponential time algorithm for deciding the membership of sets encoded by arithmetic automata into some subclasses of Presburger sets [Latour 2004, Lugiez 2004]. In 2005, we solved the general problem with a polynomial time algorithm[Leroux 2005a]. More precisely we introduced a polynomial time criterion for deciding if an arithmetic automaton encodes a set of vectors definable in the Presburger arithmetic. Moreover, in the positive case we showed how to decode in polynomial time an arithmetic automaton to a Presburger formula. We spent the 2006 year to improve the solution in order to get efficient algorithms in practice. In a 134 pages long internal report [Leroux 2006], we provided the complete solution in full detail. The algorithm is based on many algorithmic solutions to difficult problems and a surprising application of the Stallings' folding process to extract periodicities from the strongly connected components of arithmetic automata [Stallings 1983, Touikan 2006]. During the year 2007, with Gérald Point (LaBRI, Bordeaux), we implemented these algorithms in the tool BATOF. This tool is very efficient ! In fact the decoding of an arithmetic automaton with more than 100k states takes less than one minute on a standard computer. BATOF is the unique tool that implements a decoding algorithm. By combining BATOF with a tool computing arithmetic automata from Presburger formulas we obtain the unique

implementation of a simplification and normalization procedure for Presburger formulas based on the minimization procedure for automata. These encouraging results were published in [Leroux 2009b] and provide a framework for solving the Presburger arithmetic with hybrid representations combining formulas and automata.

The reachability problem for Petri nets

Petri Nets is one of the most popular formal models for the representation and the analysis of parallel processes [Esparza 1994]. The reachability problem is central since many computational problems (even outside the realm of parallel processes) reduce to this problem[Bojańczyk 2006, Demri 2009, Figueira 2009]. The reachability problem is difficult and its decidability remained open during 20 years. Sacerdote and Tenney provided in [Sacerdote 1977] a partial proof of decidability. The proof was completed in 1981 by Mayr [Mayr 1981b] and simplified by Kosaraju [Kosaraju 1982] from [Sacerdote 1977, Mayr 1981b]. Ten years later [Lambert 1992], Lambert provided a further simplified version based on [Kosaraju 1982]. Nowadays, no decision procedure are implemented. In fact the known algorithm is conceptually difficult (to be understood and to be implemented) and with high computational complexity bound (Ackermann). Since the proof of decidability of the reachability problem in the early 80's, very few results improved the original solution except the book of Reutenauer [Reutenauer 1990] and the Lambert's structures [Lambert 1992]. Some extensions of the problem were shown to be decidable by Jançar in [Jančar 1990a] and Reinhardt in [Reinhardt 2005] by considering richer properties or more expressive models. Recently we obtained a significant progress by introducing in [Leroux 2009a] a simple algorithm for deciding the reachability problem based on inductive invariants definable in the Presburger arithmetic that separate unreachable configurations. Whereas these invariants are obtained from the Lambert's structure in the proof published in 2009, we provided in [Leroux 2011a] a new shorter direct proof of the existence of these inductive invariants. This new proof introduces the central notion of almost semilinear sets, a class of sets inspired by the characterization of Presburger sets by semilinear sets [Ginsburg 1966]. The class of almost semilinear sets provides a simple way to capture the reachability sets of vector addition systems, and in particular those with reachability sets not definable in the Presburger arithmetic. These results open many research problems.

1.2 Presburger Arithmetic

In the program analysis and verification field, one often faces the problem of finding a suitable formalism for expressing the constraints to be satisfied by the system configurations. Ideally, this formalism has to be decidable, while still remaining expressive enough for handling the class of constraints needed by the application. An example of such a formalism is the first order logic over the natural numbers with the addition which is widely used for reasoning about programs manipulating counters. This logic, also known as the *Presburger arithmetic* is indeed decid-

able [Presburger 1929, Büchi 1962], yet expressive enough for describing arbitrary linear constraints as well as discrete periodicities.

Finite state automata provide a simple way for deciding the satisfiability of a Presburger formula. Since [Büchi 1962] we know that for the positional notation of natural numbers and vectors of natural numbers, the solutions of Presburger formulas are encoded by regular languages that can be effectively represented by finite state automata. The satisfiability of a Presburger formula then simply reduces to build inductively an automaton encoding the solutions of the formula with a bottom-up algorithm and then check the emptiness of the language accepted by this automaton. In tools that manipulate sets definable in the Presburger arithmetic, these automata, called *arithmetic automata* are used as symbolic representations encoding sets of solutions of Presburger formulas. These automata are canonically associated to the sets of solutions and not on the syntax of the formulas. In fact, the arithmetic automata encoding the solutions of two equivalent Presburger formulas recognize the same regular language. Hence the minimization procedure for arithmetic automata acts like a simplification procedure for the Presburger arithmetic.

Arithmetic automata and Presburger formulas

Unfortunately, in some applications, arithmetic automata may be too large to be explicitly computed by a tool. From a theoretical viewpoint this problem cannot be avoided since Leonard Berman shown in [Berman 1977] that the satisfiability of Presburger formulas is complete for the class of problems that can be solved with alternating Turing machines in time double exponential with a linear number of alternations. Nevertheless, in practice we are looking for the most efficient algorithms for the instances we are interested in (and these instances have no reason to be the most difficult ones as already mentioned). Experimentally, we showed that the arithmetic automata and the Presburger formulas provide two symbolic representations efficient for incomparable classes of Presburger sets[Leroux 2009b]. In order to obtain efficient symbolic representations for larger classes of Presburger sets, we are interested in combining formulas and automata in an hybrid symbolic representation. Recently, with Florent Bouchy and Alain Finkel (LSV, Cachan) in [Bouchy 2008] and with Bernard Boigelot and Julien Brusten (Institut Montéfiore, Liège, Belgium) in [Boigelot 2009] we published preliminary results in this field. In this last paper we showed that an extension of the Cohbam's Theorem sheds lights on the internal structure of the strongly connected components of arithmetic automata. From this result we deduced a shape of hybrid representation with strongly connected components represented implicitly with formulas.

In order to implement an hybrid symbolic representation manipulating formulas and automata we are interested in the effective encoding and decoding of Presburger formulas to and from arithmetic automata. Whereas there exist efficient algorithms computing arithmetic automata from Presburger formulas [Wolper 2000], the inverse problem of decoding arithmetic automata to Presburger formulas is much more difficult. Since in general an arithmetic automaton may encode a set that is

not definable in the Presburger arithmetic (for instance the power of the basis of decomposition), the decoding problem is linked to deciding if an arithmetic automaton represents a Presburger set. The decidability of this problem was first established by Muchnik in 1991. The solution is based on the self-definability property satisfied by the Presburger arithmetic[Muchnik 1991, Muchnik 2003]. Intuitively for each dimension d there exists an effectively computable Presburger formula with a special uninterpreted symbol of arity d such that the formula is valid if and only if the interpretation of the special symbol is a Presburger set. From this property, deciding if an arithmetic automaton denotes a Presburger set reduces to compute another arithmetic automaton encoding the solutions of the self-definability formula introduced by Muchnik. The computational complexity of this algorithm is 4-exponential in time. In the book [Allouche 2003] published in 2003, Jean-Paul Allouche and Jeffrey Shallit conjectured that the problem in dimension one should be solvable in polynomial time. In 2004, Louis Latour and Denis Lugiez provides exponential time and double-exponential time algorithm for deciding the membership of sets encoded by arithmetic automata into some subclasses of Presburger sets [Latour 2004, Lugiez 2004]. In 2005, we solved the general problem with a polynomial time algorithm[Leroux 2005a]. More precisely we introduced a polynomial time criterion for deciding if an arithmetic automaton encodes a set of vectors definable in the Presburger arithmetic. Moreover, in the positive case we showed how to decode in polynomial time an arithmetic automaton to a Presburger formula. We spent the 2006 year to improve the solution in order to get efficient algorithms in practice. In a 134 pages long internal report [Leroux 2006], we provided the complete solution in full detail. The algorithm is based on many algorithmic solutions to difficult problems and a surprising application of the Stallings' folding process to extract periodicities from the strongly connected components of arithmetic automata [Stallings 1983, Touikan 2006]. During the year 2007, with Gérald Point (LaBRI, Bordeaux), we implemented these algorithms in the tool Batof. This tool is very efficient ! In fact the decoding of an arithmetic automaton with more than 100k states takes less than one minute on a standard computer. Batof is the unique tool that implements a decoding algorithm. By combining Batof with a tool computing arithmetic automata from Presburger formulas we obtain the unique implementation of a simplification and normalization procedure for Presburger formulas based on the minimization procedure for automata. These encouraging results were published in [Leroux 2009b] and they provide the premises for new hybrid representations based on automata and formulas for manipulating efficiently larger classes of Presburger sets.

> A polynomial time criterion for deciding if an arithmetic automaton encodes a Presburger set and a polynomial time algorithm decoding arithmetic automata to Presburger formulas. These algorithms are implemented in Batof.

Whereas there exists a polynomial time criterion for deciding if an arithmetic automaton encodes a Presburger set, the structure of these arithmetic automata

is not known. For the time being, we do not have any structural test for deciding if an arithmetic automaton encodes a Presburger set. We are interested in such a test in order to understand theoretical complexity bounds about arithmetic automata (at worst 3-exponential [Klaedtke 2004, Durand-Gasselin 2010]) compared to the general class of arithmetic automata (Ackermann in the worst case [Büchi 1962, Bruyère 1994, Blumensath 2000]). We think that a structural test should exist since in [Leroux 2008b] we proved that the arithmetic automata obtained from minimal deterministic ones encoding Presburger sets by modifying the initial states and the sets of accepting states still encode Presburger sets. We are interested in characterizing the structure of arithmetic automata encoding Presburger sets based on properties satisfied by the syntactic monoids [Pin 1996]. Nowadays, this problem is open.

Extracting geometrical properties from arithmetic automata

Decoding arithmetic automata to Presburger formulas may produce very complex formulas that can be too precise for the considered problem. Moreover the decoding can only be applied on arithmetic automata encoding Presburger sets. In [Finkel 2005, Leroux 2008a] we provided two algorithms computing automatically the convex hull of sets denoted by arithmetic automata (which is always a polyhedron with rational constraints). Whereas the algorithm presented in 2005 is based on an effective enumeration of elements in an exponentially large set, the last one presented in 2008 does not require such an enumeration. Even if these two algorithms have the same worst case computational complexity (exponential), in practice there is no hope of termination with an enumerative algorithm. In the future we are interested in implementing the 2008 algorithm in the tool Batof.

> The convex hull of sets encoded by arithmetic automata is polyhedral with rational constraints. Moreover this convex hull is effectively computable in exponential time; this computational complexity bound is tight.

Arithmetic automata

With Gérald Point (LaBRI, Bordeaux) and Jean Michel Couveur (LIFO, Orléans) we implemented the tool Presatf that implements a decision procedure for the Presburger arithmetic based on the arithmetic automata. Compared to other automata packages, Prestaf is the unique tool that symbolically represent in memory automata in a strong canonical form allowing maximal sharing[Couvreur 2004] : That means we can check if two automata recognize the same language in constant time (this problem just reduces to check the equality of two pointers). This complexity result is central in the implementation of Prestaf since it provides a simple way to implement cache algorithms. No other tool manipulating automata implements such a cache. In fact, since the equality is usually implemented in linear time, it is not possible to retrieve efficiently already computed operations on automata or

sub-automata. Experimentally, even if PRESTAF is not fully optimized, this tool is very efficient for manipulating arithmetic automata compared to MONA, LASH, and LIRA. The tool PRESTAF is presented in [Bardin 2006, Leroux 2009b]. The tool PRESTAF has many applications even outside the Presburger arithmetic since it is basically an automata package library. In the future we are interested in promoting this tool.

The tool PRESTAF for manipulating Presburger sets with automata in strong canonical forms allowing maximal sharing.

Tools for deciding the Presburger arithmetic

A lot of tools, called *solvers*, implement decision procedures for the Presburger arithmetic. Some of them work directly with Presburger formulas and apply quantification elimination algorithms like the tool OMEGA [Pugh 1992a] whereas others are based on finite state automata encoding the solutions of Presburger formulas like MONA or our tool PRESTAF [Leroux 2009b]. These list can be completed with solvers for extended logics like LASH and LIRA [Becker 2007] that implement decision procedures for the first order logic over the integers and the reals with the addition function thanks to Buchi automata encoding reals by infinite words. Selecting the most efficient solver for a given application can be difficult. This choice can change dramatically the practical performance of the application. Since solvers have incompatible application programming interfaces (API), a particular one must be first selected prior the implementation of the application; change afterward requires additional development effort. We implemented with Gérald Point (LaBRI, Bordeaux) and Sébastien Bardin (CEA, Saclay) the Generic Presburger Application Programming Interface GENEPI [Bardin 2006] to solve this problem. This tool provides a simplified and uniform interface between tools implementing solvers for the Presburger arithmetic and tools that need these solvers. The connection between applications and solvers is realized transparently using dynamically loaded modules (plugins) specified at the execution time. This way the choice of a solver can be postponed after the implementation of an application based on GENEPI. The tool GENEPI can be used in two ways, either by developers that look for Presburger arithmetic libraries but also by developers interested in evaluating a new solver against existing ones. This last way was used by Felix Klaedtke (ETH, Zurich, Switzerland) to evaluate his new solver LIRA [Becker 2007]. The tool GENEPI is presented in [Bardin 2006, Leroux 2009b] and it provides a simple way to benchmark the solvers MONA, LASH, LIRA, PRESTAF, OMEGA, and PPL.

The API GENEPI that provides a uniform and simplified interface between solvers implementing decision procedures for the Presburger arithmetic and tools using these procedures.

1.3 Reachability Problems

The verification problem of a reachability property often reduces to the computation of a precise enough inductive invariant in a decidable formalism. From a theoretical viewpoint, a simple way for computing such an invariant when it exists consists in enumerating the properties in the decidable formalism until we discover one that denotes an inductive invariant precise enough for proving the safety property. In practice such an algorithm does not terminate since only a few number of properties can be analyzed in a reasonable amount of computational time. The framework of the verification of safety properties we consider is the following one. We assume that we have a decidable formalism and we consider instances of the reachability problems that can be solved with inductive invariants in this formalism. We are looking for efficient algorithms for computing such an invariant in practice.

In this document we focus on the class of infinite state systems manipulating counters. Note however that algorithms presented in this document can be generalized to systems manipulating other data values. More formally, we introduce the class of *Presburger counter machines* corresponding to machines with a finite set of control locations equipped with a finite set of counters in such a way that the one step reachability relation is definable in the Presburger arithmetic. This class is rich enough to capture many other classes of machines manipulating counters. For instance it contains the Minsky machines and as a direct consequence the reachability problem is undecidable.

Usually the instances (c, M, c') of the reachability problems where c, c' are two configurations of a Presburger counter machine M solved by algorithms correspond either to the case c' is reachable from c or there exists a certificate of safety for (c, M, c') proving that c' is not reachable from c thanks to an inductive invariant definable in the Presburger arithmetic. For instance methods based on *abstract interpretations* for many numerical abstract domains like the convex polyhedral [Cousot 1978], the intervals [Cousot 1977], the DBM, the octagons [Miné 2001], and so on [Péron 2007] compute precise enough abstract values for proving safety properties. Such an abstract value can be directly translated into a Presburger formula denoting an inductive invariant.

In this section we present three methods for deciding the reachability problem for Presburger counter machines : (1) the computation of reachability sets based on *acceleration techniques*, (2) the extension of acceleration techniques in an *abstract domain*, and (3) the model-checking based on *Craig interpolants* and *SMT-solvers*. We do not pretend to be exhaustive since these three methods only correspond to the ones for which we provided contributions. All theses methods introduce algorithms without any termination guaranty in the general case. In fact, the reachability problem is undecidable even for the Minsky machines. However, in practice these methods provide efficient algorithms for some instances. Such an algorithm that work well in practice but without termination guaranty is called a "*good semi-algorithm*".

Acceleration in symbolic model-checking

The reachability sets of initialized systems provide the most precise inductive invariants for deciding reachability problems. For finite-state systems, an inductive algorithm provides a simple way for computing these invariants. The computation can be performed with an explicit enumeration of the reachable configurations or more efficiently with a symbolic representation that encodes with small structures large sets of configurations. For instance the BDDs provide a symbolic representation based on acyclic graphs for encoding solutions of propositional formulas[Burch 1990]. For infinite-state systems, even with a symbolic representation that can encode infinite sets of configurations, a simple iterative computation of the reachability set usually does not terminate. With Sébastien Bardin (CEA, Saclay), Alain Finkel (LSV, ENS-Cachan), Philippe Schnoebelen (LSV, ENS-Cachan) and Grégoire Sutre (LaBRI, Bordeaux) we worked on the iterative computation of the reachability sets of infinite state systems with acceleration techniques in order to help the termination. The acceleration techniques consists in computing in a decidable formalism the precise effect of iterating control flow cycles. For instance, in the case of Presburger counter machines, the acceleration of a loop that increments a counter is the relation that increase this counter arbitrary. Since such a relation is expressible in the Presburger arithmetic, it becomes possible to capture in a finite number of computational steps an infinite set of reachable configurations. Whereas the Presburger counter systems are well studied, in 2000 no tool was able to compute automatically the reachability sets of initialized Presburger counter systems. The tools Lash developed by Bernard Boigelot (Liège, Belgium) and TReX developed by the Ahmed Bouajjani team (LIAFA, Paris) were able to compute the transitive closure of the binary relation associated to a sequence of transitions; However at that time these tools did not provide automatic procedure for computing the reachability sets. With Sébastien Bardin (CEA, Saclay), we implemented the accelerated symbolic model-checker Fast. Experimentally, we manage to compute automatically the reachability set of 40 Presburger counter machines : Java program abstractions, communication protocols, industrial systems, and the TTP/C protocol with 2 fault[Kopetz 1994]. These results are presented in [Bardin 2004, Bardin 2003].

> The symbolic model-checker Fast implementing acceleration techniques.

Acceleration and abstract interpretation

Unfortunately the reachability sets may be not definable in the selected formalism whereas there exist inductive invariants in this formalism separating some unreachable configurations. For instance the reachability set of an initialized vector addition system is in general not definable in the Presburger arithmetic [Hopcroft 1979] but any reachability problem can be solved with inductive invariants definable in the Presburger arithmetic [Leroux 2011a, Leroux 2009a]. This result (detailed in the next section) shows that we should try to compute a precise enough induc-

tive invariant depending on the safety property we are interested in. The abstract interpretation[Cousot 1977] provides a nice framework for performing such a computation. The lost of preciseness in the computation of an inductive invariant by abstract interpretation comes from two factors. First, the choice of the abstract domain limits the form of the invariants that can be computed. For instance with the polyhedral convex sets, it is not possible to directly determined an invariant that requires a disjunction. Once the abstract domain is fixed, another lost of preciseness is introduced during the Kleene fix point computation. In fact, in order to enforce the termination of the iterative algorithm a widening operator is applied. The application moment of this operator must be carefully selected since otherwise inductive invariants can be too coarse for proving the property. With Grégoire Sutre (LaBRI, Bordeaux), we studied the lost of preciseness due to the abstract domain [Leroux 2007a, Leroux 2007b]. Intuitively, we replaced the widening operator with acceleration techniques in order to understand the theoretical limits of an abstract domain still preserving some termination guaranties. As a surprising result, we showed that systems of constraints over the intervals can be solved in cubic time with a simple algorithm based on acceleration techniques in the abstract domain of intervals[Leroux 2007a].

A cubic time algorithm for solving systems of constraints over the intervals with acceleration techniques in the abstract domain of intervals.

Interpolated SMT-solvers

Ken McMillan shown in [McMillan 2003, McMillan 2006] a new framework for solving the verification problems thanks to Craig interpolants. A Craig interpolant for two formulas ϕ_A and ϕ_B such that $\phi_A \wedge \phi_B$ is unsatisfiable is a formula ψ with a vocabulary included in the intersection of the vocabularies of ϕ_A and ϕ_B. For instance, with the Presburger arithmetic, if $\phi_A = (x = y \wedge y = z)$ and $\phi_B = (x < z)$, the formula $\psi = (x = z)$ is a Craig interpolant for $\phi_A \wedge \phi_B$. In order to apply the interpolation-based model checking to Presburger counter machines, formulas ϕ_A and ϕ_B are quantifier-free Presburger formulas and we are looking for interpolants in the same fragment. Since Craig interpolants can be obtained from proof of unsatifiability $\phi_A \wedge \phi_B$ obtained by SAT-solver or SMT-solver for the quantifier-free fragment of first order logic over the rational numbers[Pudlák 1995, McMillan 2003, McMillan 2005], we explored this direction for the quantifier-free fragment of the Presburger arithmetic.

A SAT-solver is a decision procedure for solving SAT problems, i.e. the satisfiability of propositional formulas. This field is very active in the research community and many efficient implementations are developed in different laboratories. The SAT problem is known to be algorithmically difficult since it is complete for the complexity class NP. This completeness is important. In fact, since any problem in the NP class can be reduced in log-space to an instance of the SAT problem, an efficient algorithm for SAT can be used to solve efficiently other NP problems.

A SMT-solver (for SAT modulo theory) is an algorithm built around a SAT-solver kernel. Intuitively it is possible to decide the validity of the formula $(x = y) \Rightarrow (x = y \vee x < y)$ by observing the validity of the propositional formula $p_1 \Rightarrow (p_1 \vee p_2)$. An SMT-solver follows this idea. In order to obtain efficient algorithms for deciding fragments of first order logics, the propositional reasoning is delegated to a SAT-solver and the reminder to a dedicated solver for the considered logic. In practice for the quantifier free fragment of the Presburger arithmetic (QFPA), the dedicated solver should be able to decide the satisfiability of a conjunction of linear equalities and disequalities over the natural numbers (the inequalities and modular constraints can be reduced to equalities by adding some fresh variables).

With Daniel Kroening and Philipp Rümmer (Oxford university, UK) we showed in [Kroening 2010] how to compute in polynomial time Craig interpolants for the quantifier free fragment of the Presburger arithmetic form proofs of unsatisfiability computed by SMT-solvers. We implemented our algorithm in the tool OpenSMT and we get encouraging results. We are still working on the subject with the long term perspective of implementing an efficient interpolation-based model-checker for Presburger counter machines.

> An algorithm computing Craig interpolants for the quantifier-free fragment of the Presburger arithmetic in polynomial time from proofs of unsatisfiability computed by SMT-solvers. This algorithm is implemented in the SMT-solver OpenSMT.

1.4 Vector Addition Systems

Vector Addition Systems (VAS), Vector Addition Systems with States (VASS), or equivalently Petri Nets are one of the most popular formal models for the representation and the analysis of parallel processes [Esparza 1994]. Their reachability problem is central since many computational problems (even outside the realm of parallel processes) reduce to the reachability problem[Bojańczyk 2006, Demri 2009, Figueira 2009]. This problem is difficult and its decidability remained open during 20 years. Sacerdote and Tenney provided in [Sacerdote 1977] a partial proof of decidability. The proof was completed in 1981 by Mayr [Mayr 1981b] and simplified by Kosaraju [Kosaraju 1982] from [Sacerdote 1977, Mayr 1981b]. Ten years later [Lambert 1992], Lambert provided a further simplified version based on [Kosaraju 1982]. Nowadays, no decision procedure are implemented. In fact the known algorithm is conceptually difficult (to be understood and to be implemented) and with a high computational complexity bound (Ackermann).

Inductive invariants

Since the proof of decidability of the reachability problem in the early 80's, very few results improved the original solution except the book of Reutenauer [Reutenauer 1990] and the Lambert's structures [Lambert 1992]. Some extensions of

the problem were shown to be decidable by Jançar in [Jančar 1990a] and Reinhardt in [Reinhardt 2005] by considering richer properties or more expressive models. Recently we obtained a significant progress by introducing in [Leroux 2009a] a simple algorithm for deciding the reachability problem based on inductive invariants definable in the Presburger arithmetic that provide certificates of non reachability. Whereas these invariants are obtained for the Lambert's structure in the proof published in 2009, we provided in [Leroux 2011a] a new shorter direct proof of the existence of these inductive invariants. This new proof introduces the central notion of almost semilinear sets, a class of sets inspired by the characterization of Presburger sets by semilinear sets [Ginsburg 1966]. The class of almost semilinear sets provides a simple way to capture the reachability sets of vector addition systems, and in particular those with reachability sets not definable in the Presburger arithmetic.

Unreachable configurations can be separated by inductive invariants definable in the Presburger arithmetic.

Reversible reachability problem

The complexity of the reachability problem is still open. The gap of complexity between best lower and upper bounds is very large. Lipton proved in [Cardoza 1976] that the reachability problem is hard for the class of problems solvable in exponential space. Concerning the upper bound of complexity, we can only extract a non-primitive recursive bound for the computational complexity of the original algorithm solving the reachability problem. We succeeded in closing the complexity gap for the reversible reachability problem. The reversible reachability problem consists to decide for two configurations c, c' if c' is reachable from c and c is reachable from c'. In [Leroux 2011b] we proved that the reversible reachability problem is EXPSPACE-complete. This result is obtained by combining Rackoff techniques [Rackoff 1978] and Kosaraju ideas [Kosaraju 1982] in order to prove that if two configurations are reachable and co-reachable, then there exist "short" paths proving this property ("short" means double exponential in the size of the input problem).

The reversible reachability problem is EXPSPACE-complete.

Part I

Presburger Arithmetic

Chapter 2

First Order Logic

In this thesis, first order logics are used as (decidable) formalism to denote properties over configurations of infinite state systems. We recall in this chapter briefly some elements of the first order logics. In the sequel X denotes an infinite countable set of elements called *variables*.

A *signature* is a tuple $(\mathbb{D}, \mathcal{S}, F)$ where $\mathbb{D}$ is a set called the *domain*, $\mathcal{S}$ is a class of *sets* $\mathbf{S} \subseteq \mathbb{D}^n$ for some arity $n \in \mathbb{N}$ depending on $\mathbf{S}$ and F is a class of totally defined *functions* $f : \mathbb{D}^n \to \mathbb{D}$ for some arity $n \in \mathbb{N}$ depending on f. A function with a zero arity is also called a *constant*. A *term* is either a variable $x \in X$ or an element of the form $f(t_1, \ldots, t_n)$ where $f : \mathbb{D}^n \to \mathbb{D}$ is a function in F and $t_1, \ldots, t_n$ are some terms. A *predicate* is an element of the form $t_1 = t_2$ where t_1, t_2 are two terms or an element of the form $\mathbf{S}(t_1, \ldots, t_n)$ where $\mathbf{S} \subseteq \mathbb{D}^n$ is a set in $\mathcal{S}$ and $t_1, \ldots, t_n$ are some terms. A *first order formula* ψ is either a predicate p, an element of the form $\neg\phi$, $\exists x\ \phi$, $\forall x\ \phi$ where ϕ is a first order formula and $x \in X$ a variable, or an element $\phi_1 \vee \phi_2$ or $\phi_1 \wedge \phi_2$ where ϕ_1 and ϕ_2 are two first order formulas. We denote by $\mathrm{FO}(\mathbb{D}, \mathcal{S}, F)$ the set of first order formulas.

Remark 2.0.1. Syntax signatures *and* interpretations of the symbols *are usually presented separately. Since we are only interested by interpreted first order logics*[1] *we only introduce the notion of* signature *that sums up these two notions.*

A *quantifier-free formula* is either a predicate p, an element of the form $\neg\phi$ where ϕ is a quantifier-free formula, or an element of the form $\phi_1 \vee \phi_2$ or $\phi_1 \wedge \phi_2$ where ϕ_1, ϕ_2 are two quantifier-free formulas. A *quantifier-free conjunctive formula* is either a predicate p or its negation $\neg p$, or a conjunction $\phi_1 \wedge \phi_2$ where ϕ_1 and ϕ_2 are two quantifier-free conjunctive formulas.

The *semantics* is defined as follows. A *valuation function* is a function $v : X \to \mathbb{D}$. Given a variable $x \in X$ and an element $d \in \mathbb{D}$, we denote by $v[x \to d]$ the valuation $w : X \to \mathbb{D}$ defined by $w(x) = d$ and $w(y) = v(y)$ for every variable $y \in X \backslash \{x\}$. Valuation functions are extended over the terms inductively by $v(f(t_1, \ldots, t_n)) = f(v(t_1), \ldots, v(t_n))$. As usual, we introduce the unique function that maps every first order formula ψ onto a set $\mathrm{model}(\psi)$ of valuations satisfying the following constraints:

[1] Uninterpreted symbols only occur in Section 4.1 with the "self-definability" of the Presburger arithmetic.

- $\text{model}(t_1 = t_2) = \{v : X \to \mathbb{D} \mid v(t_1) = v(t_2)\}$.
- $\text{model}(\mathbf{S}(t_1, \ldots, t_n)) = \{v : X \to \mathbb{D} \mid (v(t_1), \ldots, v(t_n)) \in \mathbf{S}\}$.
- $\text{model}(\neg\phi) = \{v : X \to \mathbb{D} \mid v \notin \text{model}(\phi)\}$.
- $\text{model}(\exists x\ \phi) = \bigcup_{d\in\mathbb{D}}\{v : X \to \mathbb{D} \mid v[x \to d] \in \text{model}(\phi)\}$.
- $\text{model}(\forall x\ \phi) = \bigcap_{d\in\mathbb{D}}\{v : X \to \mathbb{D} \mid v[x \to d] \in \text{model}(\phi)\}$.
- $\text{model}(\phi_1 \vee \phi_2) = \text{model}(\phi_1) \cup \text{model}(\phi_2)$.
- $\text{model}(\phi_1 \wedge \phi_2) = \text{model}(\phi_1) \cap \text{model}(\phi_2)$.

When $\text{model}(\psi)$ is non empty we say that ψ is *satisfiable*. Two first order formulas ψ_1, ψ_2 are said to be *equivalent* if $\text{model}(\psi_1) = \text{model}(\psi_2)$

The set of *variables* of a term is defined inductively by $\text{var}(x) = \{x\}$ for every variable $x \in X$ and $\text{var}(f(t_1, \ldots, t_n)) = \text{var}(t_1) \cup \ldots \cup \text{var}(t_n)$. The set of variables of a first order formula ψ is defined inductively by $\text{var}(t_1 = t_2) = \text{var}(t_1) \cup \text{var}(t_2)$, $\text{var}(\mathbf{S}(t_1, \ldots, t_n)) = \text{var}(t_1) \cup \ldots \cup \text{var}(t_n)$, $\text{var}(\neg\phi) = \text{var}(\phi)$, $\text{var}(\exists x\ \phi) = \text{var}(\phi)\backslash\{x\}$, $\text{var}(\forall x\ \phi) = \text{var}(\phi)\backslash\{x\}$, $\text{var}(\phi_1 \vee \phi_2) = \text{var}(\phi_1) \cup \text{var}(\phi_2)$, $\text{var}(\phi_1 \wedge \phi_2) = \text{var}(\phi_1) \cup \text{var}(\phi_2)$. Given a vector $\mathbf{x} \in X^n$ we denote by $\text{var}(\mathbf{x})$ the set of variables $\{\mathbf{x}(1), \ldots, \mathbf{x}(n)\}$.

A set $\mathbf{S} \subseteq \mathbb{D}^n$ is said to be *definable* if there exists a first order formula ψ and a vector $\mathbf{x} \in X^n$ of distinct variables such that $\text{var}(\psi) \subseteq \text{var}(\mathbf{x})$ and such that $\mathbf{S} = \{v(\mathbf{x}) \mid v \in \text{model}(\psi)\}$ where $v(\mathbf{x}) = (v(\mathbf{x}(1)), \ldots, v(\mathbf{x}(n)))$. In that case $\mathbf{S}$ is said to be *denoted* by $\psi(\mathbf{x})$. A totally defined function $f : \mathbb{D}^n \to \mathbb{D}$ is said to be *definable* if the set $\{(\mathbf{d}, f(\mathbf{d})) \mid \mathbf{d} \in \mathbb{D}^n\}$ is *definable* with the classical isomorphism that identify $\mathbb{D}^n \times \mathbb{D}$ with $\mathbb{D}^{n+1}$.

Remark 2.0.2. *Observe that $t_1 = t_2$ is a predicate even if the signature $(\mathbb{D}, \mathcal{S}, F)$ does not contain the equality. This choice simplifies the previous definitions of definability. In fact if we consider first order logics without the equality predicate, the previous definitions of definability must require additionally that the occuring variables are* distinct.

We say that $\text{FO}(\mathbb{D}, \mathcal{S}_1, F_1)$ and $\text{FO}(\mathbb{D}, \mathcal{S}_2, F_2)$ have the *same expressive power* if the class of sets definable in one logic coincides with the class of sets definable in the other one. The proof of the following lemma is immediate.

Lemma 2.0.3. $\text{FO}(\mathbb{D}, \mathcal{S}_1, F_1)$ *and* $\text{FO}(\mathbb{D}, \mathcal{S}_2, F_2)$ *have the same expressive power if and only if every set $S_2 \in \mathcal{S}_2$ and every function $f_2 \in F_2$ are definable in* $\text{FO}(\mathbb{D}, \mathcal{S}_1, F_1)$*, and symmetrically every set $S_1 \in \mathcal{S}_1$ and every function $f_1 \in F_1$ are definable in* $\text{FO}(\mathbb{D}, \mathcal{S}_2, F_2)$.

CHAPTER 3

Deciding The Presburger Arithmetic

In the program analysis and verification field, one often faces the problem of finding a suitable formalism for expressing the constraints to be satisfied by the configurations. Ideally, this formalism has to be decidable, while still remaining expressive enough for handling the class of constraints needed by the application. An example of such a formalism is the first order logic over the natural numbers with the addition which is widely used for reasoning about programs manipulating counters. This logic, also known as the *Presburger arithmetic* is indeed decidable [Presburger 1929, Büchi 1962], yet expressive enough for describing arbitrary linear constraints as well as discrete periodicities. In this chapter we recall the two classical ways for deciding the Presburger arithmetic either based on quantifier elimination algorithms or on automata constructions. These methods are briefly introduced in the following two Sections 3.1 and 3.2. Finally in Section 3.3 we provide some benchmarks.

We denote by $\mathbb{N}, \mathbb{N}_{>0}, \mathbb{Z}, \mathbb{Q}, \mathbb{Q}_{\geq 0}, \mathbb{Q}_{>0}, \mathbb{R}$ the set of *natural numbers*, *positive integers*, *integers*, *rational numbers*, *non negative rational numbers*, and *positive rational numbers*, and *real numbers*. *Vectors* and *sets of vectors* are denoted in bold face. The ith *component* of a vector $\mathbf{v} \in E^d$ is denoted by $\mathbf{v}(i)$. The *dot product* of two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{Q}^d$ is the rational number $\mathbf{x} \cdot \mathbf{y} = \sum_{i=1}^{d} \mathbf{x}(i)\mathbf{y}(i)$. We introduce $||\mathbf{v}||_\infty = \max_{1 \leq i \leq d} |\mathbf{v}(i)|$ where $|\mathbf{v}(i)|$ is the *absolute value* of $\mathbf{v}(i)$. The total order $\leq$ over $\mathbb{Q}$ is extended component-wise into an order $\leq$ over the set of vectors $\mathbb{Q}^d$. The addition function $+$ is also extended component-wise over $\mathbb{Q}^d$. Given two sets $\mathbf{V}_1, \mathbf{V}_2 \subseteq \mathbb{Q}^d$ we denote by $\mathbf{V}_1 + \mathbf{V}_2$ the set $\{\mathbf{v}_1 + \mathbf{v}_2 \mid (\mathbf{v}_1, \mathbf{v}_2) \in \mathbf{V}_1 \times \mathbf{V}_2\}$, and we denote by $\mathbf{V}_1 - \mathbf{V}_2$ the set $\{\mathbf{v}_1 - \mathbf{v}_2 \mid (\mathbf{v}_1, \mathbf{v}_2) \in \mathbf{V}_1 \times \mathbf{V}_2\}$. Following the same way, given $T \subseteq \mathbb{Q}$ and $\mathbf{V} \subseteq \mathbb{Q}^d$ we let $T\mathbf{V} = \{t\mathbf{v} \mid (t, \mathbf{v}) \in T \times \mathbf{V}\}$. We also denote by $\mathbf{v}_1 + \mathbf{V}_2$ and $\mathbf{V}_1 + \mathbf{v}_2$ the sets $\{\mathbf{v}_1\} + \mathbf{V}_2$ and $\mathbf{V}_1 + \{\mathbf{v}_2\}$, and we denote by $t\mathbf{V}$ and $T\mathbf{v}$ the sets $\{t\}\mathbf{V}$ and $T\{\mathbf{v}\}$. In the sequel, an empty sum of sets included in $\mathbb{Q}^d$ denotes the set reduced to the zero vector $\{\mathbf{0}\}$.

3.1 Quantifier Elimination

The original way for deciding the satisfiability of Presburger formulas is based on a quantifier elimination algorithm for a richer signature having the same expressive power [Presburger 1929]. This signature is obtained by introducing the binary rela-

tions $\equiv_m$ over $\mathbb{N}$ indexed by a natural number $m \in \mathbb{N}$ and defined by $n_1 \equiv_m n_2$ if $n_1 + m\mathbb{Z} = n_2 + m\mathbb{Z}$.

Definition 3.1.1. *The* Presburger arithmetic *is the first order logic* $\mathrm{FO}(\mathbb{N}, +)$.

Definition 3.1.2. *The* extended Presburger arithmetic *is the first order logic* $\mathrm{FO}(\mathbb{N}, \leq, (\equiv_m)_{m\in\mathbb{N}}, +, 0, 1)$.

In order to provide complexity results, we introduce the size function defined by $\mathrm{size}(=) = 1$, $\mathrm{size}(\leq) = 1$, $\mathrm{size}(+) = 1$, $\mathrm{size}(0) = \mathrm{size}(1) = 1$, and $\mathrm{size}(\equiv_m) = k$ where k is the minimal natural number in $\mathbb{N}_{>0}$ such that $m < 2^k$. The size function is extended over the terms inductively by $\mathrm{size}(x) = 1$ for every variable $x \in X$ and $\mathrm{size}(+(t_1, \ldots, t_n)) = \mathrm{size}(+) + \mathrm{size}(t_1) + \cdots + \mathrm{size}(t_n)$. The size of a first order formula ψ is defined inductively by $\mathrm{size}(\mathbf{S}(t_1, \ldots, t_n)) = \mathrm{size}(\mathbf{S}) + \mathrm{size}(t_1) + \cdots + \mathrm{size}(t_n)$, $\mathrm{size}(\neg\phi) = 1 + \mathrm{size}(\phi)$, $\mathrm{size}(\exists x\ \phi) = 1 + \mathrm{size}(\phi)$, $\mathrm{size}(\forall x\ \phi) = 1 + \mathrm{size}(\phi)$, $\mathrm{size}(\phi_1 \vee \phi_2) = \mathrm{size}(\phi_1) + 1 + \mathrm{size}(\phi_2)$, $\mathrm{size}(\phi_1 \wedge \phi_2) = \mathrm{size}(\phi_1) + 1 + \mathrm{size}(\phi_2)$.

Lemma 3.1.3. *The extended Presburger arithmetic and the Presburger arithmetic have the same expressive power. Moreover extended Presburger formulas are equivalent to Presburger formulas computable in linear time.*

Proof. Let us introducing the function $f_m : \mathbb{N} \to \mathbb{N}$ that multiply by a natural number $m \in \mathbb{N}$ and formally defined by $f_m(n) = mn$ for every $n \in \mathbb{N}$. We introduce the functions f_m since they simplify the proof that $\equiv_m$ is definable in the Presburger arithmetic. A term $f_m(t)$ is simply denoted by mt in the sequel. The proof of this lemma is obtained by observing that the constant 0 is denoted by $y + y = y$. The inequality $x_1 \leq x_2$ is denoted by $\exists x\ x_1 + x = x_2$. The constant 1 is denoted by $\forall x(\ x = 0 \vee y \leq x)$ that expresses that 1 is the smallest natural number not equal to 0. The function $y = f_0(x)$ is defined by $y = 0$. We show that the function $y = f_m(x)$ with $m \in \mathbb{N}_{>0}$ is denoted by a Presburger formula with a size bounded linearly in $\mathrm{size}(\equiv_m)$ by decomposing m in binary $m = 2^0 b_1 + \cdots + 2^{k-1} b_k$ where $b_1, \ldots, b_k \in \{0, 1\}$ and $b_k = 1$. Observe that $\mathrm{size}(\equiv_m) = k$. We introduce the sets $I_m^b = \{i \in \{1, \ldots, k\} \mid b_i = b\}$ where $b \in \{0, 1\}$. The function $y = f_m(x)$ is denoted by the following formula:

$$\exists x_0 \ldots \exists x_{k+1}(y = x_0 \wedge x_{k+1} = 0 \wedge \bigwedge_{i\in I_m^0} x_i = x_{i+1} + x_{i+1} \wedge \bigwedge_{i\in I_m^1} x_i = (x_{i+1} + x_{i+1}) + x)$$

Finally just observe that $x_1 \equiv_m x_2$ is denoted by $\exists y_1\ \exists y_2\ x_1 + f_m(y_1) = x_2 + f_m(y_2)$. □

Theorem 3.1.4 ([Oppen 1978])**.** *Extended Presburger formulas are equivalent to quantifier-free extended Presburger formulas computable in 3-EXPTIME. The computational complexity bound is tight.*

We deduce from the previous theorem that the satisfiability of a Presburger formula is decidable in 3-EXPTIME. This complexity result can be slightly improved as

follows. First of all, concerning the satisfiability of a quantifier-free extended Presburger formula. The satisfiability of a formula in this fragment is known to be an NP-complete problem. The NP hardness is obtained by encoding 3-SAT instances. We show that the problem is in NP by proving that if there exists a valuation v satisfying such a formula ψ, then there exists another one such that size$(v(x))$ is polynomially bounded by size(ψ) (by observing that v is a solution of a *linear system*). Such a bound on $v(x)$ can be extended to the full extended Presburger arithmetic (with quantifiers). In fact, in [Berman 1977] Leonard Berman proved that there exists a constant $c \in \mathbb{N}$ such that every Presburger formula $Q_1 x_1 \ \ldots \ Q_n x_n \ \phi$ where ϕ is an extended quantifier-free Presburger formula and $Q_j \in \{\exists, \forall\}$ is equisatisfiable to the formula $Q_1 x_1 \in F \ \ldots \ Q_n x_n \in F \ \phi$ where $F = \{0, \ldots, 2^{2^{ck}}\}$ and k is the size of the formula ϕ. In particular the satisfiability of a Presburger formula belongs to the class of problems that can be solved with alternating Turing machines working in 2-EXPTIME with a linear number of alternations. In [Berman 1977] the satisfiability of a Presburger formula is proved to be complete for this class.

Theorem 3.1.5 ([Berman 1977])**.** *The satisfiability of a Presburger formula is complete for the class of problems that can be solved with alternating Turing machines working in 2-EXPTIME with a linear number of alternations.*

As a direct consequence, the satisfiability of an extended Presburger formula (as well as a Presburger formula) is 2-EXPSPACE-hard and 3-EXPTIME-easy. So from a theoretical point view the satisfiability of a Presburger formula is a difficult problem. From a practical point of view, efficient decision procedures based on quantifier eliminations algorithm exist. For instance, the elimination algorithm introduced in [Pugh 1992a] (the omega test) and implemented in the tool Omega works well in practice. Section 3.3 provides experimental results.

3.2 Arithmetic Automata

A simple approach for deciding the Presburger arithmetic consists in using finite state automata. It is indeed known that, using the positional notation for encoding numbers and vectors into words, all Presburger sets are mapped onto regular languages and can thus be recognized by automata [Büchi 1962, Bruyère 1994]. A Presburger formula can be decided by recursively constructing an automaton encoding its solutions, and then checking whether this automaton accepts a nonempty language. In some program verification applications, such automata, called *Number Decision Diagrams (NDDs)* or *arithmetic automata* are actually used as data structures for representing and manipulating symbolically infinite sets of configurations that need to be handled [Boigelot 1998].

Let us introduce some formal notations. A *basis of decomposition* is a natural number $r \in \mathbb{N}_{\geq 2}$. The *alphabet* Σ_r in basis r is defined by $\Sigma_r = \{0, \ldots, r-1\}$ and an element $b \in \Sigma_r$ is called a *digit*. Given a dimension $d \in \mathbb{N}$, the *alphabet* $\Sigma_{r,d}$ is defined by $\Sigma_{r,d} = \Sigma_r^d$. An element $\mathbf{b} \in \Sigma_{r,d}$ is called a *digit vector*. As expected vectors

$\mathbf{v} \in \mathbb{N}^d$ are encoded by words of digit vectors thanks to the function $\rho_r : \Sigma_{r,d}^* \to \mathbb{N}^d$ defined by:

$$\rho_r(\mathbf{b}_1 \dots \mathbf{b}_k) = \sum_{j=1}^{k} r^{j-1}\mathbf{b}_j$$

Definition 3.2.1. *An* arithmetic automaton in basis r *is a finite state automaton that recognizes a regular language $L \subseteq \Sigma_{r,d}^*$. The set $\rho_r(L)$ is called the set* encoded *by the arithmetic automaton.*

Lemma 3.2.2. *A set $\mathbf{S} \subseteq \mathbb{N}^d$ can be encoded by an arithmetic automaton in basis r if and only if the language $\rho_r^{-1}(\mathbf{S})$ is regular.*

Proof. It is sufficient to prove that $\rho_r^{-1}(\rho_r(L))$ is regular for every regular language $L \subseteq \Sigma_{r,d}^*$. Just observe that two words $\sigma_1, \sigma_2 \in \Sigma_{r,d}^*$ satisfies $\rho_r(\sigma_1) = \rho_r(\sigma_2)$ if and only if the intersection $\sigma_1\mathbf{0}^* \cap \sigma_2\mathbf{0}^*$ is non empty. We deduce the following equality:

$$\rho_r^{-1}(\rho_r(L)) = \bigcup_{k\in\mathbb{N}} L.(\mathbf{0}^k)^{-1}\mathbf{0}^*$$

Where $L.(\mathbf{0}^k)^{-1}$ denotes the *residue language* $\{\sigma \in \Sigma_{r,d}^* \mid \sigma\mathbf{0}^k \in L\}$. Since L is regular we deduce that $L.(\mathbf{0}^k)^{-1}$ is regular and moreover the class $\{L.(\mathbf{0}^k)^{-1} \mid k \in \mathbb{N}\}$ is finite. Therefore $\rho_r^{-1}(\rho_r(L))$ is regular. □

All Presburger sets can be encoded by arithmetic automata in any basis [Büchi 1962, Bruyère 1994]. In fact, thanks to classical automata constructions we can build up inductively automata recognizing the solutions of sub-formulas. There are many variants for implementing such an algorithm. A simple way for computing an arithmetic automaton encoding the set denoted by $\psi(x_1, \dots, x_n)i$ where ψ is a Presburger formula consists to first transform in linear time ψ into an equivalent Presburger formula such that predicates are reduced to $x + y = z$ or $x = y$ for some variables x, y, z by introducing fresh variables. Then the algorithm computes inductively arithmetic automata in basis r based on procedures building automata encoding the following sets from automata encoding $\mathbf{S}, \mathbf{S}_1, \mathbf{S}_2 \subseteq \mathbb{N}^d$:

- Predicate Construction : $\{(s_1, s_2, s_3) \in \mathbb{N}^3 \mid s_1 + s_2 = s_3\}$.
- Equality : $\{(s_1, s_2) \in \mathbb{N}^2 \mid s_1 = s_2\}$.
- Complement : $\mathbb{N}^n \backslash \mathbf{S}$.
- Union: $\mathbf{S}_1 \cup \mathbf{S}_2$.
- Variable Insertion : $\{(s_1, \dots, s_i, t, s_{i+1}, \dots, s_n) \mid (s_1, \dots, s_n) \in \mathbf{S} \ \wedge \ t \in \mathbb{N}\}$.
- Quantifier Elimination : $\{(s_1, \dots, s_{i-1}, s_{i+1}, \dots, s_n) \mid (s_1, \dots, s_n) \in \mathbf{S}\}$.

The previous procedures can be implemented in polynomial time with deterministic finite state automata except the quantifier elimination procedure that requires exponential time in the worst case. In fact this procedure determinized a non deterministic automaton obtained from the input automaton by removing the ith component of every digit vectors labeling the transitions. In theory the exponential blow up cannot be avoided but in practice the quantifier elimination procedure works well. Moreover usually it produces deterministic automata with a number of states smaller than the one of the input automaton. Note that if we consider non deterministic automata or alternating automata we still have some complexity issues. Moreover the implementation of the other procedures are much more complex with these extensions. Note also that for large classes of Presburger sets $\mathbf{S} \subseteq \mathbb{N}^d$, it is proved in [Durand-Gasselin 2010] that non deterministic arithmetic automata encoding $\mathbf{S}$ have a number of states as large as the number of states of the minimal deterministic automaton that recognizes the language $\rho_r^{-1}(\mathbf{S})$. That result shows that arithmetic automata should be put in minimal deterministic form.

Although every Presburger set can be denoted by an arithmetic automaton, the reciprocal property does not hold. For instance, the set $\{r^k \mid k \in \mathbb{N}\}$, which is not a Presburger set, clearly corresponds to a regular language and is thus recognizable. The well-known Cobham's theorem states that, if a set $S \subseteq \mathbb{N}$ is simultaneously denotable by arithmetic automata in two bases $r, s \in \mathbb{N}_{>1}$ that are multiplicatively independent, i.e., such that $r^p \neq s^q$ for all $p, q \in \mathbb{N}_{>0}$, then S is Presburger definable [Cobham 1969]. This result has then been extended to subsets of $\mathbb{N}^d$, with $d > 0$ by Semenov [Semenov 1977]. Thanks to simple automata constructions we easily proved that sets recognizable in basis r are also recognizable in any basis multiplicatively dependent with r. Hence the Semenov theorem theorem provides a complete characterization of the sets definable in multiple basis. As a corollary of the Semenov's theorem, the subsets of $\mathbb{N}^d$ that can be denoted by arithmetic automata in every base $r \in \mathbb{N}_{>1}$ are exactly those that are definable in the Presburger arithmetic.

Remark 3.2.3. *We recently extended the Cobham's and Semenov's theorem to Buchi automata recognizing subsets of $\mathbb{R}^d$ [Boigelot 2009].*

The class of sets $\mathbf{S} \subseteq \mathbb{N}^d$ that can be denoted by arithmetic automata in basis r are characterized by a first order logic as follows. We introduce the *valuation function* $V_r : \mathbb{N} \to \mathbb{N}$ defined by $V_r(n) = 0$ if $n = 0$ and $V_r(n)$ is the greatest power of r that divides n otherwise. Let us observe that $\rho_r^{-1}(\{\mathbf{s} \in \mathbb{N}^2 \mid V_r(\mathbf{s}(1)) = \mathbf{s}(2)\})$ is regular since it is denoted by the following regular language:

$$\bigcup_{b \in \{1,\ldots,r-1\}} (0,0)^*(b,1)(\Sigma_r \times \{0\})^*$$

With the same procedures than the one used to build arithmetic automata from Presburger formulas we show that every set definable in the logic FO $(\mathbb{N}, +, V_r)$ can

be denoted by an arithmetic automaton in basis r. Conversely, given an arithmetic automaton encoding a set $\mathbf{S} \subseteq \mathbb{N}^d$, we show that $\mathbf{S}$ is definable in the logic $\mathrm{FO}(\mathbb{N}, +, V_r)$ thanks to a formula that denoting all the possible executions of an arithmetic automaton.

We introduce the function power : $\mathbb{N} \to \mathbb{N}$ defined by $\mathrm{power}(n) = 2^n$ for every $n \in \mathbb{N}$. The following theorem shows that $\mathrm{FO}(\mathbb{N}, +, V_r)$ provides a characterization of the sets that can be denoted by arithmetic automata.

Theorem 3.2.4 ([Büchi 1962, Bruyère 1994, Blumensath 2000])**.** *A set $\mathbf{S} \subseteq \mathbb{N}^d$ can be encoded by an arithmetic automaton in basis r if and only if it is definable in $\mathrm{FO}(\mathbb{N}, +, V_r)$. From a complexity point of view, from an arithmetic automaton encoding $\mathbf{S}$ we can compute in polynomial time a presentation $(x_1, \ldots, x_d) \mid \psi$ of $\mathbf{S}$ in the first order logic $\mathrm{FO}(\mathbb{N}, +, V_r)$, and conversely from a presentation $(x_1, \ldots, x_d) \mid \psi$ of $\mathbf{S}$ we can compute an arithmetic automaton encoding $\mathbf{S}$ in time $\mathrm{power}^n(n)$ where $n = d + \mathrm{length}(\psi)$. The computational complexity bounds are tight.*

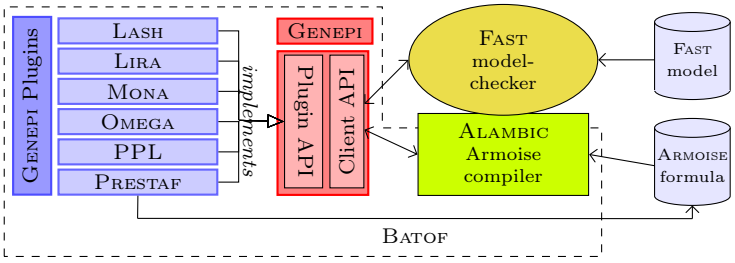
Whereas arithmetic automata recognizing sets of vectors satisfying formulas ψ of $\mathrm{FO}(\mathbb{N}, +, V_r)$ can be non-elementary [Blumensath 2000] (more precisely a tower of exponential with a height equals to the number of quantifier alternations in ψ), Felix Klaedtke showed in [Klaedtke 2004] that the minimal deterministic arithmetic automata that denotes the set of solutions of a Presburger formula ψ has a size at most 3-exponential in the size of ψ. In particular an algorithm that computes bottom up minimal deterministic automata for Presburger formulas computes intermediate automata with sizes bounded by 3-exponential in the size of ψ.

Theorem 3.2.5 ([Klaedtke 2004, Durand-Gasselin 2010])**.** *The algorithm that computes inductively minimal deterministic arithmetic automata encoding the set of solutions of a Presburger formula terminates in 3-EXPTIME. The computational complexity bound is tight.*

The previous Theorem 3.2.5 and Theorem 3.1.4 show that deciding the Presburger arithmetic with quantifier-free Presburger formulas or arithmetic automata have the same computational complexity upper bound.

3.3 TaPAS

With Gérald Point (LaBRI, Bordeaux), we developed TaPAS (The Talence Presburger Arithmetic Suite), a suite of libraries dedicated to the Presburger arithmetic (and some extensions). The suite provides the application programming interface Genepi for this logic with encapsulations of many classical solvers, the BDD-like library Prestaf used for encoding Presburger formulas by automata, and the tool Batof decoding automata to Presburger formulas. All these tools are distributed in the tool suite TaPAS at `http://altarica.labri.fr/wiki/tools:tapas:` under a GPLv2 licence. The following figure shows the architecture of TaPAS (enclosed by the dotted line). The Fast model-checker is also depicted but it does not actually belong to the suite; it is just a client application.



Tools Genepi, Alambic, and Prestaf are presented in Section 3.3.1, 3.3.2 and 3.3.3.

3.3.1 Genepi

Selecting the most efficient solver for a given application can be difficult. This choice can change dramatically the practical performance of an application. Since solvers have incompatible application programming interfaces (API), a particular one must be first selected prior the implementation of the application; changes afterwards require additional development effort. The Genepi library[Bardin 2006] addresses this issue by offering small APIs between, on one side, applications requiring solvers and, on the other side, solvers implementing decision procedures. The connection between applications and solvers is realized transparenttly using dynamically loaded modules (*plugins*) specified at the execution time. This way the choice of a solver can be postponed after the implementation of an application based on Genepi; then the best solver can be selected by performing benchmarks.

Potential applications. People concerned with Presburger packages can take advantage of our open architecture and API in at least two ways.

(1) Presburger developers. People interested in developing a Presburger package can easily linked it to Faster and use the tool and the 40 case-studies as *intensive benchmarking* for their package.

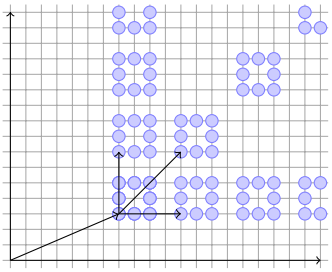
(2) Presburger users. People interested in developing any application requiring Presburger arithmetics can use our generic Presburger API, and then select through the set of implementations which one fits most their application.

Implementations of the API. In the first public release of Genepi presented in [Bardin 2006] that we developed with Gérald Point (LaBRI, Bordeaux) and Sébastien Bardin (CEA Saclay), we provided three implementations of the API based upon the standard packages Lash, Mona and Omega. Since 2006, the tool Lira developed at the ETH Zürich is provided with a Genepi implementation. We also enhanced Genepi with new features: support for $\mathbb{R}$, and the new plugins PPL and Prestaf. Actually the tools Lash and Lira are based on Buchi automata for deciding FO $(\mathbb{R}, \mathbb{Z}, \leq, +)$, tools Mona and Prestaf are based on deterministic finite state automata for deciding the Presburger arithmetic FO $(\mathbb{N}, +)$, Omega

is based on quantifier elimination for deciding FO $(\mathbb{Z}, \leq, +)$ and PPL is based on convex polyhedra for deciding FO $(\mathbb{R}, \leq, +)$.

3.3.2 Alambic

In general, specifying sets with low level logics is difficult and fastidious. The Armoise language allows to describe concisely sets of vectors (in $\mathbb{Z}$ and/or $\mathbb{R}$). The succinctness of formulae is achieved using, among other tricks, hierarchical definitions and arithmetic over sets (which hides numerous and ugly quantifications). Below is an example of an Armoise formula which specifies a repeated pattern in $\mathbb{N}^2$. Note that Armoise formulae may define sets which are not necessarily definable in FO $(\mathbb{R}, \mathbb{Z}, +, \leq)$.



```
let
   origin := (7,3);
   directions := { (4,0), (0,4), (4,4) };
   pattern := ([0...2], [0...2]) \ (1,1);
in
   origin + nat * directions + pattern;
```

TaPAS provides tools to support the Armoise language. The Alambic library permits to compile Armoise formulae into calls to Genepi functions. An important step in this compilation process is the translation of high-level constructions into low level constructions; due to the expressive power of Armoise this transformation is not always possible.

Remark 3.3.1. *In* Armoise *we can define sets as products* $S * T = \{st \mid (s,t) \in S \times T\}$ *where* $S, T \subseteq \mathbb{N}$ *are two Presburger sets. These products are effectively definable in the Presburger arithmetic when* S *or* T *is finite. In general these products are not definable in the Presburger arithmetic. For instance if* S *and* T *are equal to* $\mathbb{N}_{>2}$ *then* $\mathbb{N}_{>1} \backslash (S * T)$ *is the set of prime numbers. Up to our knowledge the decidability of checking if* $S * T$ *is definable in the Presburger arithmetic is open.*

3.3.3 Prestaf

In [Couvreur 2004], Jean Michel Couvreur introduced a new data-structure called *shared-automata*. Intuitively, the sharing of a finite set of automata is performed by merging every states equivalent with respect to the Nérode relation. Prestaf implements this data structure. In Prestaf, deciding if two automata recognize the same language reduces to check the equality of their pointers. Similarly to BDD packages, Prestaf uses a pointer-based hash-table cache algorithm. Up to our knowledge, no other automata package implements this feature. In fact, in other automata libraries testing the equality of two languages is performed in linear time. As a direct consequence, in that tools it is not possible to detect efficiently operations already performed on automata or sub-automata. Since the first version

of PRESTAF written in JAVA by Couvreur [Couvreur 2004], we have developed with Gérald Point a new version written in C and now maintained in TAPAS.

The table given in Fig. 3.1 compares performances of GENEPI plugins on FAST models (these files are distributed within the FAST package). Each process was allocated a budget of 4.5 GB and 5 minutes of CPU times. ERR indicates an abnormal termination of the program due to the consumption of its memory or time limit. The benchmarks were executed on a 64 bits machine equipped with an Intel® Xeon® 2.33 GHz CPU and 8 GB of memory. CPU Execution times are expressed in seconds and were computed with the time command of the bash shell. Columns LASH.l and LASH.m are performed with the same tool LASH but initialized in different way : columns LASH.l and LASH.m correspond to deterministic and co-deterministic arithmetic automata (corresponding to the two possible endianess). The tool PRESTAF out-performed the other tools on 10 cases over 40. Note that for many models that can be analyzed in a very short amount of time, the tool MONA outperformed PRESTAF. This property comes from non optimized initialization and closing methods implemented in PRESTAF.

3.4 Conclusion

In this chapter we recalled the two classical ways for deciding the Presburger arithmetic either based on quantifier elimination algorithms or on automata constructions. In the worst case the two approaches are computationally equivalent since they require 3-exponential time. This upper bound is tight because the satisfiability of a Presburger formula is 2-EXPSPACE-hard.

We implemented the API GENEPI in order to simplify (1) the choice of solvers for programs using the Presburger arithmetic, (2) the benchmarks of existing solvers. This last feature has been used by the tool LIRA that provides an implementation of a GENEPI plugin. With GENEPI we observed that the Presburger formulas and the arithmetic automata provide symbolic representations for solving efficiently incomparable classes of Presburger formulas. As an open problem, we are interested by symbolic representations that can combine formulas and automata in order to get efficient decision procedures for larger classes of Presburger formulas. In Chapter 4 we provide some elements for this problem.

We also implemented the solver PRESTAF for the Presburger arithmetic based on minimal deterministic automata that are represented in strong canonical forms allowing maximal sharing. That means deciding the equality of languages recognized by two automata can be done in constant time since this test reduces to check the equality of the pointers. Up to our knowledge no other automata package implements such a feature. This property is very important for implemented cache algorithms. In fact with other tools it is not possible to retrieve efficiently already performed computations on automata or sub-automata. PRESTAF is very efficient and it has many applications even outside the Presburger arithmetic since it is basically an automata package.

Models	Lash.l	Lash.m	Lira	Mona	Omega	Prestaf
barber	2.17	1.45	0.85	**0.10**	0.38	0.37
berkeley	0.35	0.21	0.13	0.03	**0.02**	0.23
BLW-CAV03	78.35	**15.33**	25.31	31.17	ERR	25.44
BLW-CAV03.s	41.51	11.71	13.89	6.57	**0.91**	5.92
centralserver	55.44	32.86	45.04	**1.49**	ERR	2.53
consistency	ERR	ERR	ERR	**22.61**	22.69	71.25
consistencyInv	6.20	3.55	4.61	**0.21**	0.29	0.45
consprod	90.31	88.65	ERR	3.15	ERR	**2.50**
consprodjava	ERR	ERR	ERR	72.76	ERR	**13.71**
consprodjavaN	ERR	ERR	ERR	71.29	ERR	**13.76**
consprodjavaNInv	ERR	186.79	ERR	6.63	ERR	**2.55**
csm	216.40	153.61	ERR	7.02	ERR	**4.69**
csmInv	6.75	4.60	7.88	**0.25**	0.45	0.69
dekker	119.54	87.06	ERR	**3.13**	ERR	4.21
DRAGON	1.71	1.06	0.75	**0.10**	0.31	0.34
efm	1.36	0.71	0.40	0.06	**0.03**	0.31
FIREFLY	1.06	0.56	0.34	0.05	**0.04**	0.24
fms	ERR	ERR	ERR	15.35	ERR	**12.50**
fmsInv	ERR	ERR	ERR	17.94	ERR	**12.46**
futurbus	4.28	2.86	2.15	**0.18**	0.54	0.48
ILLINOIS	1.01	0.62	0.41	**0.05**	0.08	0.29
incdecInv	ERR	ERR	ERR	ERR	ERR	41.09
kanban	29.08	20.77	121.77	**0.90**	ERR	2.15
lamport	7.04	4.04	3.46	**0.22**	0.88	0.67
lastinfirstserved	3.74	2.42	1.29	**0.16**	2.67	0.44
lift	5.84	3.66	2.81	**0.21**	0.48	0.49
manufacturing	2.19	1.59	1.18	21.89	**0.07**	3.91
MESI	0.40	0.27	0.18	**0.03**	0.04	0.23
MOESI	0.63	0.44	0.28	**0.04**	0.07	0.26
multipoll	112.37	59.97	210.27	**2.46**	24.81	2.47
peterson	16.14	9.16	21.95	**0.42**	3.10	1.18
PROD-CONS	0.45	0.32	0.16	0.03	**0.02**	0.23
readwrit	30.56	19.96	47.18	**1.24**	ERR	6.24
rtp	3.35	2.20	1.29	**0.13**	0.27	0.44
SYNAPSE	0.24	0.14	0.09	0.02	**0.01**	0.21
ticket2i	1.02	0.97	0.70	**0.06**	0.14	0.26
ticket3i	10.65	9.58	7.88	0.42	14.31	**0.40**
train	13.45	7.88	7.43	0.60	15.42	**0.56**
ttp2	ERR	ERR	ERR	104.19	ERR	**32.37**
ttp	ERR	ERR	ERR	**18.14**	29.90	23.56

Figure 3.1: Benchmarks of Genepi plugins performed with Fast.

Chapter 4

Decoding Arithmetic Automata To Presburger Formulas

Presburger formulas lack canonicity since a Presburger set can be denoted by many equivalent Presburger formulas. As a direct consequence, a set that possesses a simple representation could unfortunately be represented in an unduly complicated way. On the other hand, a minimization procedure for arithmetic automata provides a canonical representation. That means, the automaton encoding a Presburger set only depends on this set and not on the way the automaton have been computed. For these reasons, automata seem to be well adapted for applications requiring a lot of boolean manipulations such as the symbolic model-checking.

In practice, Presburger formulas and arithmetic automata are efficient symbolic representations for incomparable classes of Presburger sets (see Section 3.3). In order to obtain efficient symbolic representations for larger classes of Presburger sets, we are interested in defining hybrid representations combining formulas and automata. Whereas there exist efficient algorithms for encoding the set of solutions of Presburger formula into arithmetic automata [Klaedtke 2004, Wolper 2000, Boudet 1996], the inverse problem of decoding arithmetic automata to Presburger formulas is much more complex. Let us observe that with an encoding and a decoding algorithm, we get an algorithm that normalizes Presburger formulas. In fact, from a Presburger formula that denotes a set $\mathbf{S}$ we compute a minimal deterministic arithmetic automaton encoding $\mathbf{S}$ with usual automata constructions. This automaton only depends on $\mathbf{S}$. That means the Presburger formula decoded from the arithmetic automaton only depends on $\mathbf{S}$. Intuitively, the automata minimization algorithm acts like a minimization procedure on the Presburger formulas.

In section 4.1 we recall the 4-exponential time decision procedure introduced by Muchnik for checking if an arithmetic automaton encodes a Presburger set. In Section 4.2 we present our polynomial time algorithm that checks if an arithmetic automaton encodes a Presburger set and decodes the arithmetic automaton into a Presburger formula in the positive case. We provide experiments performed with the tool Batof that implements these algorithms in Section 4.3.

4.1 The Self-Definability

The problem of computing a Presburger formula from an arithmetic automaton is naturally related to the problem of deciding whether an automaton represents a

Presburger set. In fact, there exists automata denoting sets that are not definable in the Presburger arithmetic. For instance the set of powers $\{r^n \mid n \in \mathbb{N}\}$ of a natural number $r \geq 2$ is not definable in the Presburger arithmetic, but it can be denoted by a very simple arithmetic automaton in basis r. This well-known hard problem was first solved by Muchnik in 1991 [Muchnik 1991, Muchnik 2003] with a 4-exponential time algorithm. The Muchnik approach is based on the following property. He proved that for every dimension d, there exists a formula ϕ_d in $\mathrm{FO}(\mathbb{N}, +, P_d)$ where P_d is a d-dimensional uninterpreted predicate such that this formula is valid when P_d is interpreted as a set $\mathbf{S} \subseteq \mathbb{N}^d$ if and only if $\mathbf{S}$ is Presburger definable. In dimension $d = 1$ such a formula is obtained as follows. We observe that a set $N \subseteq \mathbb{N}$ is definable in the Presburger arithmetic if and only if N is asymptotically periodic. This property is equivalent with the fact that there exists $M \in \mathbb{N}$ and $m \in \mathbb{N}_{>0}$ such that for every $n \geq M$ we have $n \in N$ if and only if $n + m \in N$. We deduce that the following formula ϕ_1 satisfies the Muchnik criterion:

$$\phi_1 \;=\; \exists M \exists m \; (\; m \geq 1 \;\wedge\; \forall n \; n \geq M \Rightarrow (P_1(n) \Longleftrightarrow P_1(n+m)) \;)$$

With a non trivial induction on d, Muchnik provided an explicit formula ϕ_d for every $d \geq 1$. In particular, if a set $\mathbf{S} \subseteq \mathbb{N}^d$ is represented by an arithmetic automaton, we can compute an arithmetic automaton representing the set of solutions of ϕ_d where P_d is interpreted as $\mathbf{S}$. This technique provides an algorithm deciding if the set denoted by an arithmetic automaton is definable in the Presburger arithmetic. Due to the quantifier alternations in ϕ_d this algorithm is shown to terminate in 4-exponential time in the size of the arithmetic automaton.

Remark 4.1.1. *The Muchnik criterion shows that the Presburger arithmetic* $\mathrm{FO}(\mathbb{N}, +)$ *is* "self-definable". *In 1991 Muchnik open the problem of finding other logics satisfying this criterion.*

4.2 Polynomial Time Algorithms

In 2003, we started working on efficient solutions for decoding arithmetic automata. In the book [Allouche 2003] published in 2003, Jean-Paul Allouche and Jeffrey Shallit conjectured that the problem in dimension one should be solvable in polynomial time. We provided in [Leroux 2003] a polynomial time algorithm for strongly connected arithmetic automata encoding quantifier-free first order formulas in $\mathrm{FO}(\mathbb{N}, +, 0, 1)$. In particular this algorithm was not able to detect inequality constraints. In 2004 two other algorithms where provided by Louis Latour and Denis Lugiez. In [Latour 2004] the decoding is solved with an exponential time algorithm for arithmetic automata encoding *convex sets*. In [Lugiez 2004] the decoding is solved with a double exponential time algorithm for arithmetic automata encoding sets of the form $\mathbf{B} + \mathbb{N}\mathbf{p}_1 + \cdots + \mathbb{N}\mathbf{p}_k$ where $\mathbf{B} \subseteq \mathbb{N}^d$ is finite and $\mathbf{p}_j \in \mathbb{N}^d$.

In [Leroux 2005a] we solved the general problem with a polynomial time algorithm. More precisely our algorithm decides in polynomial time if a deterministic arithmetic automaton encodes a Presburger set and it decodes it into a Presburger

formula in the positive case. In order to obtain an efficient algorithm in practice, I spent the whole year 2006 to find out efficient algorithms for solving this difficult problem. The results of this research effort are available in an internal report of 134 pages containing a lot of algorithms for solving difficult problems[Leroux 2006]. In 2007, we implemented these algorithms with Gérald Point (LaBRI, Bordeaux) in the tool BATOF distributed with the tool suite TAPAS [Leroux 2009b] (See the previous Chapter 3). In practice our algorithm is very efficient since in less than one minute it can compute a Presburger formula from an arithmetic automaton with more than 100k states. Experimental results are provided in Section 4.3.

High Level View-Point

Let us describe from an high level view-point the algorithm implemented in BATOF. The algorithm is based on the following idea : by changing the set of accepting states of a deterministic arithmetic automaton A that encodes a Presburger set $\mathbf{S}$, we obtain arithmetic automata that encodes "simple" Presburger sets. Moreover, a boolean combination of these simple "Presburger" sets provides $\mathbf{S}$. By "simple" Presburger sets we mean sets of the form $\mathbb{N}^d \cap (\mathbf{B} + (\mathbf{G} \cap \mathbf{H}))$ where $\mathbf{B}$ is a finite subset of $\mathbb{Z}^d$, $\mathbf{G} \subseteq \mathbb{Z}^d$ is a lattice, i.e. a set of the form $\mathbb{Z}\mathbf{p}_1 + \cdots + \mathbb{Z}\mathbf{p}_r$, and $\mathbf{H} = \{\mathbf{z} \in \mathbb{Z}^d \mid \mathbf{h} \cdot \mathbf{z} \geq 0\}$ is an *half-space* parameterized by a vector $\mathbf{h} \in \mathbb{Z}^d$.

Remark 4.2.1. *We proved in [Leroux 2008b] that if A is a minimal deterministic arithmetic automaton encoding a Presburger set $\mathbf{S} \subseteq \mathbb{N}^d$ then the arithmetic automaton obtained from A by changing the initial state and changing the set of accepting states to any set is still an arithmetic automaton encoding a Presburger set.*

The lattices $\mathbf{G}$ on which the "simple" Presburger sets are defined are obtained by applying the Stalling's Folding Process on the underlying graph of A. This process works as follows. It computes inductively a sequence of graphs by merging pairs of states in such a way that we get a deterministic and co-deterministic graph. More precisely, the folding process merges two states q_1, q_2 if either there exist transitions $q \xrightarrow{a} q_1$ and $q \xrightarrow{a} q_2$, or there exist transitions $q_1 \xrightarrow{a} q$ and $q_2 \xrightarrow{a} q$. Our implementation follows the algorithm presented in [Touikan 2006] which performs the computation in time $O(n \operatorname{Ack}^{-1}(n))$ where Ack is the Ackermann function and $n = r|C|$. Note that the Stallings' folding process has applications for canonically representing finitely generated subgroups of the free group [Stallings 1983] by graphs.

We do not provide more information on the way the "simple" Presburger sets are computed since this difficult problem is out of the scope of this thesis.

At some point the algorithm computes from an arithmetic automaton A encoding a set $\mathbf{S}$, a finite class of pairs $(H, \mathbf{S}_H)$ where H is a set of control states and $\mathbf{S}_H$ is a simple Presburger denoted by A^H, the automaton obtained from A by replacing the set of accepting states by H. Since $\mathbf{S}$ is a Presburger set if and only if $\mathbf{S}$ is a boolean combination of the simple Presburger sets $\mathbf{S}_H$, we deduce that $\mathbf{S}$ is a Presburger set if and only if the set of final states F is a boolean combination of

the sets H. This property can be decided in polynomial time. Moreover when such a boolean combination exists we can compute a propositional formula that denotes this combination. The following Example 4.2.2 shows on an example how to perform the computation.

Example 4.2.2. *Let us consider the class $\mathcal{F} = \{\{a,b\},\{b,c,d\},\{e,f\}\}$ of subsets of the finite set $C = \{a,b,c,d,e,f\}$. In order to check if a set S is a boolean combination of sets in $\mathcal{F}$ we first introduce the equivalence relation $\equiv_{\mathcal{F}}$ over Q defined by $q_1 \equiv_{\mathcal{F}} q_2$ if $q_1 \in F \Leftrightarrow q_2 \in F$ for every $F \in \mathcal{F}$. Observe that the equivalence classes are $\{\{a\},\{b\},\{c,d\},\{e,f\}\}$. This class can be computed inductively by considering the equivalence classes $\equiv_{\mathcal{F}_0}$, $\equiv_{\mathcal{F}_1}$, $\equiv_{\mathcal{F}_2}$, $\equiv_{\mathcal{F}_3}$ where $\mathcal{F}_0 = \emptyset$, $\mathcal{F}_1 = \mathcal{F}_0 \cup \{\{a,b\}\}$, $\mathcal{F}_2 = \mathcal{F}_1 \cup \{\{b,c,d\}\}$, $\mathcal{F}_3 = \mathcal{F}_2 \cup \{\{e,f\}\}$. We deduce inductively the following equivalence classes $\{\{a,b,c,d,e,f\}\}$, then $\{\{a,b\},\{c,d,e,f\}\}$, then $\{\{a\},\{b\},\{c,d,e,f\}\}$ and finally $\{\{a\},\{b\},\{c,d\},\{e,f\}\}$. With a nice data structure such a computation can be performed in time $O(n\,\mathrm{Ack}^{-1}(n))$ where* Ack *is the Ackermann function[Cormen 1989]. Now, just observe that a set S is a boolean combination of sets in $\mathcal{F}$ if and only if it is a finite union of equivalence classes. Naturally deciding this last property can be done in linear time from the equivalence classes of $\equiv_{\mathcal{F}}$.*

4.3 Batof

Extracting geometrical properties (for instance linear constraints) from automata is a challenging problem. From a theoretical point of view, this problem has been solved in [Leroux 2005a] with a polynomial time algorithm computing Presburger formulae from automata representing Presburger sets. We implemented this algorithm in the tool Batof distributed in the tool suite TaPAS, a suite of tools for the Presburger arithmetic already mentioned in the previous Chapter 3 that contains the tools Prestaf that encodes Presburger formulas into arithmetic automata. Prestaf and Batof provide together the very first implementation of an algorithm that normalizes Presburger formulas into unique canonical forms that only depend on the denoted sets. Intuitively in the normalization process, the minimization procedure for automata acts like a simplification procedure for the Presburger arithmetic. We experimented our implementation in two ways. First we computed a Presburger formula from arithmetic automata produced by the symbolic model-checker Fast. Then we tried the normalization process on Presburger formulas by applying first Prestaf to produce an arithmetic automaton and then Batof to compute back a Presburger formula.

Fast [Bardin 2006] is a symbolic model-checker for verifying reachability properties of infinite-state systems. Benchmarks of Fast are available in Section 3.3.3. We experimented the decoding algorithm over arithmetic automata encoding the reachability sets of 40 systems. Some results are presented in the following table. Column "$\rightarrow A$" is the time in seconds spent by Fast for computing an automaton A encoding the reachability set, a subset of $\mathbb{N}^n$, "$|A|$" is the number of states of A,

"$A \to \psi$" is the time in seconds to decode A into a Presburger formula ψ, $|\psi|$ is the length of ψ, and "l" is the number of "linear inequalities" extracted (0 means that the set can be denoted with equality and modular constraints).

system	$\to A$	$\lvert A\rvert$	$A \to \psi$	$\lvert\psi\rvert$	n	l
`Central Server system`	9.57	75	0.07	3716	12	10
`Consistency Protocol`	194.96	90	0.06	3811	11	11
`CSM - N`	24.38	66	0.05	3330	14	11
`Dekker ME`	17.73	200	0.01	13811	22	0
`Multipoll`	11.38	612	2.54	60995	18	19
`SWIMMING POOL`	71.03	553	0.14	1803	9	18
`Time-Triggered Protocol`	116.89	17971	90.53	984047	11	44

We also experimented the normalization of Presburger formulas on various examples. Four representative examples are presented: `modulo`, `large-coef`, `large-var` and `unsimplified`. Example `modulo` denotes the Cartesian product $(11\mathbb{N}) \times (7\mathbb{N}) \times (5\mathbb{N}) \times (3\mathbb{N})$, `large-coef` is the conjunction of three linear constraints with coefficients larger than 20, `large-var` is a linear constraint defined over 36 variables, and `unsimplified` is a disjunction of two sets of linear constraints with redundant constraints. Benchmark results are summarized in the following table. Columns have the following meaning: "example" provides the name of the formula ψ_0, "$|\psi_0|$" is the length of the formula ψ_0, "$\psi_0 \to A$" is the time in seconds spent by PRESTAF to produce an automaton A encoding the solutions of ψ_0. The other columns have been defined previously.

example	$\lvert\psi_0\rvert$	$\psi_0 \to A$	$\lvert A\rvert$	$A \to \psi$	$\lvert\psi\rvert$	n	l
`modulo`	40	0.1	4620	35.8	335	4	0
`large-coef`	167	1.7	147378	25.1	957	4	3
`large-var`	543	0.2	4320	12.7	2220	36	1
`unsimplified`	529	1.1	16530	1.3	1026	5	2

We observe that the computation of automata A from formulas ψ_0 takes less than 2 seconds. Moreover, the computation of formula ψ from A takes less than half a minute even on automata with more than 100k states. In practice the implementation slows down in presence of modular constraints. Note that $|\psi_0| < |\psi|$ in all our examples due to non-optimized outputs. However, even if $|\psi_0| < |\psi|$ in the last example `unsimplified`, the redundant linear constraints of ψ_0 no longer appears in ψ.

Contrary to previous results, we observe that $|\psi|$ is quite smaller than $|A|$. In both series of benchmarks formulas contain a few number of constraints (see the "l" column). In practice, we observe that small automata can encode complex boolean combinations but only a small number of "simple" Presburger sets.

4.4 Conclusion

We solved the decoding problem of arithmetic automata into Presburger formulas with a polynomial time algorithm. We presented an high level viewpoint of the algorithm since the complete solution requires a lot of sub-algorithms for many difficult problems. The complete solution is available in an internal report of 134 pages[Leroux 2006]. Note that this paper opens research perspectives and interesting problems. We implemented these algorithms in the tool Batof. The implementation is efficient since we decoded in less than one minute an automaton with more than 100k states. These good results show that it is possible to implement a decision procedure for the Presburger arithmetic that can switch between formulas and automata depending on the most suitable symbolic representation during the computation.

In the future we are interested in developing hybrid representations that can manipulate arithmetic automata with some parts that are implicitly represented by Presburger formulas. With Bernard Boiglot and Julien Brusten (Institut Montéfiore, Belgium) we started working on this problem and published some preliminary results in [Boigelot 2009].

From a theoretical viewpoint, we are interested in characterizing the structure of arithmetic automata encoding Presburger sets based on properties satisfied by the syntactic monoid [Pin 1996]. In fact, in [Leroux 2008b], we proved that the arithmetic automata obtained from minimal deterministic ones encoding Presburger sets by modifying the initial states and the sets of accepting states still encode Presburger sets.

Part II

Reachability Problems

Chapter 5

Presburger Counter Machines

A *Counter Machine* is a finite state machine equipped with a finite set of counters that hold natural numbers. Figure 5.1 provides a counter machine computing the Syracuse sequence. Starting from the configuration $(\texttt{ini}, 10)$ this machine exhibits the following run:

$$\texttt{ini}, 10 \;\to\; \texttt{even}, 10 \;\to\; \texttt{ini}, 5 \;\to\; \texttt{odd}, 5 \;\to\; \texttt{ini}, 16 \;\to\; \texttt{even}, 16 \;\to\; \texttt{ini}, 8$$

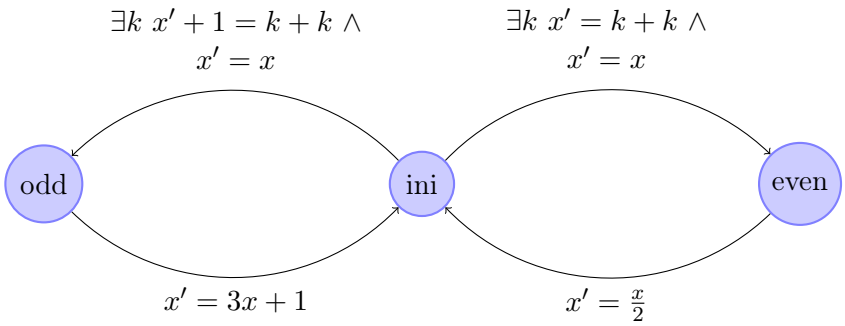


Figure 5.1: A counter machine computing the Syracuse sequence.

In this thesis transitions are labeled by binary relations over $\mathbb{N}^n$ definable in the Presburger arithmetic.

Remark 5.0.1. *Note that* $x' = 3x + 1$ *and* $x' = \frac{x}{2}$ *simply denotes the formulas* $x' = x + x + x + 1$ *and* $x' + x' = x$.

Definition 5.0.2. *A* Presburger Counter Machine (PCM) *with* n *counters is a tuple* $M = (Q, \mathbf{x}, \mathbf{x}', T)$ *where* Q *is a non-empty finite set of elements called* control states, $\mathbf{x}, \mathbf{x}' \in X^n$ *are vectors of distinct variables such that* $\mathrm{var}(\mathbf{x}) \cap \mathrm{var}(\mathbf{x}') = \emptyset$, *and* T *is a finite set of triples called* transitions *of the form* (p, ψ, q) *where* $p, q \in Q$ *and* ψ *is a Presburger formula such that* $\mathrm{var}(\psi) \subseteq \mathrm{var}(\mathbf{x}) \cup \mathrm{var}(\mathbf{x}')$.

A *trace* is a word $\tau \in T^*$ of the form $\tau = (q_0, \psi_1, q_1) \dots (q_{k-1}, \psi_k, q_k)$. A *configuration* of M is a couple $c \in Q \times \mathbb{N}^n$. We denote by $\mathrm{conf}(M)$ the set of configurations of M. A *run* is a non-empty word $\rho = c_0 \dots c_k$ of configurations $c_j = (q_j, \mathbf{m}_j)$ such that there exists a trace $\tau = (q_0, \psi_1, q_1) \dots (q_{k-1}, \psi_k, q_k)$ and a sequence $v_1, \dots, v_k$ of valuations $v_j \in \mathrm{model}(\psi_j)$ such that $\mathbf{m}_{j-1} = v_j(\mathbf{x})$, $\mathbf{m}_j = v_j(\mathbf{x}')$ for every $1 \leq j \leq k$.

The configurations c_0 and c_k are called the *source* and *target* of ρ, and we say that ρ is a run from c_0 to c_k labeled by τ. Given a trace τ, we introduce the binary relation $\xrightarrow{\tau}$ over the set of configurations defined by $c \xrightarrow{\tau} c'$ if there exists a run ρ from c to c' labeled by τ. We also denote by $\xrightarrow{\tau}$ the empty relation if τ is a word in T^* that is not a trace.

Remark 5.0.3. *The semantics of a PCM can also be defined thanks to* Labeled Transitions Systems (LTS).

Remark 5.0.4. *The set of traces of a PCM is a regular language.*

5.1 The Reachability Problem

The verification problem often reduces to the reachability problem. Whereas reachability properties are algorithmically checkable for *finite-state* systems (and efficient implementations exist), the situation is more complex for *infinite-state* systems. Formally, the *reachability problem for Presburger counter machines* consists in deciding for a triple (c, M, c') where c, c' are two configurations of a Presburger counter machine M if there exists a run from c to c'.

Remark 5.1.1. *The reachability problem for Presburger counter machines has applications outside the verification of infinite state systems. Several logics for data words over an infinite alphabet have been defined by [Bojańczyk 2006], [Demri 2009], and [Figueira 2009]. These works provide some links between logics and classes of Presburger counter machines. For instance, [Bojańczyk 2006] show that reachability problem for vector addition systems with states is equivalent to the satisfiability problem for the first-order logic over data words restricted to two individual variables.*

This problem is clearly recursively enumerable thanks to the algorithm that enumerates all the possible words ρ of configurations of M and check if ρ is a run from c to c'. In that case the algorithm terminates and returns a positive answer, i.e. the fact that there exists a run from c to c'. However, the reachability problem is undecidable even for the subclass of *Minsky machines* with 2 counters [Minsky 1967].

A *Minsky machine* is a Presburger counter machine $(Q, \mathbf{x}, \mathbf{x}', T)$ such that transitions are labeled by Presburger formulas ψ of the form $\mathbf{x}' = \mathbf{x} + \varepsilon \mathbf{e}_i$ where $\mathbf{e}_i = (0, \ldots, 0, 1, 0, \ldots, 0)$ is the ith unit vector and $\varepsilon \in \{-1, 0, 1\}$ denotes a decrement, no-operation, increment of the ith counter, or the Presburger formula $\mathbf{x}(i) = 0 \ \wedge \ \mathbf{x}' = \mathbf{x}$ that denotes a test to zero of the ith counter.

A result of complexity or decidability is important to understand the theoretical limits of a problem. However, for an important problem like the reachability problem, an undecidability result cannot be the final solution to the problem. In fact, in practice the Presburger counter systems we consider have no reason to be the must difficult instances of the problem. With this point of view, we are interested in classes of Presburger counter machines for which the reachability problem becomes

decidable. In this thesis we consider Presburger counter machines such that the reachability problem can be decided with the Presburger arithmetic. More formally we are interested by instances (c, M, c') of the reachability problem such that either c' is reachable from c or such that there exists an inductive invariant definable in the Presburger arithmetic proving that c' is not reachable from c.

Let us introduce the definition of *inductive invariants*. Given a language $L \subseteq T^*$ we denote by $\xrightarrow{L}$ the binary relation over the configurations defined by $c \xrightarrow{L} c'$ if there exists $\tau \in L$ such that $c \xrightarrow{\tau} c'$. Given a Language $L \subseteq T^*$ and sets $C, C' \subseteq \mathrm{conf}(M)$ of configurations, we introduce the sets $\mathrm{Post}^L(C)$ and $\mathrm{Pre}^L(C)$ of configurations defined by:

$$\mathrm{Post}^L(C) = \{c' \in \mathrm{conf}(M) \mid \exists c \in C \ c \xrightarrow{L} c'\}$$
$$\mathrm{Pre}^L(C') = \{c \in \mathrm{conf}(M) \mid \exists c' \in C' \ c \xrightarrow{L} c'\}$$

The sets $\mathrm{Post}^{T^*}(C)$ and $\mathrm{Pre}^{T^*}(C')$ are called the *forward reachability set* from C and the *backward reachability set* from C'. These sets are denoted by $\mathrm{Post}^*(C)$ and $\mathrm{Pre}^*(C')$. The binary relation $\xrightarrow{T^*}$ is called the *reachability relation* of M and it is denoted by $\xrightarrow{*}$. A set of configurations $C \subseteq \mathrm{conf}(M)$ is called a *forward inductive invariant* if $C = \mathrm{Post}^*(C)$. Symmetrically, a set of configurations $C' \subseteq \mathrm{conf}(M)$ is called a *backward inductive invariant* if $C' = \mathrm{Pre}^*(C')$.

Remark 5.1.2. *Forward and backward inductive invariants are dual since any partition $C \cup C'$ of the set of configurations satisfies C is a forward inductive invariant if and only C' is a backward inductive invariant.*

Example 5.1.3. *Let us come back to the PCM depicted in Figure 5.1. The* Syracuse problem *consists to decide if for every $n \in \mathbb{N}$ there exists a run from $(\mathtt{ini}, n)$ to $(\mathtt{ini}, 1)$. This problem is equivalent to the inclusion $\{\mathtt{ini}\} \times \mathbb{N} \subseteq \mathrm{Pre}^*(\{(\mathtt{ini}, 1)\})$. This problem is open.*

We are interested by inductive invariants that can be denoted by formulas in the Presburger arithmetic. A *set of configurations* $C \subseteq \mathrm{conf}(M)$ is said to be *definable in the Presburger arithmetic* if the unique decomposition of C into $C = \bigcup_{q \in Q} \{q\} \times \mathbf{S}_q$ is such that $\mathbf{S}_q$ is definable in the Presburger arithmetic. In that case, a sequence $(\psi_q(\mathbf{x}_q))_{q \in Q}$ of presentations $\psi_q(\mathbf{x}_q)$ of $\mathbf{S}_q$ is called a *presentation* of C (note that presentations are defined in Chapter 2).

Definition 5.1.4. *A* certificate of safety *for an instance (c, M, c') of the reachability problem is a presentation of a forward inductive invariants that contains c but not c'.*

Since the Presburger arithmetic is a decidable logic we deduce the following theorem.

Theorem 5.1.5. *We can decide if presentations of Presburger sets of configurations are certificates of safety.*

Proof. Let us consider a presentation $(\psi_q(\mathbf{x}_q))_{q\in Q}$ of a Presburger set of configurations C of a Presburger counter machine $M = (Q, \mathbf{x}, \mathbf{x}', T)$. By renaming the variables occuring in the presentation, we can assume without loss of generality that $\mathrm{var}(\mathbf{x}_q)$ is disjoint from $\mathrm{var}(\mathbf{x})$, $\mathrm{var}(\mathbf{x}')$, and $\mathrm{var}(\mathbf{x}_p)$ for every $p \neq q$. Now just observe that C is a forward inductive invariant if and only if for every transition $(p, \psi, q) \in T$ the following formula is *valid* (i.e. its negation is unsatisfiable):

$$(\psi_p \wedge \mathbf{x}_p = \mathbf{x} \wedge \psi \wedge \mathbf{x}_q = \mathbf{x}') \Longrightarrow \psi_q$$

□

We deduce an algorithm for the reachability problem that takes as input a triple (c, M, c') where c, c' are configurations of the Presburger counter machine M, and enumerates the runs and the presentations of Presburger sets of configurations until it discovers either (1) a run from c to c' and returns that c' is reachable from c, or (2) a certificate of safety for (c, M, c') and returns that c' is not reachable from c. Note that this algorithm is correct but the termination is not guaranty. However for some classes of Presburger counter machines, this algorithm always terminates. This is for instance the case for the *Vector Addition Systems with States* (see Chapter 7), and the class of *lossy Presburger counter machines* introduced in the sequel.

5.2 Subclasses of Presburger Counter Machines

Lossy Presburger Counter Machines

Lossy Presburger counter machines were first introduced in [Mayr 2000, Mayr 2003] as a simpler subclass of the lossy channel systems where counters holds natural numbers rather than channels holding words of messages in transit. Intuitively a lossy PCM is a PCM such that the value of any counter can be decreased during the execution. More formally given a PCM M we introduce the order $\sqsubseteq$ over the configurations defined by $(q_1, \mathbf{m}_1) \sqsubseteq (q_2, \mathbf{m}_2)$ if $q_1 = q_2$ and $\mathbf{m}_1 \leq \mathbf{m}_2$ where $\leq$ is the component-wise extension of the usual order over $\mathbb{N}$. The PCM M is said to be *lossy* if $c' \xrightarrow{\tau} d'$ for every $c' \sqsupseteq c \xrightarrow{\tau} d \sqsupseteq d'$ where c', c, d, d' are configurations and τ is a trace.

Reachability sets of lossy PCM are proved to be definable in the Presburger arithmetic thanks to the Dickson's Lemma [Dickson 1913].

Lemma 5.2.1 (Dickson's Lemma)**.** *$(\mathbb{N}^n, \leq)$ is a well-ordered set, i.e. for every infinite sequence $(\mathbf{m}_j)_{j\in\mathbb{N}}$ of vectors in $\mathbb{N}^n$, there exists $j < k$ such that $\mathbf{m}_j \leq \mathbf{m}_k$.*

From the previous lemma we deduce that $\sqsubseteq$ is a well-order over the set of configurations. As a direct consequence, *upward closed sets* of configurations, i.e. sets $C \subseteq \mathrm{conf}(M)$ such that $c \leq d \wedge c \in C$ implies $d \in C$ for every $d \in \mathrm{conf}(M)$ are finite unions of sets of the form $\{q\} \times (\mathbf{m} + \mathbb{N}^n)$. Hence upward closed sets are definable in the Presburger arithmetic.

If c' is a configuration not reachable from c in a lossy counter machine, there exists a certificate of safety for (c, M, c'). The existence of such a certificate is obtained just by observing that $C = \text{Post}^*(\{c\})$ is a Presburger set of configurations. In fact, since $\text{conf}(M)\backslash C$ is upward closed, this set is definable in the Presburger arithmetic. In particular C is also definable in the Presburger arithmetic. So, the reachability problem is decidable for lossy PCM. This proof idea is present in [Schnoebelen 2010a]. In [Mayr 2000, Mayr 2003] a different proof based on the fact that we can compute a presentation of the backward reachability sets.

Remark 5.2.2. *The decidability of the reachability problem as previously presented is based on the fact that the forward reachability sets are definable in the Presburger arithmetic. Note however that we cannot effectively compute a presentation of this set since checking the finiteness of* $\text{Post}^*(\{c\})$ *is undecidable[Mayr 2000, Mayr 2003].*

Functional Presburger Counter Machines

A transition $t = (p, \psi, q)$ of a Presburger counter machine $M = (Q, \mathbf{x}, \mathbf{x}', T)$ is said to be *functional* if for every configurations c, c_1, c_2 such that $c \xrightarrow{t} c_1$ and $c \xrightarrow{t} c_2$ we have $c_1 = c_2$. A Presburger counter machine is said to be *functional* if every transition is functional. We decide if a transition $t = (p, \psi, q)$ is functional as follows. We consider a vector $\mathbf{z} \in X^n$ of distinct variables such that $\text{var}(\mathbf{z}) \cap (\text{var}(\mathbf{x}) \cup \text{var}(\mathbf{x}')) = \emptyset$. Now, just observe that t is functional if and only if the following Presburger formula is valid:

$$\forall \mathbf{x}\ \exists \mathbf{z}\ \forall \mathbf{x}'\ \psi \Rightarrow \mathbf{x}' = \mathbf{z}$$

Since the Presburger arithmetic is decidable, we deduce that the class of functional PCM is recursive.

Functional transitions can be geometrically characterized by introducing the semilinear sets [Ginsburg 1966].

Definition 5.2.3. *A* linear set *is a set of the form* $\mathbf{b} + \mathbb{N}\mathbf{p}_1 + \cdots + \mathbb{N}\mathbf{p}_k$ *where* $\mathbf{b} \in \mathbb{Z}^n$ *and* $\mathbf{p}_j \in \mathbb{Z}^n$. *A* semilinear set *is a finite union of linear sets.*

Theorem 5.2.4 ([Ginsburg 1966])**.** *A subset of* $\mathbb{N}^n$ *is definable in the Presburger arithmetic if and only if and only if it is semilinear.*

Remark 5.2.5. *The class of* almost semilinear sets *introduced in Chapter 7 is central for deciding the reachability problem for vector addition systems with states. This class is inspired by the* semilinear sets.

A function $f : \mathbb{N}^n \to \mathbb{N}$ partially defined over a set $\mathbf{S} \subseteq \mathbb{N}^n$ is said to be *definable in the Presburger arithmetic* if the set $\{(\mathbf{s}, f(\mathbf{s})) \mid \mathbf{s} \in \mathbf{S}\}$, called the *graph of* f is definable in the Presburger arithmetic with the classical identification of $\mathbb{N}^n \times \mathbb{N}$ with $\mathbb{N}^{n+1}$. Observe that a transition (p, ψ, q) of a Presburger counter machine

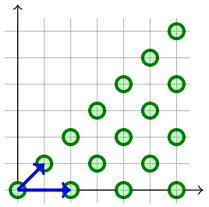


Figure 5.2: The linear set $(0,0) + \mathbb{N}(1,1) + \mathbb{N}(2,0)$.

$M = (Q, \mathbf{x}, \mathbf{x}', T)$ is functional if and only if $\{(v(\mathbf{x}), v(\mathbf{x}'(i))) \mid v \in \text{model}(\psi)\}$ is the graph of such a function for every $i \in \{1, \ldots, n\}$.

As an application of the semilinear sets, functions $f : \mathbb{N}^n \to \mathbb{N}$ definable in the Presburger arithmetic can be characterized as follows. A function $f : \mathbb{N}^n \to \mathbb{N}$ is said to be *piecewise-Presburger linear* if there exists a sequence $(\mathbf{S}_j)_{1 \leq j \leq k}$ of Presburger sets such that $\bigcup_{j=1}^{k} \mathbf{S}_j$ is equal to the definition domain of f, a sequence $(\mathbf{h}_j)_{1 \leq j \leq k}$ of vectors $\mathbf{h}_j \in \mathbb{Q}^n$ and a sequence $(c_j)_{1 \leq j \leq k}$ of rational values $c_j \in \mathbb{Q}$ such that $f(\mathbf{s}) = \mathbf{h}_j \cdot \mathbf{s} + c_j$ for every $\mathbf{s} \in \mathbf{S}_j$ and every $j \in \{1, \ldots, k\}$.

Theorem 5.2.6 (unpublished result of A.Finkel and J.Leroux)**.** *A function is definable in the Presburger arithmetic if and only if it is piecewise-Presburger linear.*

Proof. Let $f : \mathbb{N}^n \to \mathbb{N}$ be a function definable in the Presburger arithmetic. We denote by $\mathbf{S}$ is definition domain. The graph $\{(\mathbf{s}, f(\mathbf{s})) \mid \mathbf{s} \in \mathbf{S}\}$ of f is definable in the Presburger arithmetic. Hence it can be decomposed into $\bigcup_{L \in \mathcal{L}} L$ where $\mathcal{L}$ is a finite class of linear sets included in $\mathbb{N}^d \times \mathbb{N}$. We introduce the projection function $\pi : \mathbb{N}^d \times \mathbb{N} \to \mathbb{N}^d$ that projects away the last component. Observe that the definition domain of f is equal to $\bigcup_{L \in \mathcal{L}} \pi(L)$. In order to prove that f is piecewise-Presburger linear, it is sufficient to prove that for every linear set $L \in \mathcal{L}$ the set $\pi(L)$ is definable in the Presburger arithmetic and there exists $\mathbf{h} \in \mathbb{Q}^d$ and $c \in \mathbb{Q}$ such that $f(\mathbf{s}) = \mathbf{h} \cdot \mathbf{s} + c$ for every $\mathbf{s} \in \pi(L)$.

Let us consider such a linear set $L = (\mathbf{p}_0, c_0) + \mathbb{N}(\mathbf{p}_1, c_1) + \cdots + \mathbb{N}(\mathbf{p}_k, c_k)$. Observe that $\pi(L)$ is the linear set $\mathbf{p}_0 + \mathbb{N}\mathbf{p}_1 + \cdots + \mathbb{N}\mathbf{p}_k$. In particular $\pi(L)$ is a Presburger set. Let us introduce the vector space $\mathbf{V} = \mathbb{Q}\mathbf{p}_1 + \cdots + \mathbb{Q}\mathbf{p}_k$ generated by $\mathbf{p}_1, \ldots, \mathbf{p}_k$. From this generating sequence, we can extract a basis of the vector space $\mathbf{V}$. By reordering the sequence, without loss of generality we can assume that there exists $d \in \{0, \ldots, k\}$ such that $\mathbf{p}_1, \ldots, \mathbf{p}_d$ is a basis of $\mathbf{V}$. We consider the *linear form* $g : \mathbb{Q}^n \to \mathbb{Q}$ partially defined over $\mathbf{V}$ by $g(\sum_{j=1}^{d} \lambda_j \mathbf{p}_j) = \sum_{j=1}^{d} \lambda_j c_j$ for every sequence $(\lambda_j)_{1 \leq j \leq d}$ of rational values $\lambda_j \in \mathbb{Q}$. We observe that $\pi(L) \subseteq \mathbf{p}_0 + \mathbf{V}$. Let us prove that $f(\mathbf{s}) = c_0 + g(\mathbf{s} - \mathbf{p}_0)$ for every $\mathbf{s} \in \pi(L)$.

Since $\mathbf{s} \in \pi(L)$ we deduce that there exists a sequence $(n_j)_{1 \leq j \leq k}$ of natural numbers $n_j \in \mathbb{N}$ such that $\mathbf{s} = \mathbf{p}_0 + \sum_{j=1}^{k} n_j \mathbf{p}_j$. As $\mathbf{s} - \mathbf{p}_0 \in \mathbf{V}$, there exists a sequence $(\lambda_j)_{1 \leq j \leq k}$ of rational values $\lambda_j \in \mathbb{Q}$ such that $\mathbf{s} - \mathbf{p}_0 = \sum_{j=1}^{k} \lambda_j \mathbf{p}_j$ and $\lambda_j = 0$ for every $j > d$. Hence $g(\mathbf{s} - \mathbf{p}_0) = \sum_{j=1}^{k} \lambda_j c_j$. Let $a \in \mathbb{N}_{>0}$ such that

$a\lambda_j \in \mathbb{Z}$ for every j. Such an integer can be decomposed into $a\lambda_j = n_j^+ - n_j^-$ where $n_j^+, n_j^- \in \mathbb{N}$. We get:

$$\sum_{j=1}^{k} an_j\mathbf{p}_j = a(\mathbf{s}_0 - \mathbf{p}_0) = \sum_{j=1}^{k}(n_j^+ - n_j^-)\mathbf{p}_j$$

And in particular there exists $\mathbf{s}' \in \pi(L)$ such that:

$$\mathbf{p}_0 + \sum_{j=1}^{k}(n_j^- + an_j)\mathbf{p}_j = \mathbf{s}' = \mathbf{p}_0 + \sum_{j=1}^{k} n_j^+\mathbf{p}_j$$

Let us introduce $c_- = c_0 + \sum_{j=1}^{k}(n_j^- + an_j)c_j$ and observe that $(\mathbf{s}', c_-)$ is in the linear set $(\mathbf{p}_0, c_0) + \mathbb{N}(\mathbf{p}_1, c_1) + \cdots + \mathbb{N}(\mathbf{p}_k, c_k)$. As this linear set is included in the graph of f we get $c_- = f(\mathbf{s}')$. Symmetrically the natural number $c_+ = c_0 + \sum_{j=1}^{k} n_j^+ c_j$ satisfies $c_+ = f(\mathbf{s}')$. Therefore $c_+ = c_-$ and we get the following equality:

$$\sum_{j=1}^{k} n_j c_j = g(\mathbf{s} - \mathbf{p}_0)$$

Now, let $c = c_0 + \sum_{j=1}^{k} n_j c_j$ and observe that $(\mathbf{s}, c)$ is in the linear set $(\mathbf{p}_0, c_0) + \mathbb{N}(\mathbf{p}_1, c_1) + \cdots + \mathbb{N}(\mathbf{p}_k, c_k)$. As this linear set is included in the graph of f we get $c = f(\mathbf{s})$. We have proved that $f(\mathbf{s}) = c_0 + g(\mathbf{s} - \mathbf{p}_0)$ for every $\mathbf{s} \in \pi(L)$. Since $g : \mathbb{Q}^d \to \mathbb{Q}$ is a linear form there exists $\mathbf{h} \in \mathbf{V}$ such that $g(\mathbf{v}) = \mathbf{h} \cdot \mathbf{v}$ for every $\mathbf{v} \in \mathbf{V}$. Hence $f(\mathbf{s}) = \mathbf{h} \cdot \mathbf{s} + c$ for every $\mathbf{s} \in \pi(L)$ where $c = c_0 - \mathbf{h} \cdot \mathbf{p}_0$. □

Remark 5.2.7. *The function $f : \mathbb{N} \to \mathbb{N}$ defined by $f(x) = \frac{x}{2} + \frac{1}{2}$ if x is odd shows that we cannot restrict the vectors $\mathbf{h}_j$ and the rational values c_j to be in $\mathbb{Z}^d$ and $\mathbb{Z}$ respectively.*

Since the decomposition of a Presburger set into semilinear sets is effective and since the proof of the previous Theorem 5.2.6 is also effective, without modifying reachability problems, we can effectively replace transitions $t = (p, \psi, q)$ in a Presburger counter machine $M = (Q, \mathbf{x}, \mathbf{x}', T)$ by transitions of the form $(p, \psi_G \wedge \mathbf{x}' = M\mathbf{x} + \mathbf{a}, q)$ where ϕ_G is the Presburger formula such that $\mathrm{var}(\psi_G) \subseteq \mathrm{var}(\mathbf{x})$ denoting a guard, $M \in \mathbb{Q}^{n \times n}$ is a matrix, and $\mathbf{a} \in \mathbb{Q}^n$.

Remark 5.2.8. *The reachability problem for functional Presburger counter machines is undecidable since this class contains the Minsky machines.*

Translation Presburger Counter Machines

A transition $t = (p, \psi, q)$ is said to be a *translation* if there exists a vector $\mathbf{a} \in \mathbb{Z}^n$ such that $\mathbf{m}' = \mathbf{m} + \mathbf{a}$ for every $(p, \mathbf{m}) \xrightarrow{t} (q, \mathbf{m}')$. A *translation Presburger counter machine* is a Presburger counter machine with transitions that are translations.

Translation Presburger counter machines are functional. We decide that a transition $t = (p, \psi, q)$ is a translation as follows. We consider vectors $\mathbf{y}, \mathbf{y}' \in X^n$ of distinct variables such that $\text{var}(\mathbf{x}), \text{var}(\mathbf{x}'), \text{var}(\mathbf{y}), \text{var}(\mathbf{y}')$ are disjoint. Observe that t is a translation if and only if the following Presburger formula is satisfiable:

$$\forall \mathbf{x} \forall \mathbf{x}' \ \psi \Rightarrow \mathbf{x} + \mathbf{y} = \mathbf{x}' + \mathbf{y}'$$

Moreover from a valuation v that satisfies this formula we deduce a vector $\mathbf{a} = v(\mathbf{y}) - v(\mathbf{y}')$ such that ψ is equivalent to $\phi_G \wedge \mathbf{x}' = \mathbf{x} + \mathbf{a}$ where ϕ_G is the Presburger formula $\exists \mathbf{x}' \psi$. Intuitively, ψ_G is a guard and $\mathbf{a}$ is the vector added when the transition is executed.

The class of translation Presburger counter machines contains the class of Minsky machines and in particular the reachability problem is undecidable. However, for this class we can apply acceleration techniques, a method introduced in Chapter 6 that helps the termination of inductive computations of reachability sets.

Vector Addition Systems With States

A vector addition system with states (VASS) is a translation Presburger counter machines such that transitions are labeled by formulas of the form $\mathbf{x}' = \mathbf{x} + \mathbf{a}$ where $\mathbf{a} \in \mathbb{Z}^d$. The reachability problem for VASS is proved decidable in Chapter 7. In fact we prove that for every instance (c, M, c') of the reachability problem where M is a VASS, either c' is reachable from c or there exists a certificate of safety for (c, M, c').

Remark 5.2.9. *The reachability problem for lossy PCM is an Ackermann-complete problem [Schnoebelen 2010b] whereas the reachability problem is simply EXPSPACE-complete for lossy VASS [Rackoff 1978, Cardoza 1976].*

5.3 Conclusion

We introduced in this chapter the Presburger counter machines and some known subclasses. We provided a reduction of the functional Presburger counter machines based on the piecewise-Presburger linear functions.

We also introduced the reachability problem for the class of Presburger counter machines. Even if the problem is undecidable for restricted classes, it becomes decidable for the class of instances (c, M, c') such that either c' is reachable from c or there exists a certificate of safety for (c, M, c'). In fact for this class it is sufficient to enumerate in parallel all the possible runs and all the possible presentations of Presburger sets of configurations until either we discover a run from c to c' and in this case c' is reachable from c or we discover a certificate of safety for (c, M, c') and in this case c' is not reachable from c.

Naturally such an algorithm is useless in practice since such an enumeration can only consider a very small number of cases in a reasonable amount of time. In

the next Chapter 6 we introduce different semi-algorithms that try to decide the reachability problem without using such an enumeration.

Chapter 6

Good Semi-Algorithms

Many specialized algorithms have been designed to solve verification problems for various classes of Presburger counter machines (PCM). The reachability problem for Petri nets has been proved decidable [Mayr 1981b, Kosaraju 1982]. The binary reachability relation is effectively Presburger for reversible Petri nets [Taiclin 1968] and for BPP-nets [Esparza 1997], and the forward reachability set is effectively Presburger for cyclic Petri nets [Araki 1977], for persistent Petri nets [Landweber 1978, Mayr 1981a] and for regular Petri nets [Valk 1981]. The reachability sets are effectively Presburger for reversal-bounded Minsky machines [Ibarra 1978], for lossy VASS [Bouajjani 1999b] and for VASS with 2 counters [Hopcroft 1979]. It was later shown that reachability are still effectively Presburger for various extensions of VASS with 2 counters [Finkel 2000b, Finkel 2000a]. However, these methods suffer from serious drawbacks: (1) they cannot be easily extended or combined, (2) from an implementation perspective, a dedicated tool would be needed for each specialized algorithm, and (3) in practice, PCM rarely belong entirely to one of these classes. Thus, generic symbolic model-checking techniques for general (undecidable) classes have been developed and implemented.

Usually the instances (c, M, c') of the reachability problem solved by algorithms correspond either to the case c' is reachable from c or there exists a certificate of safety for (c, M, c') proving that c' is not reachable from c based on the Presburger arithmetic. This is the case for the previously mentioned algorithms but also for methods based on *abstract interpretations* for many numerical abstract domains like the convex polyhedra [Cousot 1978], the intervals [Cousot 1977], the DBM, the octagons [Miné 2001], and so on [Péron 2007]. In fact, when these methods succeed in computing a precise enough abstract value for proving the reachability problem, we can extract from these values certificates of safety definable in the Presburger arithmetic.

Remark 6.0.1. *The inductive invariants computed by ellipsoid abstract domains cannot be captured with the Presburger arithmetic [Feret 2004]. More generally, invariants requiring non-linear constraints are out of the scope of the Presburger arithmetic. Since the logic* $\mathrm{FO}(\mathbb{N}, +, *)$ *is undecidable, specific techniques are required to denote non-linear inductive invariants in a decidable formalism.*

In this section we present three methods for deciding the reachability problem for Presburger counter machines. We do not pretend to be exhaustive since these three methods only correspond to the ones for which we provided contributions.

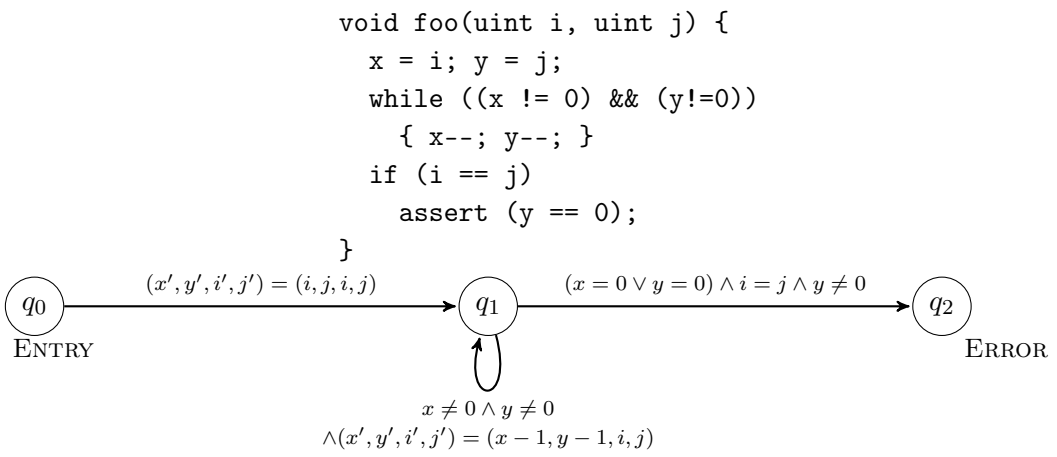


Figure 6.1: A PCM that models a C program.

All these methods provide algorithms without any termination guaranty in the general case. In fact, the reachability problem is undecidable even for the restrictive classes of Minsky machines. However, in practice these methods provide efficient algorithms for some instances. Such an algorithm that work well in practice but without termination guaranty is called a "*good semi-algorithm*". In Section 6.1 we present the computation of forward reachability sets based on acceleration techniques. In Section 6.2 we present a method combining abstract interpretation and acceleration techniques. Finally, in Section 6.3 we sketch up our contribution to the model-checking based on *Craig interpolations*.

6.1 Acceleration

Verification of reachability properties usually proceeds through an iterative fixpoint computation of the *forward reachability set* starting from the set of initial configurations. To help termination of this fixpoint computation, so-called *acceleration* techniques (or *meta-transitions*) are applied [Boigelot 1994, Boigelot 1997, Bouajjani 1999a, Finkel 2003, Finkel 2002]. Basically, acceleration consists in computing in a decidable formalism the effect of iterating loops. Symbolic model checkers Lash, TReX [Annichini 2001], and Fast [Bardin 2003] implement this approach. As an illustration, consider the PCM shown on Figure 6.1 taken from [Jhala 2006], and well-known in the CEGAR framework [Gulavani 2006]. We prove that the assertion is always satisfied as follows. The acceleration of the loop on q_1 provides the binary relation denoted by $\exists k\, (x', y', i', j') = (x - k, y - k, i, j)$ corresponding to the exact effect of iterating an arbitrary number of times the loop. Thanks to this meta-transition the forward reachability set from $\{q_0\} \times \mathbb{N}^4$ is an effectively computable Presburger set of configurations.

We implemented the tool Fast which follows the accelerated symbolic model-

checking framework. We experimented FAST with more than 40 systems modeled by PCM. In 80% of case, the tool succeeded in computing the reachability sets of PCM [Finkel 2002, Bardin 2003, Bardin 2004, Bardin 2006]. We tried to explain these good results by investigating termination of symbolic model-checking based on acceleration techniques for known classes of PCM having a Presburger definable set of reachable configurations. A natural notion in this framework is *flatness* [Fribourg 1997a, Comon 1998]: an initialized Presburger counter machine (c, M) is called *flat*[1] when its control flow graph can be "replaced", equivalently w.r.t. the reachability, by another one with no nested loops. We show that flatness is a necessary and sufficient condition for termination of reachability set computations by acceleration-based semi-algorithms. In particular, we get that accelerated symbolic model checkers terminate on a given system iff this system is flat (and a suitable search strategy is used). More formally, the definition of *flat PCM* is based on the notion of *bounded languages*. A language $L \subseteq T^*$ is said to be *bounded* if there exists a sequence $\sigma_1, \ldots, \sigma_k$ of words in T^* such that $L \subseteq \sigma_1^* \ldots \sigma_k^*$.

Definition 6.1.1 (Flat PCM)**.** *An initialized PCM (c, M) is said to be* flat *if there exists a bounded regular language $L \subseteq T^*$ such that* $\mathrm{Post}^*(\{c\}) = \mathrm{Post}^L(\{c\})$.

We then turn our attention to the analysis of flatness for known classes of Presburger counter automata with reachability sets definable in the Presburger arithmetic. We show that most of the known classes (in particular the ones cited in this chapter introduction) are flat [Leroux 2005b]. Our main technical contributions are the proofs of flatness for the following classes: reversal-bounded counter machines, reversible Petri nets and conflict-free Petri nets. In particular, we obtain that the binary reachability relation is effectively Presburger for conflict-free Petri nets. We also show that cyclic Petri nets, persistent Petri nets, regular Petri nets and *Inserting counter machines* are flat. As flatness implies effective computation (see the following Theorem 6.1.2) of the forward / binary reachability set, our results give new "uniform" proofs that these classes have reachability sets or relations effectivelly definable in the Presburger arithmetic. In particular, we obtain simpler proofs for reversal-bounded counter machines and reversible Petri nets.

Theorem 6.1.2 ([Finkel 2002])**.** *Forward reachability sets of flat initialized translation Presburger counter machines are effectively definable in the Presburger arithmetic.*

Remark 6.1.3. *The previous Theorem 6.1.2 is proved in [Finkel 2002] for the class of functional Presburger counter machines satisfying an algebraic conditions. This condition allows translations as well as* reset/transfer transitions.

It is remarkable that accelerated symbolic model checkers designed to analyse PCM, such as LASH and FAST, terminate on all these classes. From a practical

[1] Our notion of flatness is actually more general than in [Comon 1998]: there, a system is called flat when it contains no nested loops.

viewpoint, our approach has several benefits: (1) we can apply a *generic* algorithm, which was designed for a much larger class of (undecidable) systems, and (2) the — forward, backward and binary — reachability sets can be computed using the same generic algorithm.

Related Work.

The following approaches and tools have been developed to check correctness of PCM.

Forward reachability set computation. Tools ALV [Bultan 2001, Yavuz-Kahveci 2005], LASH and TREX [Annichini 2001] implement symbolic methods to compute the forward reachability set of Presburger counter machines. ALV provides two different symbolic representations for integer vectors: Presburger formula or automata as in FAST. Acceleration is available for the formula-based representation [Kelly 1995], but not for the automata-based representation. The tool is mostly used in backward computation or in approximated forward computation [Bartzis 2004]. LASH foundations are close to those of FAST, with similar symbolic representations and acceleration algorithms. The main difference is that LASH does not implement any circuit search and the user has to provide circuits to the tool. TREX [Annichini 2001] follows the same framework but uses rather different technologies.

Backward reachability set computation. One of the most interesting results is the computability of the backward reachability set for lossy Presburger counter machines with efficient symbolic representations. We can cite the work on covering sharing trees of Delzanno, Raskin and Van Begin [Delzanno 2004] and the tool BRAIN by Voronkov and Rybina [Rybina 2002]. These approaches are more specific than the one of FAST: computation is backward only[2], properties are reduced to upward-closed sets and systems are lossy.

Remark 6.1.4. *A computation of the backward reachability set for lossy VASS provides an algorithm with an optimal computational complexity[Bozzelli 2011].*

Reachability set approximation. Finally, some approaches relax the exactness of computation to ensure computation termination or at least simpler computational steps. However the superset obtained in the end may not be tight enough to decide the property. We can cite the classic tool HYTECH [Alur 1995], as well as the *abstract-check and refine* technique of Raskin et al. [Geeraerts 2005] to compute iteratively covering trees of monotonic Petri nets.

[2] FAST can also be used for backward reachability computations.

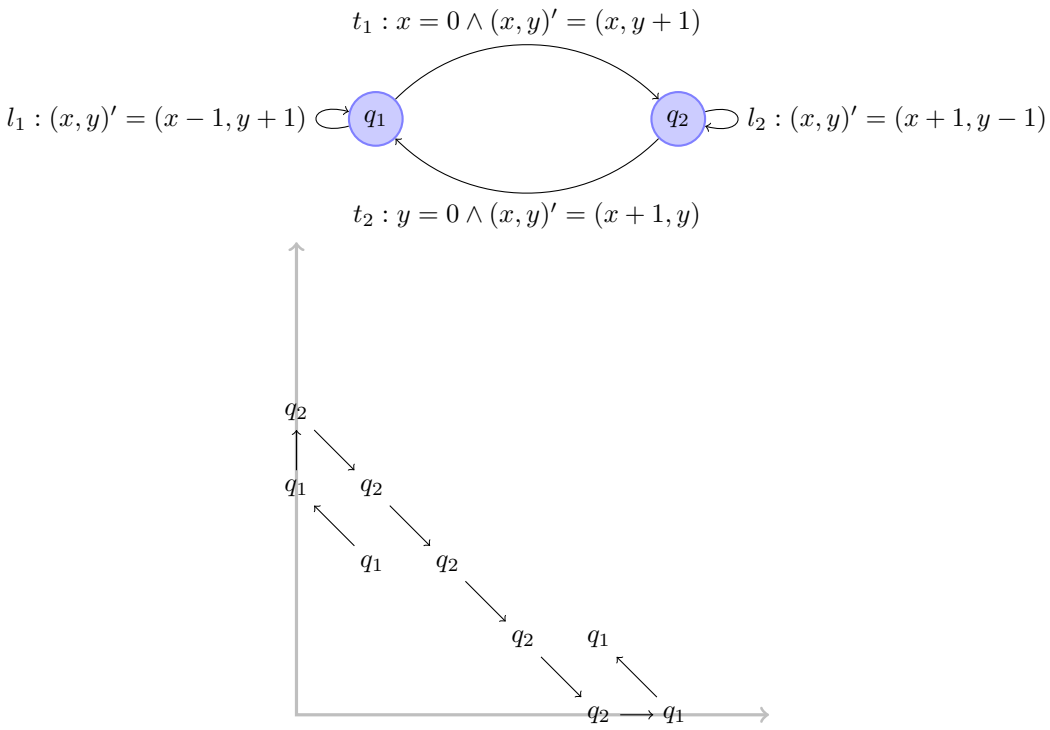


Figure 6.2: A non-flat translation Presburger counter machine.

Experimental Evaluation

With Sebastien Bardin and Gérald Point, we implemented the tool FAST based on acceleration techniques. We use a large pool of counter systems and case studies analyzed by tools ALV, BABYLON[3], BRAIN, LASH and TREX to evaluate FAST. They range from tricky academic puzzles like the swimming pool protocol [Fribourg 1997b] to industrial case studies like the cache coherence protocol for the Futurebus+.

FAST appears to be a very efficient tool for the forward computation of reachability sets of functional Presburger counter machines. In experiments, FAST performance is clearly superior to that of similar tools ALV, LASH and TREX (see [Bardin 2008]). Experimentals results are presented in Chapter 3.

Again, recall that it does not necessarily imply that FAST is better then the other tools for PCM validation since we restricted the experiments to exact forward computation while other approaches exist. Moreover, recall that we use restrictions of ALV and TREX which are primarily designed to handle different machines (TREX) or richer properties (ALV).

Even though it behaves well in practice, accelerated symbolic model-checking is only a *semi*-algorithm: it does not provide any guarantee of termination even if the forward reachability set is definable in the Presburger arithmetic. For instance,

[3] `http://www.ulb.ac.be/di/ssd/lvanbegin/CST`

iteration of loops is not sufficient for computing the forward reachability set of the PCM depicted in Figure 6.2, with initial configuration $(q_1, 0, 0)$ whereas this set is clearly definable in the Presburger arithmetic. In order to improve the acceleration frameworks, a first step consists in characterizing classes for which the generic accelerated semi-algorithm fails to terminate whereas the reachability sets are effectively definable in the Presburger arithmetic. The class of VASS with forward reachability sets definable in the Presburger arithmetic seems to be such a good class. In fact this class is known to be recursive and there exists an algorithm computing a presentation of the forward reachability set in that case (unpublished result of Hauschildt and Lambert). Naturally flat VASS have forward reachability sets definable in the Presburger arithmetic thanks to Theorem 6.1.2. Up to our knowledge, we do not know if this implication is an equivalence.

6.2 Abstract Acceleration

Model-checking safety properties on a given system usually reduces to the computation of a precise enough invariant of the system. In traditional symbolic verification, the set of all reachable (concrete) configurations is computed iteratively from the initial states by a standard fix-point computation. This reachability set is the most precise invariant, but quite often (in particular for software systems) a much coarser invariant is sufficient to prove correctness of the system. Data-flow analysis, and in particular abstract interpretation [Cousot 1977], provides a powerful framework to develop analysis for computing such approximate invariants.

A data-flow analysis of a program basically consists in the choice of a (potentially infinite) complete lattice of data properties for program variables together with transfer functions for program instructions. The merge over all path (MOP) solution, which provides the most precise abstract invariant, is in general over-approximated by the minimum fix-point (MFP) solution, which is computable by Kleene fix-point iteration. However the computation may diverge and *widening/-narrowing operators* are often used in order to enforce convergence at the expense of precision [Cousot 1977, Cousot 1992]. While often providing very good results, the solution computed with widenings and narrowings may not be the MFP solution. This may lead to abstract invariants that are too coarse to prove safety properties on the system under check.

Techniques to help convergence of Kleene fix-point iterations have also been investigated in symbolic verification of infinite-state systems. In these works, the objective is to compute the (potentially infinite) reachability set for automata with variables ranging over unbounded data, such as counters, clocks, stacks or queues. So-called *acceleration* techniques (or *meta-transitions*) have been developped [Boigelot 1994, Boigelot 1997, Comon 1998, Finkel 2003, Finkel 2002] to speed up the iterative computation of the reachability set. Basically, acceleration consists in computing in one step the effect of iterating a given loop (of the control flow graph). Accelerated symbolic model checkers such as LASH,

TReX [Annichini 2001], and Fast [Bardin 2003] successfully implement this approach.

Our contribution. In [Leroux 2007a], we extended acceleration techniques to data-flow analysis and we applied these ideas to interval analysis. Acceleration techniques for (concrete) reachability set computations may be equivalently formalized "semantically" in terms of control-flow path languages [Leroux 2005b] or "syntactically" in terms of control-flow graph unfoldings [Bardin 2005]. We extended these concepts to the MFP solution in a generic data-flow analysis framework, and we established several links between the resulting notions. It turns out that, for data-flow analysis, the resulting "syntactic" notion, based on graph *flattenings*, is more general that the resulting "semantic" notion, based on restricted regular expressions. We then propose a generic flattening-based semi-algorithm for computing the MFP solution. This semi-algorithm may be viewed as a generic template for applying acceleration-based techniques to constraint solving.

We then showed how to instantiate the generic flattening-based semi-algorithm in order to obtain an efficient constraint solver[4] for integers, for a rather large class of constraints using addition, (monotonic) multiplication, factorial, or any other *bounded-increasing* function. The intuition behind our algorithm is the following: we propagate constraints in a breadth-first manner as long as the least solution is not obtained, and variables involved in a "useful" propagation are stored in a graph-like structure. As soon as a cycle appears in this graph, we compute the least solution of the set of constraints corresponding to this cycle. It turns out that this acceleration-based algorithm always terminates in cubic-time.

In [Leroux 2007b], we aimed at developing methods that speed up the iterative computation of the MFP-solution, *without any loss of precision*. We focus on dataflow analysis with the complete lattice of convex sets of real vectors. A set $\mathbf{S} \subseteq \mathbb{R}^d$ is said to be *convex* if for every $\mathbf{s}_1, \mathbf{s}_2 \in \mathbf{S}$ and for every $\lambda_1, \lambda_2 \in \mathbb{R}_{\geq 0}$ such that $\lambda_1 + \lambda_2 = 1$ then $\lambda_1\mathbf{s}_1 + \lambda_2\mathbf{s}_2 \in \mathbf{S}$. We introduced the class of *guarded translation systems* (*GTSs*). This class intuitively represents programs where conditions are convex sets and transformations are restricted to translations.

Definition 6.2.1. *A* guarded translation system (GTS) *is a graph* $G = (Q, T)$ *where* Q *is a non-empty finite set of states and* T *is a finite set of transition* $(p, (\mathbf{x}, \mathbf{y}) \mid \psi \wedge \mathbf{y} = \mathbf{x} + \mathbf{a}, q)$ *where* $\mathbf{x}, \mathbf{y} \in X^d$ *are vectors of dictinct variables,* $\mathbf{a} \in \mathbb{Q}^d$*, and* ψ *is a first order formula of* $\mathrm{FO}(\mathbb{R}, +, \leq)$ *such that* $\mathrm{var}(\psi) \subseteq \mathrm{var}(\mathbf{x})$ *and such that* $\mathbf{x} \mid \psi$ *denotes a convex subset of* $\mathbb{R}^d$*.*

Recast in our setting, the (exact) acceleration techniques mentioned above consist in computing the merge over all path (MOP) solution along some (simple) cycle, which we call *MOP-acceleration*. We show that the MOP-acceleration of any cycle is a convex sets with a computable presentation. However MOP-acceleration

[4] By solver, we mean an algorithm computing the least solution of constraint systems.

is not in general sufficient to guarantee termination of the Kleene fix-point iteration, even for cyclic GTSs (GTSs with control flow graphs reduced to single cycles). We therefore investigate *MFP-acceleration*, which basically amounts to computing the MFP-solution of the system restricted to a given cycle. In other words, MFP-acceleration directly gives the MFP-solution for cyclic GTSs.

We obtained a surprisingly simple expression of the MFP-acceleration for cycles with a unique initial location. This characterization shows that the MFP-acceleration is definable in FO $(\mathbb{R}, +, \leq)$ and a presentation is computable. This result cannot be extended to arbitrary cycles, as we give a 3-dim (i.e. three real-valued variables) cyclic example where the MFP-solution is not definable in FO $(\mathbb{R}, +, \leq)$ neither FO $(\mathbb{R}, +, *, \leq)$. We then focused on 2-dim GTSs and we proved that even if the MFP-solution is not definable in FO $(\mathbb{R}, +, \leq)$ then it is definable in the decidable logic FO $(\mathbb{R}, +, *, \leq)$. Even for cyclic GTSs in dimension 2, the MFP-solution requires this extended logic.

Related work. In [Karr 1976], Karr presented a polynomial-time algorithm that computes the set of all affine relations that hold in a given control location of a (numerical) program. Recently, the complexity of this algorithm was revisited in [Müller-Olm 2004] and a fine upper-bound was presented. Many refinements of this original widening operator have since been studied [Bagnara 2005] to limit the loss of precision. Recently Gonnord and Halbwachs [Gonnord 2006] introduced the notion of abstract-acceleration as a complement to widening for linear relation analysis. We show that while maintaining the same computational complexity, our MFP-acceleration is "better" than abstract-acceleration in the sense that MFP-acceleration enforces convergence of the Kleene fix-point iteration strictly more often than abstract-acceleration. On another hand [Gonnord 2006] also investigates acceleration of multiple loops and the combination of translations and resets.

For interval constraints with affine transfer functions, the exact least solution may be computed in cubic-time [Su 2004]. Strategy iteration was proposed in [Costan 2005] to speed up Kleene fix-point iteration with better precision than widenings and narrowings, and this approach has been developped in [T. Gawlitza 2007] for interval constraint solving with full addition, multiplication and intersection. Strategy iteration may be viewed as an instance of our generic flattening-based semi-algorithm. The class of interval constraints that we consider in this paper contains the one in [Su 2004] (which does not include interval multiplication) but it is more restrictive than the one in [T. Gawlitza 2007].

6.3 Interpolation

Counterexample-guided abstraction refinement (CEGAR) paradigm [Clarke 2003] makes it possible to perform efficient verification of real-life software. In this approach, an initial coarse predicate abstraction [Graf 1997] of the concrete model is first derived and explored by a model-checker. If no error trace is found, the system

is said to be '*safe*'. If an abstract error-trace is found, it is checked against the concrete model. When the error also exists in the concrete model, the system is said to be '*unsafe*' and a concrete error trace is provided to the operator. Finally, when the error is found to be spurious, a proof of the spuriousness of the trace is used to build a refinement of the abstraction.

Interpolation-based model-checking [McMillan 2003, McMillan 2005] is a CEGAR framework where checking the error-trace is performed using decision procedures for various logics and refinements are produced by computing interpolants, which provide sets of predicates needed to invalidate the considered spurious error-traces in the abstraction. Interpolation-based model-checking technique has been proved robust and efficient. Recently, a '*lazy*' [Henzinger 2002] approach of this method has been introduced [McMillan 2006], allowing it to deal with infinite systems.

Let us first recall the notion of interpolants as introduced by McMillan[McMillan 2006] in the context of Presburger counter machines.

Definition 6.3.1 (Interpolant)**.** *An* interpolant *for a non-empty sequence* $(\phi_1, \ldots, \phi_k)$ *of Presburger formulas* ϕ_j *is a sequence* $(\psi_0, \ldots, \psi_k)$ *of Presburger formulas such that* ψ_0 *is valid,* ψ_k *is unsatifiable, such that for every* $j \in \{0, \ldots, k\}$*:*

$$\mathrm{var}(\psi_j) \subseteq (\mathrm{var}(\phi_1) \cup \ldots \cup \mathrm{var}(\phi_j)) \cap (\mathrm{var}(\phi_{j+1}) \cup \ldots \cup \mathrm{var}(\phi_k))$$

and the following formula is valid for every $j \in \{1, \ldots, k\}$*:*

$$\psi_{j-1} \wedge \phi_j \Longrightarrow \psi_j$$

Remark 6.3.2. *The previous definiton of interpolants extends the notion of* Craig interpolants *for a pair* (ϕ_A, ϕ_B) *of formulas such that* $\phi_A \wedge \phi_B$ *is unsatisfiable [Pudlák 1995].*

Theorem 6.3.3 ([McMillan 2006])**.** *A non empty sequence* $(\phi_1, \ldots, \phi_k)$ *of Presburger formulas* ϕ_j *admits an interpolant if and only if* $\phi_1 \wedge \ldots \wedge \phi_k$ *is unsatisfiable.*

The lazy interpolation model-checking is based on the fact that if a trace is not the label of a run then there exists an interpolant proving this property from which interesting predicates can be extracted.

Definition 6.3.4. *A trace* $\tau = t_1 \ldots t_k$ *of a Presburger counter machine is said to be* spurious *if there does not exist a run labeled by* τ*.*

We characterize spurious traces with interpolants as follows. Let us consider a non-empty trace $\tau = (q_0, \theta_1, q_1) \ldots (q_{k-1}, \theta_k, q_k)$ of a Presburger counter machine $M = (Q, \mathbf{x}, \mathbf{y}, T)$. Let us consider a sequence $(\mathbf{x}_j)_{0 \leq j \leq k}$ of vectors $\mathbf{x}_j \in X^n$ of distinct variables such that the sets $\mathrm{var}(\mathbf{x}_j)$ are disjoint. By renaming the variables of θ_j we deduce a Presburger formula ϕ_j such that $\mathrm{var}(\phi_j) \subseteq \mathrm{var}(\mathbf{x}_{j-1}) \cup \mathrm{var}(\mathbf{x}_j)$ and

such that $\theta_j(\mathbf{x}, \mathbf{y})$ and $\phi_j(\mathbf{x}_{j-1}, \mathbf{x}_j)$ denotes the same Presburger set. We observe that τ is spurious if and only if the following Presburger formula is unsatisfiable:

$$\phi_1 \wedge \ldots \wedge \phi_k$$

In that case, theorem 6.3.3 shows that there exists an interpolant $(\psi_0, \ldots, \psi_k)$ for $(\phi_1, \ldots, \phi_k)$. Since $\mathrm{var}(\phi_1 \wedge \ldots \wedge \phi_j) \subseteq \mathrm{var}(\mathbf{x}_0) \cup \ldots \cup \mathrm{var}(\mathbf{x}_j)$ and $\mathrm{var}(\phi_{j+1} \wedge \ldots \wedge \phi_k) \subseteq \mathrm{var}(\mathbf{x}_j) \cup \ldots \cup \mathrm{var}(\mathbf{x}_k)$ we deduce that $\mathrm{var}(\psi_j) \subseteq \mathrm{var}(\mathbf{x}_j)$. In particular $\psi_j(\mathbf{x}_j)$ denotes a Presburger set $\mathbf{S}_j \subseteq \mathbb{N}^n$. The sequence $(\mathbf{S}_j)_{0 \leq j \leq k}$ is in fact an inductive invariant for the trace τ. In fact, since ψ_0 is valid and ψ_k is unsatisfiable we get $\mathbf{S}_0 = \mathbb{N}^n$ and $\mathbf{S}_k = \emptyset$. Moreover, as $\psi_{j-1} \wedge \phi_j \Rightarrow \psi_j$ is valid we get the following inclusion for every $j \in \{1, \ldots, k\}$:

$$\mathrm{Post}^{t_j}(\{q_{j-1}\} \times \mathbf{S}_{j-1}) \subseteq \{q_j\} \times \mathbf{S}_j$$

That explains why the sequence $(\{q_j\} \times \mathbf{S}_j)_{0 \leq j \leq k}$ is kind of *inductive invariant* proving that τ is a spurious trace. The lazy interpolation-based model-checking consider the presentations $\psi_j(\mathbf{x}_j)$ in order to refine abstract models. We do not present in the sequel the way these predicates are used (The reader can have a look at the algorithms presented in [McMillan 2006]) but we focus on the computation of interpolants.

Theorem 6.3.3 can be proved easilly thanks to Presburger formulas ψ_j obtained from $\phi_1 \wedge \ldots \wedge \phi_j$ by quantifying existentially variables not in $\phi_j \wedge \ldots \wedge \phi_k$. Even if the formulas occuring in interpolants obtained this way have small length, they are quantified. In practice, we are interested in quantifier free interpolants since manipulating formulas in this fragment is usually more efficient (see Chapter 3). Unfortunately, even if the Presburger arithmetic admits a quantifier elimination algorithm, in practice quantifier-free interpolants obtained this way are unduly complicated. For instance let us consider the sequence (ϕ_1, ϕ_2, ϕ_3) of Presburger formulas of the form $\phi_1 = (x = 0)$, ϕ_2 is a very complex Presburger formula in wich x does not occur and ϕ_3 is the conjunction of $x \neq 0$ and a complex Presburger formula sharing variables with ϕ_2. A possible interpolant is $(\psi_0, \psi_1, \psi_2, \psi_3)$ where ψ_0 is the true formula, ψ_3 the false formula, and ψ_1, ψ_2 are equal to the formula $(x = 0)$. Due to the high complexity of ϕ_2 and ϕ_3, a quantifier-free interpolant obtained by applying a quantifier elimination algorithm will provide much complex interpolants. Craig interpolation has become a key ingredient in many symbolic model checkers, serving as an approximative replacement for expensive quantifier elimination [McMillan 2005]. The application of Craig interpolants in lieu of quantifier elimination relies on the availability of an effective *interpolating decision procedure.*

We focus on an interpolating decision procedure for the *quantifier-free fragment of Presburger Arithmetic (QFPA for short)*, that is linear arithmetic over the integers, a theory which is a good fit for the analysis of Presburger counter machines. Interpolating decision procedures typically derive the interpolant from a proof of

inconsistency of $\phi_1 \wedge \ldots \wedge \phi_k$, which in turn is computed by a decision procedure for the underlying logic. Decision problems arising in software analysis are often large, and call for a scalable algorithm. The most efficient decision procedures for the quantifier-free fragment of the Presburger arithmetic known today use the Simplex algorithm in combination with a variant of the *branch-and-bound technique*. The Simplex algorithm is used to solve the *relaxed problem*, in which the variables are permitted to take fractional values. In case a variable x obtains the fractional value r, branch-and-bound will consider the two sub-problems in which $x \leq \lfloor r \rfloor$ or $x \geq \lceil r \rceil$, respectively. The original problem has an integer solution iff one of the two sub-problems has a solution. Branch-and-bound is incomplete by itself, and usually augmented by a *cutting-plane* technique, e.g., *Gomory's cutting planes*. An instance of an efficient implementation of these techniques is the SMT-solver Z3 [Dutertre 2006].

In principle, any cut-based decision procedure for Presburger can be used for the computation of interpolants. The primary problem is computational cost: for the most common cut rules (in particular for Gomory's cutting planes) it is possible to construct cases where the derivation of interpolants from proofs has exponential complexity. This high complexity is caused by *mixed cuts*, which involve rounding (rational) constant terms of inequalities that are derived from formulas ϕ_i and ϕ_j with $i \neq j$. Intuitively, interpolating calculi rely on identifying which parts in $\phi_1, \ldots, \phi_k$ are contributing to an intermediate argument; additional effort is required when rounding intermediate arguments derived from both two different formulas ϕ_i, ϕ_j.

Our Contribution.

We introduced a novel interpolating decision procedure for the full QFPA fragment[Kroening 2010]. Our algorithm computes in polynomial time interpolants for two classes of constraints (i) conjunctions of inequality constraints unsatisfiable over the rationals, and (ii) conjunctions of equality and divisibility constraints unsatisfiable over the integers. For the full QFPA fragment, the algorithm is exponential in the worst case. This complexity is proved tight since we exhibit formulas such that every interpolant is exponentially large. Moreover the algorithm improves the doubly exponential upper bound complexity known for the computation of interpolants based on the elimination of blocks of quantifiers [Weispfenning 1997]. Our general procedure integrates efficient reasoning and interpolation for equalities by means of a transformation of matrices into Smith Normal Form, which resembles a known procedure for interpolating linear diophantine equations [Jain 2008]. For reasoning about inequalities, our procedure uses a complete version of the branch-and-cut principle that avoids mixed cuts and therefore allows interpolant extraction from proofs in polynomial time. Since the proof size is exponentially large in the worst case, we deduce an exponential upper bound for the runtime of the algorithm.

Related Work.

Interpolation procedures have been proposed for various fragments of linear integer arithmetic. McMillan considers the logic of difference-bound constraints [McMillan 2006]. This logic, a fragment of QFPA, is decidable by reduction to rational arithmetic. As an extension, Cimatti et al. [Cimatti 2009] present an interpolation procedure for the unit two variables per inequality (UTVPI) fragment of linear integer arithmetic. Both fragments allow efficient reasoning and interpolation, but are not sufficient to express many typical program constructs, such as integer division. In [Jain 2008], interpolation procedures for QFPA restricted to conjunctions of integer linear (dis)equalities, and for QFPA restricted to conjunctions of divisibility constraints are given. The combination of both fragments with integer linear inequalities is not supported, however. Our work closes this gap, as it permits predicates involving all types of constraints.

Lynch et al. [Lynch 2008] define an interpolation procedure for linear rational arithmetic, and extend it to integer arithmetic by means of Gomory cuts. For integer arithmetic, however, interpolation in [Lynch 2008] can produce formulas that violate the vocabulary condition (i.e., can contain variables that are not common to ϕ_i and ϕ_j with $i \neq j$), and are therefore not true interpolants. The problem is that Gomory cuts used in [Lynch 2008] do not prevent mixed cuts, for which no efficient interpolation is possible in QFPA.

Brillout et al. [Brillout 2010] define a complete interpolating sequent calculus for QFPA. The calculus contains a rule strengthen that is general enough to simulate arbitrary (possibly mixed) Gomory cuts, but in general causes exponential complexity of interpolant extraction from proofs. In contrast, our cut rule (which is embedded in an effective decision procedure) enables extraction with polynomial complexity.

The recent SMT-solver SmtInterpol decides and interpolates problems in linear integer arithmetic, apparently using an architecture similar to the one in [McMillan 2005]. To the best of our knowledge, the precise design and calculus of SmtInterpol has not been documented in publications yet (see the sequel for an empirical comparison with our approach).

Interpolation for rational arithmetic is a well-explored field. McMillan presents an interpolating theorem prover for linear rational arithmetic and uninterpreted functions [McMillan 2005]; an interpolating SMT-solver for the same logic has been developed by Beyer et al. [Beyer 2008]. Rybalchenko et al. introduced an algorithm for interpolating rational arithmetic with uninterpreted functions without the need for explicit proofs [Rybalchenko 2007].

Experimental Evaluation

We have created a prototypical implementation of our interpolating decision procedure and integrated it as a theory solver into the SMT-solver OpenSMT [Bruttomesso 2010], with the long-term goal of creating an interpolating SMT-solver to be used in model checkers. The prototype was developed on top of

a recent development version of OpenSMT that already provided an interpolation procedure for propositional logic. To the best of our knowledge, the following tools and algorithms are the only ones available for comparison:

- the theorem prover iPrincess [Brillout 2010], which implements an interpolating decision procedure for QFPA based on a sequent calculus,
- the SMT-solver SmtInterpol,[5] a recently released interpolating decision procedure for linear integer arithmetic that uses an architecture similar to the one in Foci [McMillan 2005],
- *quantifier elimination (QE) procedures*, which can be used to generate interpolants; for our experiments, we use the implementation of the Omega test [Pugh 1992b] available in iPrincess.

The benchmarks for our experiments are derived from different families of the SMT-LIB category QF-LIA. Some of the selected families (e.g., *rings*) are specifically designed to test integer reasoning capabilities, and contain problems satisfiable over the rationals. Because SMT-LIB benchmarks are usually conjunctions at the outermost level, we partitioned them into $A \wedge B$ by choosing the first $\frac{k}{10} \cdot n$ of the benchmark conjuncts as A, the rest as B (where n is the total number of conjuncts, and $k \in \{1, \ldots, 9\}$). This yields 9 interpolation problems for each SMT-LIB benchmark.

Our experimental results are summarized in Table 6.1. The first collum provides the name of the problem clases. Each class contains a set of formulas that are either satisfiable or unsatisfiable. The distribution is given in the second collum with two values corresponding to "unsat / sat". Since we consider 9 possible interpolations, the maximum number of computed interpolants is 9 times the number of unsatisfiable problems. In the other colums experimental results performed with different tools are sumerized with 5 values "unsat / sat / average time / #interpolants / average int. size" corresponding to (1) the number of unsatisfiable problems proved to be unsatifiable, (2) the number if satisfiable problems proved to be satisfiable, (3) the average time required for the computation, (4) the number of interpolants produced, and (5) the average length of the computed interpolants. This table shows that our implementation in OpenSMT is competitive with all compared interpolation procedures: in 4 of the 8 families, it is able to prove the largest of problems unsatisfiable (and to compute interpolants for them); in all families but one, the runtime is smaller or comparable with the other tools; in 4 families, the generated interpolants are significantly smaller (on average) than the interpolants computed by the other tools.

In conclusion, we have presented an algorithm computing interpolants in the quantifier-free fragment of Presburger arithmetic in exponential time in the worst case. This algorithm combines the one presented in [Jain 2008] that computes interpolants in polynomial time for systems of equalities over the integers and the

[5] `http://swt.informatik.uni-freiburg.de/research/tools/smtinterpol`

one presented in [McMillan 2005] that computes interpolant in polynomial time for systems of inequalities over the rational numbers, without any overhead.

		OpenSMT	SmtInterpol	iPrincess	Omega QE
Averest	10/9	10/1/31.75/ 90/221	8/4/97.02/ 72/149	0/0/–/ –/–	–/–/203.89/ 8/132639
CIRC-mul	16/1	5/1/48.94/ 45/2357	5/1/24.40/ 45/48827	6/1/130.46/ 35/12764	–/–/108.71/ 125/15392
CIRC-add	17/0	7/0/102.81/ 63/23362	5/0/8.58/ 45/41077	6/0/412.82/ 49/47218	–/–/97.83/ 129/93181
check	4/1	4/1/0.77/ 36/1.7	2/1/0.17/ 18/2.3	4/1/36.65/ 33/485	–/–/0.26/ 30/0.67
nec-smt-small	17/18	1/0/251.95/ 9/36	7/0/259.86/ 63/1728	0/0/–/ –/–	–/–/134.88/ 66/15867
mathsat	100/21	74/15/52.96/ 666/2020	65/13/45.74/ 585/126705	11/11/61.78/ 99/13745	–/–/168.81/ 612/101088
rings	294/0	9/0/59.93/ 81/4611	0/0/–/ –/–	54/0/108.01/ 62/3470	–/–/227.25/ 1474/55307
wisa	2/3	0/0/–/ –/–	1/2/394.22/ 9/1039	0/0/–/ –/–	–/–/67.01/ 14/23709
	unsat/sat	*unsat / sat / average time / #interpolants / average int. size*			

Table 6.1: Results of applying the four compared tools to SMT-LIB benchmarks (times in seconds). Experiments were done on an Intel Xeon X5667 4-core machine with 3.07GHz, heap-space limited to 12GB, running Linux, with a timeout of 900s.

6.4 Conclusion

We presented three approaches for solving the reachability problem for Presburger counter machines. These approaches are incomparable and provide tools for different instances of the problem. They provide rigourous frameworks for computing presentation of inductive invariants in the Presburger arithmetic. A practical implementation of these approaches require some additional efforts and good heuristics. For instance, acceleration techniques require an algorithm for selecting cycles that must be accelerated in order to enforce the termination of the fixpoint computation (such an heuristic algorithm is implemented in Fast). Sometimes the application of widening operators in Kleene fixpoint iterations must be postponed in order to improve the analysis precision. The same problem occurs with the interpolant based model checking. The computation of "good" interpolants is still an important problem. By "good" we mean interpolants that helps the model-checker to compute a prescise enough abstraction of the system.

A possible way for understanding the theoritical limits of these approaches consists in finding classes of systems with termination guaranties. For the acceleration techniques, the class of initialized VASS with a Presburger forward reachability set seems to be such an important class. For the time beeing, we do not know if there exists an initialized VASS in this class that is not flat. We think that such an example does not exist. In fact, we proved in [Leroux 2005b] that all known subclasses of initialized VASS with a forward reachability sets definable in the Presburger arithmetic are flat. However, the problem is still open. Concerning the interpolation based model-checking we are interested in by algorithms computing interpolants that are precise enough for deciding the reachability problem for VASS. Such an algorithm should be a great progress to solve in practice the reachability problem for VASS. In fact, even if this problem is decidable, no algorithm implements a decision procedure for it since the classical solution is difficult from a computational complexity and from an implementation viewpoint (see Chapter 7).

Chapter 7

Vector Addition System Reachability Problem

This chapter follows a paper published in the proceedings of "The Alan Turing Centenary Conference, Manchester, UK, June 22-25, 2012". This paper received a best paper award.

Vector Addition Systems (VAS), Vector Addition Systems with States (VASS), or equivalently Petri Nets are one of the most popular formal methods for the representation and the analysis of parallel processes [Esparza 1994]. Their reachability problem is central since many computational problems (even outside the realm of parallel processes) reduce to the reachability problem. Therefore, improving the computational cost for solving the reachability problem for Petri nets would also improve the complexity of the formal verification of numerous classes of infinite-state systems.

Formally, Vector Addition Systems with States (VASS) with d counters are Presburger counter machines $(Q, \mathbf{x}, \mathbf{x}', T)$ with transitions labeled by $\mathbf{y}' = \mathbf{x} + \mathbf{a}$ where $\mathbf{a} \in \mathbb{Z}^d$. Usually such a formula is simply denoted by $\mathbf{a}$. Let us consider the vector addition system with two states and three counters depicted in Figure 7.1.

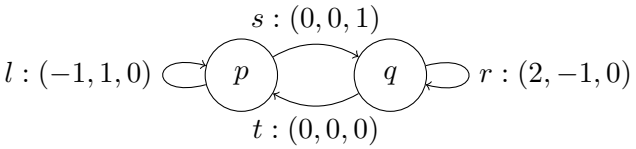


Figure 7.1: A vector addition system with states taken from [Hopcroft 1979].

We have the following run:

$$(p, 1, 0, 0) \xrightarrow{lsrt\ l^2sr^2t\ l^4sr^4t\ \ldots\ l^{2^{n-1}}sr^{2^{n-1}}t} (p, 2^n, 0, n)$$

This run corresponds to the following scheme: (1) iterate the left control loop l as much as possible, then (2) execute the control transition s, (3) iterate the right control loop r as much as possible, (4) execute the control transition t and go back to (1).

Starting from the initial configuration $(p, 1, 0, 0)$, this vector addition system with states exhibits the following reachability set (see [Hopcroft 1979] for a proof)

$$\begin{aligned}\mathrm{Post}^*(\{(p,1,0,0)\}) =& \{p\} \times \{(x_1,x_2,x_3) \in \mathbb{N}^3 \mid 1 \leq x_1 + x_2 \leq 2^{x_3}\} \\ & \cup \{q\} \times \{(x_1,x_2,x_3) \in \mathbb{N}^3 \mid 1 \leq x_1 + 2x_2 \leq 2^{x_3}\}\end{aligned}$$

This equality shows that the reachability set is not definable in the Presburger arithmetic. Hence the Presburger arithmetic is not enough expressive for denoting VASS reachability sets. It is difficult to find out a decidable formalism for denoting reachability sets for VASS. In fact, since the equality of two reachability sets is undecidable[Hack 1976, Jančar 2001] it is not possible to effectively describe reachability sets in a decidable formalism.

Nevertheless, the reachability problem is decidable. Sacerdote and Tenney provided in [Sacerdote 1977] a partial proof of decidability of this problem. The proof was completed in 1981 by Mayr [Mayr 1981b] and simplified by Kosaraju [Kosaraju 1982] from [Sacerdote 1977, Mayr 1981b]. Ten years later [Lambert 1992], Lambert provided a further simplified version based on [Kosaraju 1982]. This last proof still remains difficult and the upper-bound complexity of the corresponding algorithm is at least non-primitive recursive. Nowadays, the exact complexity of the reachability problem for VASS is still an open-problem. Even the existence of an elementary upper-bound complexity is open. In fact, the known general reachability algorithms are exclusively based on the Kosaraju-Lambert-Mayr-Sacerdote-Tenney (KLMST) decomposition.

Recently [Leroux 2009a] we proved thanks to the KLMST decomposition that Parikh images of languages accepted by VASS are semi-pseudo-linear, a class that extends the Presburger sets. An application of this result was provided; we proved that a final configuration is not reachable from an initial one if and only if there exists a forward inductive invariant definable in the Presburger arithmetic that contains the initial configuration but not the final one. Since we can decide if a Presburger formula denotes a forward inductive invariant, we deduce that there exist checkable certificates of non-reachability in the Presburger arithmetic. In particular, there exists a simple algorithm for deciding the general VASS reachability problem based on two semi-algorithms. A first one that tries to prove the reachability by enumerating finite sequences of actions and a second one that tries to prove the non-reachability by enumerating Presburger formulas.

In [Leroux 2011a] we provided a new proof of the reachability problem that is not based on the KLMST decomposition. The proof is based on the *transformer relations* inspired by the production relations introduced by Hauschildt [Hauschildt 1990] and it provides directly that reachability relations are *almost semilinear*, a class of sets inspired by the geometrical decomposition of Presburger definable sets into semilinear sets. Thanks to this characterization we deduce that if a configurations c' is not reachable from a configuration c from the VASS M, there exists a certificate of safety for (c, M, c'), that means a forward inductive invariant definable in the Presburger arithmetic that contains c but not c'.

Example 7.0.1. *Let us show on the VASS depicted in Figure 7.1 how certificate of safety can be obtained. We consider a pair (c, c') of configurations such that c' is not reachable from c. We observe that the following set $C_{c,k}$ where k is the value of the third counter of c' is a Preburger set of configurations that denotes a forward inductive invariant that contains c but not c':*

$$\begin{aligned} C_{c,k} = &\mathrm{Post}^*(\{c\}) \cap (\{p,q\} \times \mathbb{N} \times \mathbb{N} \times \{0,\ldots,k\}) \\ &\cup \{p,q\} \times \mathbb{N} \times \mathbb{N} \times \mathbb{N}_{>k} \end{aligned}$$

In fact $\{p,q\} \times \mathbb{N} \times \mathbb{N} \times \mathbb{N}_{>k}$ is a forward inductive invariant and $\mathrm{Post}^*(\{c\}) \cap (\{p,q\} \times \mathbb{N} \times \mathbb{N} \times \{0,\ldots,k\})$ is a finite set with successors either in this set or in $\{p,q\} \times \mathbb{N} \times \mathbb{N} \times \mathbb{N}_{>k}$ (just observe that the third counter can only be incremented and it is eventually incremented since the control loops s and r can be iterated only a finite number of times from any given configuration).

In the reminder of this chapter we consider *Vector Additions Systems (VAS)* rather than Vector Additions Systems with States (VASS). Informally a VAS is a VASS with a set of control states Q reduced to a single element. The VAS and VASS models are equivalent for the reachability problem since control states can be encoded by additional counters. Moreover from certificates of safety for VAS we deduce certificates of safety for VASS.

Outline of the paper: Section 7.1 recalls the definition of *almost semilinear sets*, a class of sets inspired by the decomposition of Presburger sets into semilinear sets. Section 7.2 introduces definitions related to *vector addition systems*. Section 7.3 introduces a well-order over the runs of vector addition systems. This well-order is central in the proof and it was first introduced by Petr Jančar in another context[Jančar 1990b]. Based on the definition of this well-order we introduce in Section 7.4 the notion of transformer relations and we prove that conic relations generated by transformer relations are definable in $\mathrm{FO}(\mathbb{Q}, +, \leq)$. Thanks to this result and the well-order introduced in the previous section we show in Section 7.5 that reachability sets of vector addition systems are almost semilinear. In Section 7.6 we introduce a dimension function for subsets of integer vectors. In Section 7.7 the almost semilinear sets are proved to be approximable by Presburger sets in a precise way based on the dimension function previously introduced. Thanks to this approximation and since reachability sets are almost semilinear we finally prove in Section 7.8 that the vector addition system reachability problem can be decided by inductive invariants definable in the Presburger arithmetic.

7.1 Almost Semilinear Sets

In this section we introduce the class of *almost semilinear sets*, a class of sets inspired by the geometrical characterization of the *Presburger sets* by *semilinear sets*.

A *periodic set* is a subset $\mathbf{P} \subseteq \mathbb{Z}^d$ such that $\mathbf{0} \in \mathbf{P}$ and $\mathbf{P} + \mathbf{P} \subseteq \mathbf{P}$. A *conic set* is a subset $\mathbf{C} \subseteq \mathbb{Q}^d$ such that $\mathbf{0} \in \mathbf{C}$, $\mathbf{C} + \mathbf{C} \subseteq \mathbf{C}$ and $\mathbb{Q}_{\geq 0}\mathbf{C} \subseteq \mathbf{C}$. A periodic

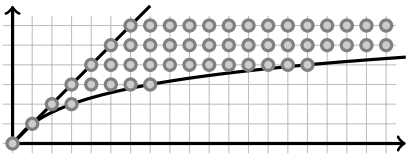


Figure 7.2: Periodic set introduced in Example 7.1.1

set $\mathbf{P}$ is said to be *finitely generated* if there exist vectors $\mathbf{p}_1, \ldots, \mathbf{p}_k \in \mathbf{P}$ such that $\mathbf{P} = \mathbb{N}\mathbf{p}_1 + \cdots + \mathbb{N}\mathbf{p}_k$. A periodic set $\mathbf{P}$ is said to be *asymptotically definable* if the conic set $\mathbb{Q}_{\geq 0}\mathbf{P}$ is definable in $\mathrm{FO}(\mathbb{Q}, +, \leq)$. Observe that finitely generated periodic sets are asymptotically definable since the conic set $\mathbb{Q}_{\geq 0}\mathbf{P}$ generated by $\mathbf{P} = \mathbb{N}\mathbf{p}_1 + \cdots + \mathbb{N}\mathbf{p}_k$ is equal to $\mathbb{Q}_{\geq 0}\mathbf{p}_1 + \cdots + \mathbb{Q}_{\geq 0}\mathbf{p}_k$.

Example 7.1.1. *The periodic set* $\mathbf{P} = \{\mathbf{p} \in \mathbb{N}^2 \mid \mathbf{p}(2) \leq \mathbf{p}(1) \leq 2^{\mathbf{p}(2)} - 1\}$ *is depicted in Figure 7.2. Observe that* $\mathbb{Q}_{\geq 0}\mathbf{P}$ *is the conic set* $\{\mathbf{0}\} \cup \{\mathbf{c} \in \mathbb{Q}^2_{>0} \mid \mathbf{c}(2) \leq \mathbf{c}(1)\}$ *which is definable in* $\mathrm{FO}(\mathbb{Q}, +, \leq)$.

A *Presburger set* is a set $\mathbf{Z} \subseteq \mathbb{Z}^d$ definable in $\mathrm{FO}(\mathbb{Z}, +, \leq)$. Recall that $\mathbf{Z} \subseteq \mathbb{Z}^d$ is a Presburger set iff it is *semilinear*, i.e. a finite union of *linear sets* $\mathbf{b} + \mathbf{P}$ where $\mathbf{b} \in \mathbb{Z}^d$ and $\mathbf{P} \subseteq \mathbb{Z}^d$ is a finitely generated periodic set [Ginsburg 1966]. The class of *almost semilinear sets* [Leroux 2011a] is obtained from the definition of semilinear sets by weakening the finiteness condition on the considered periodic sets. More formally, an *almost semilinear set* is a finite union of sets of the form $\mathbf{b} + \mathbf{P}$ where $\mathbf{b} \in \mathbb{Z}^d$ and $\mathbf{P} \subseteq \mathbb{Z}^d$ is an asymptotically definable periodic set.

7.2 Vector Addition Systems

A *Vector Addition System (VAS)* is given by a finite subset $\mathbf{A} \subseteq \mathbb{Z}^d$. A vector $\mathbf{a} \in \mathbf{A}$ is called an *action*. A *configuration* is a vector $\mathbf{c} \in \mathbb{N}^d$. A *run* ρ is a non-empty word $\rho = \mathbf{c}_0 \ldots \mathbf{c}_k$ of configurations such that the difference $\mathbf{a}_j = \mathbf{c}_j - \mathbf{c}_{j-1}$ is in $\mathbf{A}$ for every $j \in \{1, \ldots, k\}$. In that case we say that ρ is *labeled* by $w = \mathbf{a}_1 \ldots \mathbf{a}_k$, the configurations $\mathbf{c}_0$ and $\mathbf{c}_k$ are respectively called the *source* and the *target* and they are denoted by $\mathrm{src}(\rho)$ and $\mathrm{tgt}(\rho)$. The *direction* of ρ is the pair $(\mathrm{src}(\rho), \mathrm{tgt}(\rho))$, denoted by $\mathrm{dir}(\rho)$. Given a word $w \in \mathbf{A}^*$, we introduce the binary relation $\xrightarrow{w}$ over the set of configurations by $\mathbf{x} \xrightarrow{w} \mathbf{y}$ if there exists a run ρ from $\mathbf{x}$ to $\mathbf{y}$ labeled by w. Observe that in this case ρ is unique. The *displacement* of a word $w = \mathbf{a}_1 \ldots \mathbf{a}_k$ of actions $\mathbf{a}_j \in \mathbf{A}$ is the vector $\Delta(w) = \sum_{j=1}^{k} \mathbf{a}_j$. Note that $\mathbf{x} \xrightarrow{w} \mathbf{y}$ implies $\mathbf{x} + \Delta(w) = \mathbf{y}$ but the converse is not true in general. The *reachability relation* is the relation $\xrightarrow{*}$ over $\mathbb{N}^d$ defined by $\mathbf{x} \xrightarrow{*} \mathbf{y}$ if there exists a run from $\mathbf{x}$ to $\mathbf{y}$. The following simple lemma is central in this paper.

Lemma 7.2.1 (Monotony)**.** *We have* $\mathbf{c} + \mathbf{x} \xrightarrow{w} \mathbf{c} + \mathbf{y}$ *for every* $\mathbf{x} \xrightarrow{w} \mathbf{y}$ *and for every* $\mathbf{c} \in \mathbb{N}^d$.

Proof. Just observe that if $\rho = \mathbf{c}_1 \dots \mathbf{c}_k$ is a run from $\mathbf{x}$ to $\mathbf{y}$ labeled by w where $\mathbf{c}_j \in \mathbb{N}^d$ then $\rho' = \mathbf{c}'_1 \dots \mathbf{c}'_k$ where $\mathbf{c}'_j = \mathbf{c} + \mathbf{c}_j$ is a run from $\mathbf{c} + \mathbf{x}$ to $\mathbf{c} + \mathbf{y}$ labeled by w. □

The set of configurations *forward reachable* from a configuration $\mathbf{x} \in \mathbb{N}^d$ is the set $\{\mathbf{c} \in \mathbb{N}^d \mid \mathbf{x} \xrightarrow{*} \mathbf{c}\}$ denoted by $\mathrm{Post}^*(\mathbf{x})$. Symmetrically the set of configurations *backward reachable* from a configuration $\mathbf{y} \in \mathbb{N}^d$ is the set $\{\mathbf{c} \in \mathbb{N}^d \mid \mathbf{c} \xrightarrow{*} \mathbf{y}\}$ denoted by $\mathrm{Pre}^*(\mathbf{y})$. These definitions are extended over sets of configurations $\mathbf{X}, \mathbf{Y} \subseteq \mathbb{N}^d$ by $\mathrm{Post}^*(\mathbf{X}) = \bigcup_{\mathbf{x} \in \mathbf{X}} \mathrm{Post}^*(\mathbf{x})$ and $\mathrm{Pre}^*(\mathbf{Y}) = \bigcup_{\mathbf{y} \in \mathbf{Y}} \mathrm{Post}^*(\mathbf{y})$. A set $\mathbf{X} \subseteq \mathbb{N}^d$ is said to be a *forward inductive invariant* if $\mathbf{X} = \mathrm{Post}^*(\mathbf{X})$. Symmetrically a set $\mathbf{Y} \subseteq \mathbb{N}^d$ is said to be a *backward inductive invariant* if $\mathbf{Y} = \mathrm{Pre}^*(\mathbf{Y})$.

In this chapter we prove that for every $\mathbf{x}, \mathbf{y} \in \mathbb{N}^d$ such that there does not exist a run from $\mathbf{x}$ to $\mathbf{y}$, then there exists a pair $(\mathbf{X}, \mathbf{Y})$ of disjoint Presburger sets $\mathbf{X}, \mathbf{Y} \subseteq \mathbb{N}^d$ such that $\mathbf{X}$ is a forward inductive invariant that contains $\mathbf{x}$ and $\mathbf{Y}$ is a backward inductive invariant that contains $\mathbf{y}$. This result will provide directly the following theorem.

Theorem 7.2.2. *The reachability problem for vector addition systems is decidable.*

Proof. See Chapter 5 for more details. □

Remark 7.2.3. *The set* $\mathrm{Post}^*(\mathbf{x})$ *is a forward inductive invariant that contains* $\mathbf{x}$ *and* $\mathrm{Pre}^*(\mathbf{y})$ *is a backward inductive invariant that contains* $\mathbf{y}$*. Moreover, if there does not exist a run from* $\mathbf{x}$ *to* $\mathbf{y}$ *then these two reachability sets are disjoint. However in general reachability sets are not definable in the Presburger arithmetic [Hopcroft 1979].*

7.3 Well-Order Over The Runs

An order $\sqsubseteq$ over a set S is said to be a *well-order* if for every sequence $(s_j)_{j \in \mathbb{N}}$ of elements $s_j \in S$ there exist $j < k$ such that $s_j \sqsubseteq s_k$. Observe that $(\mathbb{N}, \leq)$ is a well-ordered set whereas $(\mathbb{Z}, \leq)$ is not well-ordered. As another example, the pigeon-hole principle shows that a set S is well-ordered by the equality relation if and only if S is finite. Well-orders can be easily defined thanks to *Dickson's lemma* and *Higman's lemma* as follows.

Dickson's lemma: Dickson's lemma shows that the Cartesian product of two well-ordered sets is well-ordered. More formally, given two ordered sets $(S_1, \sqsubseteq_1)$ and $(S_2, \sqsubseteq_2)$ we denote by $\sqsubseteq_1 \times \sqsubseteq_2$ the order defined component-wise over the Cartesian product $S_1 \times S_2$ by $(s_1, s_2) \sqsubseteq_1 \times \sqsubseteq_2 (s'_1, s'_2)$ if $s_1 \sqsubseteq_1 s'_1$ and $s_2 \sqsubseteq_2 s'_2$. *Dickson's lemma* says that $(S_1 \times S_2, \sqsubseteq_1 \times \sqsubseteq_2)$ is well-ordered for every well-ordered sets $(S_1, \sqsubseteq_1)$ and $(S_2, \sqsubseteq_2)$. As a direct application, the set $\mathbb{N}^d$ equipped with the component-wise extension of $\leq$ is well-ordered.

Higman's lemma: Higman's lemma shows that words over well-ordered alphabets can be well-ordered. More formally, given an ordered set $(S, \sqsubseteq)$, we introduce

the set S^* of words over S equipped with the order $\sqsubseteq^*$ defined by $w \sqsubseteq^* w'$ if w and w' can be decomposed into $w = s_1 \dots s_k$ and $w' \in S^* s'_1 S^* \dots s'_k S^*$ where $s_j \sqsubseteq s'_j$ are in S for every $j \in \{1, \dots, k\}$. *Higman's lemma* says that $(S^*, \sqsubseteq^*)$ is well-ordered for every well-ordered set $(S, \sqsubseteq)$. As a classical application, the set of words over a finite alphabet S is well-ordered by the sub-word relation $=^*$.

We define a well-order over the runs as follows. We introduce the relation $\trianglelefteq$ over the runs defined by $\rho \trianglelefteq \rho'$ if ρ is a run of the form $\rho = \mathbf{c}_0 \dots \mathbf{c}_k$ where $\mathbf{c}_j \in \mathbb{N}^d$ and if there exists a sequence $(\mathbf{v}_j)_{0 \leq j \leq k+1}$ of vectors $\mathbf{v}_j \in \mathbb{N}^d$ such that ρ' is a run of the form $\rho' = \rho_0 \dots \rho_k$ where ρ_j is a run from $\mathbf{c}_j + \mathbf{v}_j$ to $\mathbf{c}_j + \mathbf{v}_{j+1}$.

Lemma 7.3.1. *The relation $\trianglelefteq$ is a well-order over the runs.*

Proof. A proof of this lemma with different notations can be obtained from Section 6 of [Jančar 1990b] with a simple reduction. For sake of completeness, we prefer to give a direct proof of this important result. To do so, we introduce a well-order $\preceq$ over the runs based on Dickson's lemma and Higman's lemma and we show that $\preceq$ and $\trianglelefteq$ are equal. We first associate to a run $\rho = \mathbf{c}_0 \dots \mathbf{c}_k$ the word $\alpha(\rho) = (\mathbf{a}_1, \mathbf{c}_1) \dots (\mathbf{a}_k, \mathbf{c}_k)$ over the set $S = \mathbf{A} \times \mathbb{N}^d$ where $\mathbf{a}_j = \mathbf{c}_j - \mathbf{c}_{j-1}$. The set S is well-ordered by the relation $\sqsubseteq$ defined by $(\mathbf{a}_1, \mathbf{c}_1) \sqsubseteq (\mathbf{a}_2, \mathbf{c}_2)$ if $\mathbf{a}_1 = \mathbf{a}_2$ and $\mathbf{c}_1 \leq \mathbf{c}_2$. Dickson's lemma shows that $\sqsubseteq$ is a well-order. The set of words S^* is well-ordered thanks to Higman's lemma by the relation $\sqsubseteq^*$. The well-order $\preceq$ over the runs is defined by $\rho \preceq \rho'$ if $\mathrm{dir}(\rho) \leq \mathrm{dir}(\rho')$ and $\alpha(\rho) \sqsubseteq^* \alpha(\rho')$. Now, let us prove that $\preceq$ and $\trianglelefteq$ are equal. We consider a run $\rho = \mathbf{c}_0 \dots \mathbf{c}_k$ with $\mathbf{c}_j \in \mathbb{N}^d$ and we introduce the action $\mathbf{a}_j = \mathbf{c}_j - \mathbf{c}_{j-1}$ for each $j \in \{1, \dots, k\}$.

Assume first that $\rho \preceq \rho'$ for some run ρ'. Since $\alpha(\rho) = (\mathbf{a}_1, \mathbf{c}_1) \dots (\mathbf{a}_k, \mathbf{c}_k)$ and $\alpha(\rho) \sqsubseteq^* \alpha(\rho')$ we deduce a decomposition of $\alpha(\rho')$ into the following word where $\mathbf{c}'_j \geq \mathbf{c}_j$ for every $j \in \{1, \dots, k\}$ and $w_0, \dots, w_k \in S^*$:

$$\alpha(\rho') = w_0(\mathbf{a}_1, \mathbf{c}'_1)w_1 \dots (\mathbf{a}_k, \mathbf{c}'_k)w_k$$

In particular ρ' can be decomposed in $\rho' = \rho_0 \dots \rho_k$ where ρ_0 is a run from $\mathrm{src}(\rho')$ to $\mathbf{c}'_1 - \mathbf{a}_1$, ρ_j is a run from $\mathbf{c}'_j$ to $\mathbf{c}'_{j+1} - \mathbf{a}_{j+1}$ for every $j \in \{1, \dots, k-1\}$, and ρ_k is a run from $\mathbf{c}'_k$ to $\mathrm{tgt}(\rho')$. Let us introduce the sequence $(\mathbf{v}_j)_{0 \leq j \leq k+1}$ of vectors defined by $\mathbf{v}_0 = \mathrm{src}(\rho') - \mathrm{src}(\rho)$, $\mathbf{v}_j = \mathbf{c}'_j - \mathbf{c}_j$ for every $j \in \{1, \dots, k\}$ and $\mathbf{v}_{k+1} = \mathrm{tgt}(\rho') - \mathrm{tgt}(\rho)$. Note that $\mathbf{v}_j \in \mathbb{N}^d$ for every $j \in \{0, \dots, k+1\}$. Observe that for every $j \in \{1, \dots, k-1\}$ we have $\mathbf{c}'_{j+1} - \mathbf{a}_j = \mathbf{c}_{j+1} - \mathbf{a}_j + \mathbf{v}_{j+1} = \mathbf{c}_j + \mathbf{v}_{j+1}$. Hence ρ_j is a run from $\mathbf{c}_j + \mathbf{v}_j$ to $\mathbf{c}_j + \mathbf{v}_{j+1}$ for every $j \in \{0, \dots, k\}$. Therefore $\rho \trianglelefteq \rho'$.

Conversely, let us assume that $\rho \trianglelefteq \rho'$ for some run ρ'. We introduce a sequence $(\mathbf{v}_j)_{0 \leq j \leq k+1}$ of vectors in $\mathbb{N}^d$ such that $\rho' = \rho_0 \dots \rho_k$ where ρ_j is a run from $\mathbf{c}_j + \mathbf{v}_j$ to $\mathbf{c}_j + \mathbf{v}_{j+1}$. We deduce the following equality where $\mathbf{a}'_j = \mathrm{src}(\rho_j) - \mathrm{tgt}(\rho_{j-1})$:

$$\alpha(\rho') = \alpha(\rho_0)(\mathbf{a}'_1, \mathbf{c}_1 + \mathbf{v}_1)\alpha(\rho_1) \dots (\mathbf{a}'_k, \mathbf{c}_k + \mathbf{v}_k)\alpha(\rho_k)$$

Observe that $\mathbf{a}'_j = (\mathbf{c}_j + \mathbf{v}_j) - (\mathbf{c}_{j-1} + \mathbf{v}_j) = \mathbf{a}_j$. We deduce that $\alpha(\rho) \sqsubseteq^* \alpha(\rho')$. Moreover, since $\mathrm{dir}(\rho) \leq \mathrm{dir}(\rho')$ we get $\rho \preceq \rho'$. □

Example 7.3.2. *The well-order $\trianglelefteq$ provides a simple criterion for deciding if a component i of a VAS is* unbounded *from an initial configuration $\mathbf{c}$, i.e. for every $n \in \mathbb{N}$ there exists a run from $\mathbf{c}$ to a configurations $\mathbf{y}_n$ such that $\mathbf{y}_n(i) \geq n$. In fact, we directly show that a component i is unbounded from $\mathbf{c}$ if and only if there exists a sequence $(\mathbf{a}_j)_{1\leq j\leq k}$ of actions $\mathbf{a}_j \in \mathbf{A}$ and a sequence $(w_j)_{0\leq j\leq k}$ of words $w_j \in \mathbf{A}^*$ such that $\mathbf{v}_j = \Delta(w_0) + \cdots + \Delta(w_j)$ is in $\mathbb{N}^d$ for every $j \in \{0, \ldots, k\}$, $\mathbf{v}_k(i) > 0$ and such that there exists a run from $\mathbf{c}$ labeled by $w_0\mathbf{a}_1w_1 \ldots \mathbf{a}_kw_k$.*

7.4 Transformer Relations

Based on the definition of $\trianglelefteq$, we introduce the *transformer relation with capacity* $\mathbf{c} \in \mathbb{N}^d$ as the binary relation $\overset{\mathbf{c}}{\curvearrowright}$ over $\mathbb{N}^d$ defined by $\mathbf{x} \overset{\mathbf{c}}{\curvearrowright} \mathbf{y}$ if there exists a run from $\mathbf{c} + \mathbf{x}$ to $\mathbf{c} + \mathbf{y}$. We also associate to every run $\rho = \mathbf{c}_0 \ldots \mathbf{c}_k$ with $\mathbf{c}_j \in \mathbb{N}^d$ the *transformer relation along the run* ρ denoted by $\overset{\rho}{\curvearrowright}$ and defined as the following composition:

$$\overset{\rho}{\curvearrowright} \quad = \quad \overset{\mathbf{c}_0}{\curvearrowright} \circ \cdots \circ \overset{\mathbf{c}_k}{\curvearrowright}$$

In this section transformer relations are shown to be *asymptotically definable periodic*. Thanks to the following Lemma 7.4.1, it is sufficient to prove that $\overset{\mathbf{c}}{\curvearrowright}$ is in this class for every capacity $\mathbf{c} \in \mathbb{N}^d$.

Lemma 7.4.1. *Asymptotically definable periodic relations are stable by composition.*

Proof. Assume that $R, S \subseteq \mathbb{Z}^d \times \mathbb{Z}^d$ are two periodic relations and observe that $(\mathbf{0}, \mathbf{0}) \in R \circ S$. Let us consider two pairs $(\mathbf{x}_1, \mathbf{z}_1)$ and $(\mathbf{x}_2, \mathbf{z}_2)$ in $R \circ S$. For each $k \in \{1, 2\}$, there exists $\mathbf{y}_k \in \mathbb{Z}^d$ such that $(\mathbf{x}_k, \mathbf{y}_k) \in R$ and $(\mathbf{y}_k, \mathbf{z}_k) \in S$. As R and S are periodic we get $(\mathbf{x}, \mathbf{y}) \in R$ and $(\mathbf{y}, \mathbf{z}) \in S$ where $\mathbf{x} = \mathbf{x}_1 + \mathbf{x}_2$, $\mathbf{y} = \mathbf{y}_1 + \mathbf{y}_2$ and $\mathbf{z} = \mathbf{z}_1 + \mathbf{z}_2$. Thus $(\mathbf{x}, \mathbf{z}) \in R \circ S$ and we have proved that $R \circ S$ is periodic. Now just observe that $\mathbb{Q}_{\geq 0}(R \circ S) = (\mathbb{Q}_{\geq 0}R) \circ (\mathbb{Q}_{\geq 0}S)$. Hence if R and S are asymptotically definable then $R \circ S$ is also asymptotically definable. □

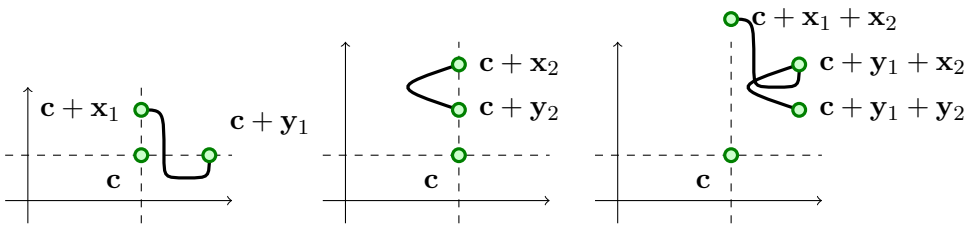


Figure 7.3: Transformer relations are periodic.

Lemma 7.4.2. *The transformer relation $\overset{\mathbf{c}}{\curvearrowright}$ is periodic.*

Proof. Assume that $\mathbf{c} + \mathbf{x}_1 \xrightarrow{w_1} \mathbf{c} + \mathbf{y}_1$ and $\mathbf{c} + \mathbf{x}_2 \xrightarrow{w_2} \mathbf{c} + \mathbf{y}_2$ for words $w_1, w_2 \in \mathbf{A}^*$ and vectors $\mathbf{x}_1, \mathbf{y}_1, \mathbf{x}_2, \mathbf{y}_2 \in \mathbb{N}^d$. By monotony $\mathbf{c} + \mathbf{x}_1 + \mathbf{x}_2 \xrightarrow{w_1w_2} \mathbf{c} + \mathbf{y}_1 + \mathbf{y}_2$ (see Figure 7.3). □

In the remainder of this section, we show that $\mathbb{Q}_{\geq 0} \overset{\mathbf{c}}{\curvearrowright}$ is definable in $\mathrm{FO}(\mathbb{Q}, +, \leq)$. We introduce the set $\Gamma_{\mathbf{c}}$ of triples $\gamma = (\mathbf{x}, \mathbf{c}, \mathbf{y})$ such that $\mathbf{x} \overset{\mathbf{c}}{\curvearrowright} \mathbf{y}$ and the set $\Gamma = \bigcup_{\mathbf{c} \in \mathbb{N}^d} \Gamma_{\mathbf{c}}$. Given a triple $\gamma \in \Gamma$, the vectors $\mathbf{x}, \mathbf{c}, \mathbf{y}$ implicitly denote the components of γ. We introduce the set Ω_γ of runs ρ such that $\mathrm{dir}(\rho) \in (\mathbf{c}, \mathbf{c}) + \mathbb{N}(\mathbf{x}, \mathbf{y})$ and the set $\mathbf{Q}_\gamma$ of configurations $\mathbf{q} \in \mathbb{N}^d$ such that there exists a run $\rho \in \Omega_\gamma$ in which $\mathbf{q}$ occurs. We denote by I_γ the set of indexes $i \in \{1, \ldots, d\}$ such that $\{\mathbf{q}(i) \mid \mathbf{q} \in \mathbf{Q}_\gamma\}$ is finite.

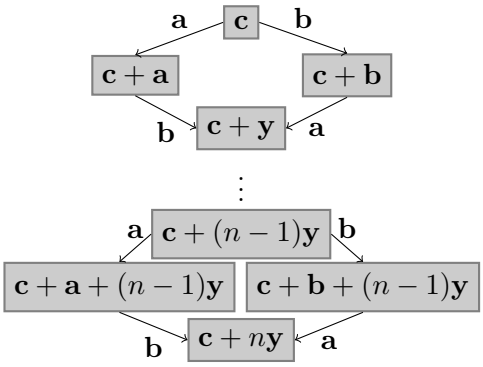


Figure 7.4: Figure for Example 7.4.3

Example 7.4.3. *Let us consider the VAS* $\mathbf{A} = \{\mathbf{a}, \mathbf{b}\}$ *where* $\mathbf{a} = (1, 1, -1)$ *and* $\mathbf{b} = (-1, 0, 1)$ *and let* $\gamma = (\mathbf{x}, \mathbf{c}, \mathbf{y})$ *where* $\mathbf{x} = (0, 0, 0)$, $\mathbf{c} = (1, 0, 1)$ *and* $\mathbf{y} = (0, 1, 0)$. *Since* $\mathbf{x} = (0, 0, 0)$, *we observe that* $\Omega_\gamma = \{\mathbf{c} \xrightarrow{w_1 \ldots w_n} \mathbf{c} + n\mathbf{y} \mid n \in \mathbb{N}\ w_j \in \{\mathbf{ab}, \mathbf{ba}\}\}$. *This set of runs is depicted in Figure 7.4. Observe that* $\mathbf{Q}_\gamma = (\mathbf{c} + \mathbf{a} + \mathbb{N}\mathbf{y}) \cup (\mathbf{c} + \mathbb{N}\mathbf{y}) \cup (\mathbf{c} + \mathbf{b} + \mathbb{N}\mathbf{y})$. *Hence the set of bounded components is* $I_\gamma = \{1, 3\}$.

In section 7.4.1 we show that for every configuration $\mathbf{q} \in \mathbf{Q}_\gamma$, there exist configurations $\mathbf{q}' \in \mathbf{Q}_\gamma$ that coincide with $\mathbf{q}$ on components indexed by I_γ and such that $\mathbf{q}'$ is as large as expected on all the other components. Based on a projection of the unbounded components of vectors in $\mathbf{Q}_\gamma$, i.e. the components not indexed by I_γ, we show in Section 7.4.3 that a finite graph G_γ called *production graph* can be canonically associated to every triple γ. We also prove that the class $\{G_\gamma \mid \gamma \in \Gamma_{\mathbf{c}}\}$ is finite. Finally in Section 7.4.2 we introduce a binary relation $R_\gamma \subseteq \mathbb{Q}_{\geq 0} \overset{\mathbf{c}}{\curvearrowright}$ definable in $\mathrm{FO}(\mathbb{Q}, +, \leq)$ associated to the production graphs G_γ and such that $(\mathbf{x}, \mathbf{y}) \in R_\gamma$. By observing that $\mathbb{Q}_{\geq 0} \overset{\mathbf{c}}{\curvearrowright} = \bigcup_{\gamma \in \Gamma_{\mathbf{c}}} R_\gamma$ and the class $\{R_\gamma \mid \gamma \in \Gamma_{\mathbf{c}}\}$ is finite we deduce that the periodic relation $\overset{\mathbf{c}}{\curvearrowright}$ is asymptotically definable.

7.4.1 Intraproductions

An *intraproduction* for γ is a vector $\mathbf{h} \in \mathbb{N}^d$ such that there exists $n \in \mathbb{N}$ satisfying $n\mathbf{x} \overset{\mathbf{c}}{\curvearrowright} \mathbf{h} \overset{\mathbf{c}}{\curvearrowright} n\mathbf{y}$. We denote by $\mathbf{H}_\gamma$ the set of intraproductions for γ. This set is periodic since $\overset{\mathbf{c}}{\curvearrowright}$ is periodic. In particular for every $\mathbf{h} \in \mathbf{H}_\gamma$ we have $\mathbb{N}\mathbf{h} \subseteq \mathbf{H}_\gamma$ and the

following lemma shows that $\mathbf{Q}_\gamma + \mathbb{N}\mathbf{h} \subseteq \mathbf{Q}_\gamma$. Hence, the components of every vector $\mathbf{q} \in \mathbf{Q}_\gamma$ indexed by i such that $\mathbf{h}(i) > 0$ can be increased to arbitrary large values by adding a large number of times the vector $\mathbf{h}$. In order to increase simultaneously all the components not indexed by I_γ we are interested by intraproductions $\mathbf{h}$ such that $\mathbf{h}(i) > 0$ for every $i \notin I_\gamma$. Note that components indexed by I_γ are necessarily zero since $\mathbf{c} + \mathbb{N}\mathbf{h} \subseteq \mathbf{Q}_\gamma$ for every intraproduction $\mathbf{h}$.

Example 7.4.4. *Let us come back to Example 7.4.3. We have* $\mathbf{H}_\gamma = \mathbb{N}\mathbf{y}$.

Lemma 7.4.5. *We have* $\mathbf{Q}_\gamma + \mathbf{H}_\gamma \subseteq \mathbf{Q}_\gamma$.

Proof. Let $\mathbf{q} \in \mathbf{Q}_\gamma$ and $\mathbf{h} \in \mathbf{H}_\gamma$. As $\mathbf{q} \in \mathbf{Q}_\gamma$, there exist $n \in \mathbb{N}$ and words $u, v \in \mathbf{A}^*$ such that $\mathbf{c} + n\mathbf{x} \xrightarrow{u} \mathbf{q} \xrightarrow{v} \mathbf{c} + n\mathbf{y}$. Since $\mathbf{h} \in \mathbf{H}_\gamma$ there exist $n' \in \mathbb{N}$ and words $u', v' \in \mathbf{A}^*$ such that $\mathbf{c} + n'\mathbf{x} \xrightarrow{u'} \mathbf{c} + \mathbf{h} \xrightarrow{v'} \mathbf{c} + n'\mathbf{y}$. Let $m = n + n'$. By monotony, we have $\mathbf{c} + m\mathbf{x} \xrightarrow{u'u} \mathbf{q} + \mathbf{h} \xrightarrow{vv'} \mathbf{c} + m\mathbf{y}$. Hence $\mathbf{q} + \mathbf{h} \in \mathbf{Q}_\gamma$. □

Lemma 7.4.6. *For every* $\mathbf{q} \leq \mathbf{q}'$ *in* $\mathbf{Q}_\gamma$ *there exists* $\mathbf{h} \in \mathbf{H}_\gamma$ *such that* $\mathbf{q}' \leq \mathbf{q} + \mathbf{h}$.

Proof. As $\mathbf{q}, \mathbf{q}' \in \mathbf{Q}_\gamma$ there exists $m, m' \in \mathbb{N}$ and $u, v, u', v' \in \mathbf{A}^*$ such that:

$$\mathbf{c} + m\mathbf{x} \xrightarrow{u} \mathbf{q} \xrightarrow{v} \mathbf{c} + m\mathbf{y} \qquad \text{and} \qquad \mathbf{c} + m'\mathbf{x} \xrightarrow{u'} \mathbf{q}' \xrightarrow{v'} \mathbf{c} + m'\mathbf{y}$$

Let us introduce $\mathbf{v} = \mathbf{q}' - \mathbf{q}$, $\mathbf{h} = \mathbf{v} + m(\mathbf{x} + \mathbf{y})$, and $n = m + m'$. By monotony:

$$\begin{array}{rcl} \mathbf{c} + n\mathbf{x} & \xrightarrow{u'} & \mathbf{q}' + m\mathbf{x} \qquad \text{and} \quad \mathbf{q} + \mathbf{v} + m\mathbf{x} \xrightarrow{v} \mathbf{c} + \mathbf{h} \\ \mathbf{c} + \mathbf{h} & \xrightarrow{u} & \mathbf{q} + \mathbf{v} + m\mathbf{y} \qquad \text{and} \quad \mathbf{q}' + m\mathbf{y} \xrightarrow{v'} \mathbf{c} + n\mathbf{y} \end{array}$$

Since $\mathbf{q}' + m\mathbf{x} = \mathbf{q} + \mathbf{v} + m\mathbf{x}$ and $\mathbf{q} + \mathbf{v} + m\mathbf{y} = \mathbf{q}' + m\mathbf{y}$, we have proved that $\mathbf{c}+n\mathbf{x} \xrightarrow{u'v} \mathbf{c}+\mathbf{h} \xrightarrow{uv'} \mathbf{c}+n\mathbf{y}$. Hence $\mathbf{h} \in \mathbf{H}_\gamma$. Observe that $\mathbf{q}+\mathbf{h} = \mathbf{q}'+m(\mathbf{x}+\mathbf{y}) \geq \mathbf{q}'$. We are done. □

Lemma 7.4.7. *There exist* $\mathbf{h} \in \mathbf{H}_\gamma$ *such that* $I_\gamma = \{i \mid \mathbf{h}(i) = 0\}$.

Proof. Let $i \notin I_\gamma$. There exists a sequence $(\mathbf{q}_j)_{j\in\mathbb{N}}$ of configurations $\mathbf{q}_j \in \mathbf{Q}_\gamma$ such that $(\mathbf{q}_j(i))_{j\in\mathbb{N}}$ is strictly increasing. Since $(\mathbb{N}^d, \leq)$ is well-ordered there exists $j < k$ such that $\mathbf{q}_j \leq \mathbf{q}_k$. Lemma 7.4.6 shows that there exists an intraproduction $\mathbf{h}_i$ for γ such that $\mathbf{q}_k \leq \mathbf{q}_j + \mathbf{h}_i$. In particular $\mathbf{h}_i(i) > 0$ since $\mathbf{q}_j(i) < \mathbf{q}_k(i)$. As the set of intraproductions $\mathbf{H}_\gamma$ is periodic we deduce that $\mathbf{h} = \sum_{i\notin I} \mathbf{h}_i$ is an intraproduction for γ. By construction we have $\mathbf{h}(i) > 0$ for every $i \notin I_\gamma$. Since $\mathbf{h} \in \mathbf{H}_\gamma$ we deduce that $\mathbf{h}(i) = 0$ for every $i \in I_\gamma$. Therefore $I_\gamma = \{i \mid \mathbf{h}(i) = 0\}$. □

7.4.2 Production Graphs

Finite graphs G_γ, called *production graphs* can be associated to every triple γ as follows. The set of *states* is obtained from $\mathbf{Q}_\gamma$ by projecting away the unbounded components. More formally, we introduce the projection function $\pi_\gamma : \mathbf{Q}_\gamma \to \mathbb{N}^{I_\gamma}$ defined by $\pi_\gamma(\mathbf{q})(i) = \mathbf{q}(i)$ for every $\mathbf{q} \in \mathbf{Q}_\gamma$ and for every $i \in I_\gamma$. We consider the

finite set of *states* $S_\gamma = \pi_\gamma(\mathbf{Q}_\gamma)$ and the set T_γ of *transitions* $(\pi_\gamma(\mathbf{q}), \mathbf{q}' - \mathbf{q}, \pi_\gamma(\mathbf{q}'))$ where $\mathbf{q}\mathbf{q}'$ is a factor of a run in Ω_γ. Since $T_\gamma \subseteq S_\gamma \times \mathbf{A} \times S_\gamma$ we deduce that T_γ is finite. We introduce the finite graph $G_\gamma = (S_\gamma, T_\gamma)$, called the *production graph* of γ. Since $\mathbf{c} \in \mathbf{Q}_\gamma$ we deduce that $\pi_\gamma(\mathbf{c})$ is a state of G_γ. This state, called the *special state* for γ, is denoted by s_γ.

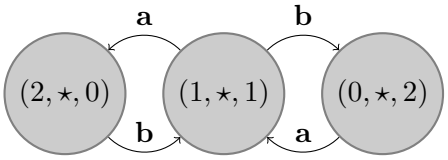


Figure 7.5: Figure for Example 7.4.8

Example 7.4.8. *Let us come back to Example 7.4.3. Observe that* $\pi_\gamma(\mathbf{c} + \mathbf{a} + n\mathbf{y}) = (2, \star, 0)$, $\pi_\gamma(\mathbf{c} + n\mathbf{y}) = (1, \star, 1)$, *and* $\pi_\gamma(\mathbf{c} + \mathbf{b} + n\mathbf{y}) = (0, \star, 2)$ *where* $\star$ *denotes a projected component. The graph* G_γ *is depicted in Figure 7.5. Note that* $s_\gamma = (1, \star, 1)$.

Corollary 7.4.9. *We have* $\pi_\gamma(\mathrm{src}(\rho)) = s_\gamma = \pi_\gamma(\mathrm{tgt}(\rho))$ *for every run* $\rho \in \Omega_\gamma$.

Proof. Since $\rho \in \Omega_\gamma$ there exists $n \in \mathbb{N}$ such that ρ is a run from $\mathbf{c} + n\mathbf{x}$ to $\mathbf{c} + n\mathbf{y}$. In particular $n\mathbf{x}$ and $n\mathbf{y}$ are two intraproductions for γ. We get $n\mathbf{x}(i) = 0 = n\mathbf{y}(i)$ for every $i \in I_\gamma$. Hence $\pi_\gamma(\mathrm{src}(\rho)) = \pi_\gamma(\mathbf{c}) = \pi_\gamma(\mathrm{tgt}(\rho))$. □

A *path* in G_γ is a word $p = (s_0, \mathbf{a}_1, s_1) \dots (s_{k-1}, \mathbf{a}_k, s_k)$ of transitions $(s_{j-1}, \mathbf{a}_j, s_j)$ in T_γ. Such a path is called a path from s_0 to s_k *labeled* by $w = \mathbf{a}_1 \dots \mathbf{a}_k$. When $s_0 = s_k$ the path is called a *cycle*. The previous corollary shows that for every run $\rho = \mathbf{c}_0 \dots \mathbf{c}_k$ in Ω_γ the following word θ_ρ is a cycle on s_γ in G_γ labeled by w:

$$\theta_\rho = (\pi_\gamma(\mathbf{c}_0), \mathbf{a}_1, \pi_\gamma(\mathbf{c}_1)) \dots (\pi_\gamma(\mathbf{c}_{k-1}), \mathbf{a}_k, \pi_\gamma(\mathbf{c}_k))$$

Corollary 7.4.10. *The graph* G_γ *is strongly connected.*

Proof. Let $s \in S_\gamma$. There exists $\mathbf{q} \in \mathbf{Q}_\gamma$ that occurs in a run $\rho \in \Omega_\gamma$ such that $s = \pi_\gamma(\mathbf{q})$. Hence there exist $u, v \in \mathbf{A}^*$ such that $\mathrm{src}(\rho) \xrightarrow{u} \mathbf{q} \xrightarrow{v} \mathrm{tgt}(\rho)$. Note that θ_ρ is the concatenation of a path from s_γ to s and a path from s to s_γ labeled by u, v. □

Corollary 7.4.11. *States in* S_γ *are incomparable.*

Proof. Let us consider $s \leq s'$ in S_γ. There exists $\mathbf{q}, \mathbf{q}' \in \mathbf{Q}_\gamma$ such that $s = \pi_\gamma(\mathbf{q})$ and $s' = \pi_\gamma(\mathbf{q}')$. Lemma 7.4.7 shows that there exists an intraproduction $\mathbf{h}' \in \mathbf{H}_\gamma$ such that $I_\gamma = \{i \mid \mathbf{h}'(i) = 0\}$. By replacing $\mathbf{h}'$ by a vector in $\mathbb{N}_{>0}\mathbf{h}'$ we can assume without loss of generality that $\mathbf{q}(i) \leq \mathbf{q}'(i) + \mathbf{h}'(i)$ for every $i \notin I_\gamma$. As

$\mathbf{q}(i) = s(i) \leq s'(i) = \mathbf{q}'(i) = \mathbf{q}'(i) + \mathbf{h}'(i)$ for every $i \in I_\gamma$ we deduce that $\mathbf{q} \leq \mathbf{q}' + \mathbf{h}'$. Lemma 7.4.5 shows that $\mathbf{q}' + \mathbf{h}' \in \mathbf{Q}_\gamma$. Lemma 7.4.6 shows that there exists an intraproduction $\mathbf{h} \in \mathbf{H}_\gamma$ such that $\mathbf{q}' + \mathbf{h}' \leq \mathbf{q} + \mathbf{h}$. As $\mathbf{h} \in \mathbf{H}_\gamma$ we deduce that $\mathbf{h}(i) = 0$ for every $i \in I_\gamma$. In particular $\mathbf{q}'(i) \leq \mathbf{q}(i)$ for every $i \in I_\gamma$. Hence $s' \leq s$ and we get $s = s'$. □

Corollary 7.4.12. *The class* $\{G_\gamma \mid \gamma \in \Gamma_{\mathbf{c}}\}$ *is finite.*

Proof. Given $I \subseteq \{1, \ldots, d\}$ we introduce the state $s_{\mathbf{c},I} \in \mathbb{N}^I$ defined by $s_{\mathbf{c},I}(i) = \mathbf{c}(i)$ for every $i \in I$. We also introduce the set $\Gamma_{\mathbf{c},I}$ of triples $\gamma \in \Gamma_{\mathbf{c}}$ such that $I_\gamma = I$. Note that in this case $s_{\mathbf{c},I}$ is equal to the special state s_γ for γ. Assume by contradiction that $S_{\mathbf{c},I} = \bigcup_{\gamma \in \Gamma_{\mathbf{c},I}} S_\gamma$ is infinite. For every $s \in S_{\mathbf{c},I}$ there exists $\gamma \in \Gamma_{\mathbf{c},I}$ such that $s \in S_\gamma$. Hence there exists a path p_s in G_γ from $s_{\mathbf{c},I}$ to s. Since the states in S_γ are incomparable, we can assume that the states occurring in p_s are incomparable. By inserting the paths p_s in a tree rooted by $s_{\mathbf{c},I}$ with transitions labeled by actions in $\mathbf{A}$ we deduce an infinite tree such that each node has a finite number of children (at most $|\mathbf{A}|$). Koenig's lemma shows that this tree has an infinite branch. Since $(\mathbb{N}^I, \leq)$ is well-ordered, there exists two comparable distinct nodes in this branch. There exists $s \in S_{\mathbf{c},I}$ such that these two comparable states occurs in p_s. We get a contradiction. Thus $S_{\mathbf{c},I}$ is finite. We deduce the corollary. □

7.4.3 Kirchhoff's Functions

We associate to the production graph G_γ a binary relation R_γ included in $\mathbb{Q}_{\geq 0} \overset{\mathbf{c}}{\curvearrowright}$ and such that $(\mathbf{x}, \mathbf{y}) \in R_\gamma$. This relation is based on *Kirchhoff's functions*.

A *Kirchhoff's function* for γ is a function $f : T_\gamma \to \mathbb{Q}$ labeling transitions of the production graph G_γ by rational numbers satisfying the following equality for every $s \in S_\gamma$:

$$\sum_{t \in T_\gamma \cap (\{s\} \times \mathbf{A} \times S_\gamma)} f(t) = \sum_{t \in T_\gamma \cap (S_\gamma \times \mathbf{A} \times \{s\})} f(t)$$

Kirchhoff's functions $f : T_\gamma \to \mathbb{N}_{>0}$ are characterized as follows. A cycle θ in G_γ is said to be *total* for γ if every transition in T_γ occurs in θ. The *Parikh image* of a path is the function $f : T_\gamma \to \mathbb{N}$ where $f(t)$ denotes the number of occurrences of t in the path. Since G_γ is strongly connected, *Euler's lemma* shows that a function $f : T_\gamma \to \mathbb{N}_{>0}$ is a Kirchhoff' function for γ if and only if f is the Parikh image of a total cycle for γ.

The *displacement* of a function $f : T_\gamma \to \mathbb{Q}$ is the sum $\sum_{t \in T_\gamma} f(t)\Delta(t)$ where $\Delta(t) = \mathbf{a}$ if $\mathbf{a}$ is the label of the transition t. This displacement is denoted by $\Delta(f)$. Let us observe that if f is the Parikh's image of a path in G_γ labeled by a word w then $\Delta(f) = \Delta(w)$. Intuitively the displacement of w only depends on the number of times transitions in T_γ occur in the path.

We introduce the relation R_γ of pairs $(\mathbf{u},\mathbf{v}) \in \mathbb{Q}^d_{\geq 0} \times \mathbb{Q}^d_{\geq 0}$ satisfying $\mathbf{u}(i) > 0$ iff $\mathbf{x}(i) > 0$, $\mathbf{v}(i) > 0$ iff $\mathbf{y}(i) > 0$, and such that there exists a Kirchhoff's function $f : T_\gamma \to \mathbb{Q}_{>0}$ such that $\mathbf{v}-\mathbf{u} = \Delta(f)$. Observe that R_γ is definable in FO $(\mathbb{Q},+,\leq)$.

Example 7.4.13. *Let us come back to Examples 7.4.3 and 7.4.8. A function $f : T_\gamma \to \mathbb{Q}$ is a Kirchhoff's function for γ if and only if $f((1,\star,1),\mathbf{a},(2,\star,0)) = f((2,\star,0),\mathbf{b},(1,\star,1))$ and $f((1,\star,1),\mathbf{b},(0,\star,2)) = f((0,\star,2),\mathbf{a},(1,\star,1))$. We get $R_\gamma = \{((0,0,0),(0,n,0)) \mid n \in \mathbb{Q}_{>0}\}$.*

Lemma 7.4.14. *We have $(\mathbf{x},\mathbf{y}) \in R_\gamma$.*

Proof. Assume that $T_\gamma = \{t_1,\dots,t_k\}$. By definition of T_γ, for every $j \in \{1,\dots,k\}$, there exists a run ρ_j such that t_j occurs in the cycle θ_{ρ_j}. Let w_j be the label of ρ_j and $n_j \in \mathbb{N}$ such that $\mathrm{dir}(\rho_j) \in (\mathbf{c},\mathbf{c}) + n_j(\mathbf{x},\mathbf{y})$. As $\mathbf{x} \overset{\mathbf{c}}{\curvearrowright} \mathbf{y}$ there exists a run ρ from $\mathbf{c}+\mathbf{x}$ to $\mathbf{c}+\mathbf{y}$ labeled by a word w. The cycle θ_ρ shows that w is the label of a cycle on s_γ. Let us consider $n = 1 + \sum_{j=1}^k n_j$ and $\sigma = ww_1\dots w_k$. Observe that σ is the label of a total cycle on s_γ. Hence the Parikh's image of this total cycle provides a Kirchhoff's function f for γ such that $\Delta(\sigma) = \Delta(f)$. Observe that $\Delta(\sigma) = n(\mathbf{y}-\mathbf{x})$. Hence $\mathbf{y}-\mathbf{x} = \Delta(\frac{1}{n}f)$ and we have proved that $(\mathbf{x},\mathbf{y}) \in R_\gamma$. □

Lemma 7.4.15. *We have $R_\gamma \subseteq \mathbb{Q}_{\geq 0} \overset{\mathbf{c}}{\curvearrowright}$.*

Proof. Lemma 7.4.7 shows that there exists $\mathbf{h}' \in \mathbf{H}_\gamma$ such that $I_\gamma = \{i \mid \mathbf{h}'(i) = 0\}$. From $\mathbf{h}' \in \mathbf{H}_\gamma$ we have a run ρ of the form $\mathbf{c}+n\mathbf{x} \xrightarrow{w_1} \mathbf{c}+\mathbf{h}' \xrightarrow{w_2} \mathbf{c}+n\mathbf{y}$ for some $n \in \mathbb{N}$ and $w_1,w_2 \in \mathbf{A}^*$. The cycle θ_ρ shows that there exist cycles θ_1,θ_2 on s_γ labeled by w_1,w_2. We denote by f_1 and f_2 the Parikh images of these two cycles. Let $(\mathbf{u},\mathbf{v}) \in R_\gamma$. By replacing $(\mathbf{u},\mathbf{v})$ by a pair in $\mathbb{N}_{>0}(\mathbf{u},\mathbf{v})$ we can assume without loss of generality that $\mathbf{u}' = \mathbf{u}-n\mathbf{x}$ and $\mathbf{v}' = \mathbf{v}-n\mathbf{y}$ are both in $\mathbb{N}^d$, and there exists a Kirchhoff's function f such that $f(t) \in \mathbb{N}_{>0}$ and $f(t) > f_1(t)+f_2(t)$ for every $t \in T_\gamma$, and such that $\mathbf{v}-\mathbf{u} = \Delta(f)$. Since $g = f-(f_1+f_2)$ is a Kirchhoff's function satisfying $g(t) \in \mathbb{N}_{>0}$ for every $t \in T_\gamma$, Euler's Lemma shows that g is the Parikh's image of a total cycle θ in G_γ on s_γ. Let σ be the label of this cycle and observe that $\Delta(\sigma) = \Delta(g) = \Delta(f)-(\Delta(f_1)+\Delta(f_2)) = \mathbf{v}-\mathbf{u}-((\mathbf{h}'-n\mathbf{x})+(n\mathbf{y}-\mathbf{h}')) = \mathbf{v}'-\mathbf{u}'$. Since $\mathbf{c}+n\mathbf{x} \xrightarrow{w_1} \mathbf{c}+\mathbf{h}' \xrightarrow{w_2} \mathbf{c}+n\mathbf{y}$ and $n\mathbf{x} \leq \mathbf{u}$, $n\mathbf{y} \leq \mathbf{v}$ we deduce by monotony that for every $m \in \mathbb{N}$ we have:

$$\mathbf{c}+m\mathbf{u} \xrightarrow{w_1^m} \mathbf{c}+m(\mathbf{h}'+\mathbf{u}') \qquad \mathbf{c}+m(\mathbf{h}'+\mathbf{v}') \xrightarrow{w_2^m} \mathbf{c}+m\mathbf{v}$$

We prove that there exists a run labeled by σ from $\mathbf{c}+m\mathbf{h}'$ for some $m \in \mathbb{N}_{>0}$ large enough as follows. We introduce the decomposition of σ into $\sigma = \mathbf{a}_1\dots\mathbf{a}_k$ where $\mathbf{a}_j \in \mathbf{A}$. Since θ is a cycle on the special state s_γ labeled by σ, there exists a sequence $(s_j)_{0\leq j\leq k}$ of states $s_j \in S_\gamma$ such that $\theta = (s_0,\mathbf{a}_1,s_1)\dots(s_{k-1},\mathbf{a}_k,s_k)$. Let $i \notin I_\gamma$ and $j \in \{0,\dots,k\}$. Since $\mathbf{h}'(i) > 0$ there exists $m_{i,j} \in \mathbb{N}$ such that the ith component of $\mathbf{c}+m_{i,j}\mathbf{h}'+\Delta(\mathbf{a}_1\dots\mathbf{a}_j)$ is in $\mathbb{N}$. Let $m \in \mathbb{N}_{>0}$ such that $m \geq m_{i,j}$ for every $i \notin I_\gamma$ and $j \in \{0,\dots,k\}$. Note that for every $i \in I_\gamma$ and for every $j \in \{0,\dots,k\}$, the ith component of $\mathbf{c}+\Delta(\mathbf{a}_1\dots\mathbf{a}_j)$ is equal to $s_j(i)$ which

is in $\mathbb{N}$. We have proved that $\mathbf{c} + m\mathbf{h}' + \Delta(\mathbf{a}_1 \dots \mathbf{a}_j) \in \mathbb{N}^d$ for every $j \in \{0, \dots, k\}$. Hence there exists a run from $\mathbf{c} + m\mathbf{h}'$ labeled by σ.

Let us consider $\ell \in \{0, \dots, m\}$ and let us introduce $\mathbf{z}_\ell = (m - \ell)\mathbf{u}' + \ell\mathbf{v}'$. Note that $\mathbf{z}_\ell \in \mathbb{N}^d$. By monotony there exists a run from $\mathbf{c} + m\mathbf{h}' + \mathbf{z}_\ell$ labeled by σ. Since $\Delta(\sigma) = \mathbf{v}' - \mathbf{u}'$, we get $\mathbf{z}_\ell + \Delta(\sigma) = \mathbf{z}_{\ell+1}$. We deduce that $\mathbf{c} + m\mathbf{h}' + \mathbf{z}_\ell \xrightarrow{\sigma} \mathbf{c} + m\mathbf{h}' + \mathbf{z}_{\ell+1}$. Therefore:

$$\mathbf{c} + m(\mathbf{h}' + \mathbf{u}') \xrightarrow{\sigma^m} \mathbf{c} + m(\mathbf{h}' + \mathbf{v}')$$

We have proved the lemma by observing that $\mathbf{c} + m\mathbf{u} \xrightarrow{w_1^m \sigma^m w_2^m} \mathbf{c} + m\mathbf{v}$. □

Corollary 7.4.16. *Transformer relations are asymptotically definable periodic relations.*

Proof. Lemma 7.4.14 and Lemma 7.4.15 show that $\mathbb{Q}_{\geq 0} \overset{\mathbf{c}}{\curvearrowright} = \bigcup_{\gamma \in \Gamma_{\mathbf{c}}} R_\gamma$. Since the class $\{G_\gamma \mid \gamma \in \Gamma_{\mathbf{c}}\}$ is finite we deduce that the class $\{R_\gamma \mid \gamma \in \Gamma_{\mathbf{c}}\}$ is finite. Recall that relations R_γ are definable in FO $(\mathbb{Q}, +, \leq)$. □

7.5 Reachability Relations Are Almost Semilinear

In this section the intersection of the reachability relation $\xrightarrow{*}$ with any Presburger relation $R \subseteq \mathbb{N}^d \times \mathbb{N}^d$ is proved to be almost semilinear. As a direct corollary we will deduce that $\mathrm{Post}^*(\mathbf{X}) \cap \mathbf{Y}$ and $\mathrm{Pre}^*(\mathbf{Y}) \cap \mathbf{X}$ are almost semilinear for every Presburger sets $\mathbf{X}, \mathbf{Y} \subseteq \mathbb{N}^d$. Since Presburger relations are finite unions of linear relations, we can assume that $R = r + P$ where $r \in \mathbb{N}^d \times \mathbb{N}^d$ and $P \subseteq \mathbb{N}^d \times \mathbb{N}^d$ is a finitely generated periodic relation. We introduce the set Ω of runs ρ such that $\mathrm{dir}(\rho) \in R$ equipped with the order $\sqsubseteq$ defined by $\rho \sqsubseteq \rho'$ if $\mathrm{dir}(\rho') \in \mathrm{dir}(\rho) + P$ and $\rho \trianglelefteq \rho'$. Since P is finitely generated, Dickson's lemma shows that $\sqsubseteq$ is a well-order. In particular we deduce that the set of minimal runs in Ω for $\sqsubseteq$, denoted by $\min_{\sqsubseteq}(\Omega)$ is finite.

Lemma 7.5.1. *The intersection of $\xrightarrow{*}$ with R is equal to:*

$$\bigcup_{\rho \in \min_{\sqsubseteq}(\Omega)} \mathrm{dir}(\rho) + (\overset{\rho}{\curvearrowright} \cap P)$$

Proof. Let us first prove that $\mathrm{dir}(\rho) + (\overset{\rho}{\curvearrowright} \cap P)$ is included in $\xrightarrow{*} \cap R$ for every run $\rho \in \Omega$. Assume that $\rho = \mathbf{c}_0 \dots \mathbf{c}_k$ with $\mathbf{c}_j \in \mathbb{N}^d$ and let $(\mathbf{u}, \mathbf{v}) \in P$ such that $\mathbf{u} \overset{\rho}{\curvearrowright} \mathbf{v}$. As $\rho \in \Omega$ we deduce that $(\mathbf{c}_0, \mathbf{c}_k) \in R$. As $\mathbf{u} \overset{\rho}{\curvearrowright} \mathbf{v}$ there exists a sequence $(\mathbf{v}_j)_{0 \leq j \leq k+1}$ of vectors $\mathbf{v}_j \in \mathbb{N}^d$ such that $\mathbf{v}_0 = \mathbf{u}$, $\mathbf{v}_{k+1} = \mathbf{v}$ and such that $\mathbf{v}_j \overset{\mathbf{c}_j}{\curvearrowright} \mathbf{v}_{j+1}$ for every $j \in \{0, \dots, k\}$. In particular there exists a run from $\mathbf{c}_j + \mathbf{v}_j$ to $\mathbf{c}_j + \mathbf{v}_{j+1}$ labeled by a word $w_j \in \mathbf{A}^*$. Now just observe that we have a run from $\mathbf{c}_0 + \mathbf{v}_0$ to $\mathbf{c}_k + \mathbf{v}_{k+1}$ labeled by $w_0 \mathbf{a}_1 w_1 \dots \mathbf{a}_k w_k$ where $\mathbf{a}_j = \mathbf{c}_j - \mathbf{c}_{j-1}$. Since $(\mathbf{c}_0, \mathbf{c}_k) \in r + P$ and $(\mathbf{u}, \mathbf{v}) \in P$ we deduce that $(\mathbf{c}_0 + \mathbf{u}, \mathbf{c}_k + \mathbf{v}) \in r + P + P \subseteq R$. Hence $\mathrm{dir}(\rho) + (\mathbf{u}, \mathbf{v})$ is in $\xrightarrow{*} \cap R$.

Now, let us prove that for every $(\mathbf{x}, \mathbf{y}) \in R$ such that $\mathbf{x} \xrightarrow{*} \mathbf{y}$ there exists $\rho \in \min_{\sqsubseteq}(\Omega)$ such that $(\mathbf{x}, \mathbf{y}) \in \mathrm{dir}(\rho) + (\overset{\rho}{\curvearrowright} \cap P)$. There exists a run $\rho' \in \Omega$ such that $\mathrm{dir}(\rho') = (\mathbf{x}, \mathbf{y})$. Since $\sqsubseteq$ is a well-order, there exists a run $\rho \in \min_{\sqsubseteq}(\Omega)$ such that $\rho \sqsubseteq \rho'$. By definition of $\sqsubseteq$ we deduce that $\mathrm{dir}(\rho') \in \mathrm{dir}(\rho) + (\overset{\rho}{\curvearrowright} \cap P)$. □

Since P is finitely generated it is asymptotically definable. From the following lemma we deduce that $\overset{\rho}{\curvearrowright} \cap P$ is an asymptotically definable periodic relation. Hence, the previous lemma proved that the intersection of the reachability relation $\xrightarrow{*}$ with every Presburger relation is almost semilinear.

Lemma 7.5.2. *Asymptotically definable periodic sets are stable by intersection.*

Proof. If $\mathbf{P}_1, \mathbf{P}_2 \subseteq \mathbb{Z}^d$ are two periodic sets then $\mathbf{P} = \mathbf{P}_1 \cap \mathbf{P}_2$ is a periodic set. Moreover, observe that $\mathbb{Q}_{\geq 0}(\mathbf{P}_1 \cap \mathbf{P}_2) = (\mathbb{Q}_{\geq 0}\mathbf{P}_1) \cap (\mathbb{Q}_{\geq 0}\mathbf{P}_2)$. Hence, if $\mathbf{P}_1, \mathbf{P}_2$ are asymptotically definable then $\mathbf{P}$ is also asymptotically definable. □

We deduce the following corollary.

Corollary 7.5.3. *The sets* $\mathrm{Post}^*(\mathbf{X}) \cap \mathbf{Y}$ *and* $\mathrm{Pre}^*(\mathbf{Y}) \cap \mathbf{X}$ *are almost semilinear for every Presburger sets* $\mathbf{X}, \mathbf{Y} \subseteq \mathbb{N}^d$.

Proof. Let us consider the Presburger relation $R = \mathbf{X} \times \mathbf{Y}$ and observe that $\mathrm{Post}^*(\mathbf{X}) \cap \mathbf{Y} = f(\xrightarrow{*} \cap R)$ and $\mathrm{Pre}^*(\mathbf{Y}) \cap \mathbf{X} = g(\xrightarrow{*} \cap R)$ where $f, g : \mathbb{Q}^d \times \mathbb{Q}^d \to \mathbb{Q}^d$ and defined by $f(\mathbf{x}, \mathbf{y}) = \mathbf{y}$ and $g(\mathbf{x}, \mathbf{y}) = \mathbf{x}$. Now just observe that for every $r \in \mathbb{N}^d \times \mathbb{N}^d$, for every asymptotically definable periodic relation $P \subseteq \mathbb{N}^d \times \mathbb{N}^d$, and for every $h \in \{f, g\}$ we have $h(r + P) = h(r) + h(P)$. Moreover $h(P)$ is a periodic set and the conic set $\mathbb{Q}_{\geq 0}h(P)$ is equal to $h(\mathbb{Q}_{\geq 0}P)$ which is definable in $\mathrm{FO}(\mathbb{Q}, +, \leq)$. □

7.6 Dimension

In this section we introduce a dimension function for the subsets of $\mathbb{Z}^d$ and we characterize the dimension of periodic sets.

A *vector space* is a set $\mathbf{V} \subseteq \mathbb{Q}^d$ such that $\mathbf{0} \in \mathbf{V}$, $\mathbf{V} + \mathbf{V} \subseteq \mathbf{V}$ and such that $\mathbb{Q}\mathbf{V} \subseteq \mathbf{V}$. Let $\mathbf{X} \subseteq \mathbb{Q}^d$. The following set $\mathbf{V}$ is a vector space called the *vector space generated* by $\mathbf{X}$.

$$\mathbf{V} = \left\{ \sum_{j=1}^{k} \lambda_j \mathbf{x}_j \mid k \in \mathbb{N} \text{ and } (\lambda_j, \mathbf{x}_j) \in \mathbb{Q} \times \mathbf{X} \right\}$$

This vector space is the minimal for the inclusion among the vector spaces that contain $\mathbf{X}$. Let us recall that every vector space $\mathbf{V}$ is generated by a finite set. The *rank* $\mathrm{rank}(\mathbf{V})$ of a vector space $\mathbf{V}$ is the minimal natural number $m \in \mathbb{N}$ such that there exists a finite set $\mathbf{X}$ with m vectors that generates $\mathbf{V}$. Let us recall that $\mathrm{rank}(\mathbf{V}) \leq d$ for every vector space $\mathbf{V} \subseteq \mathbb{Q}^d$ and $\mathrm{rank}(\mathbf{V}) \leq \mathrm{rank}(\mathbf{W})$ for

every pair of vector spaces $\mathbf{V} \subseteq \mathbf{W}$. Moreover, if $\mathbf{V}$ is strictly included in $\mathbf{W}$ then $\text{rank}(\mathbf{V}) < \text{rank}(\mathbf{W})$.

Example 7.6.1. *Vector spaces $\mathbf{V}$ included in $\mathbb{Q}^2$ satisfy $\text{rank}(\mathbf{V}) \in \{0,1,2\}$. Moreover these vectors spaces can be classified as follows : $\text{rank}(\mathbf{V}) = 0$ if and only if $\mathbf{V} = \{\mathbf{0}\}$, $\text{rank}(\mathbf{V}) = 1$ if and only if $\mathbf{V} = \mathbb{Q}\mathbf{v}$ with $\mathbf{v} \in \mathbb{Q}^2 \backslash \{\mathbf{0}\}$, and $\text{rank}(\mathbf{V}) = 2$ if and only if $\mathbf{V} = \mathbb{Q}^2$.*

The *dimension* of a set $\mathbf{X} \subseteq \mathbb{Z}^d$ is the minimal integer $m \in \{-1, \dots, d\}$ such that $\mathbf{X} \subseteq \bigcup_{j=1}^{k} \mathbf{b}_j + \mathbf{V}_j$ where $\mathbf{b}_j \in \mathbb{Z}^d$ and $\mathbf{V}_j \subseteq \mathbb{Q}^d$ is a vector space satisfying $\text{rank}(\mathbf{V}_j) \leq m$ for every j. We denote by $\dim(\mathbf{X})$ the dimension of $\mathbf{X}$. Observe that $\dim(\mathbf{v} + \mathbf{X}) = \dim(\mathbf{X})$ for every $\mathbf{X} \subseteq \mathbb{Z}^d$ and for every $\mathbf{v} \in \mathbb{Z}^d$. Moreover we have $\dim(\mathbf{X}) = -1$ if and only if $\mathbf{X}$ is empty. Note that $\dim(\mathbf{X} \cup \mathbf{Y}) = \max\{\dim(\mathbf{X}), \dim(\mathbf{Y})\}$ for every subsets $\mathbf{X}, \mathbf{Y} \subseteq \mathbb{Z}^d$.

Example 7.6.2. *Let $\mathbf{X} = \{-10, \dots, 10\} \times \mathbb{Z}$. Observe that $\dim(\mathbf{X}) \leq 1$ since the set $\mathbf{X}$ is included in $\bigcup_{\mathbf{b} \in \{-10,\dots,10\} \times \{0\}} \mathbf{b} + \mathbf{V}$ where $\mathbf{V} = \{0\} \times \mathbb{Q}$.*

Lemma 7.6.3. *Let $\mathbf{P} \subseteq \mathbb{Z}^d$ be a periodic set included in $\bigcup_{j=1}^{k} \mathbf{b}_j + \mathbf{V}_j$ where $k \in \mathbb{N}_{>0}$, $\mathbf{b}_j \in \mathbb{Z}^d$ and $\mathbf{V}_j \subseteq \mathbb{Q}^d$ is a vector space. There exists $j \in \{1, \dots, k\}$ such that $\mathbf{P} \subseteq \mathbf{V}_j$ and $\mathbf{b}_j \in \mathbf{V}_j$.*

Proof. Let us first prove by induction over $k \in \mathbb{N}_{>0}$ that for every periodic set $\mathbf{P} \subseteq \mathbb{Z}^d$ included in $\bigcup_{j=1}^{k} \mathbf{V}_j$ where $\mathbf{V}_j \subseteq \mathbb{Q}^d$ is a vector space, there exists $j \in \{1, \dots, k\}$ such that $\mathbf{P} \subseteq \mathbf{V}_j$. The rank $k = 1$ is immediate. Assume the rank k proved and let us prove the rank $k+1$. Let $\mathbf{P}$ be a periodic set included in $\bigcup_{j=1}^{k+1} \mathbf{V}_j$ where $\mathbf{V}_j \subseteq \mathbb{Q}^d$ is a vector space. If $\mathbf{P} \subseteq \mathbf{V}_{k+1}$ the induction is proved. So we can assume that there exists $\mathbf{p} \in \mathbf{P} \backslash \mathbf{V}_{k+1}$. Let $\mathbf{x} \in \mathbf{P}$. Since $\mathbf{p} + n\mathbf{x} \in \mathbf{P}$ for every $n \in \mathbb{N}$, the pigeon-hole principle shows that there exist $j \in \{1, \dots, k+1\}$ and $n < m$ such that $n\mathbf{p} + \mathbf{x}$ and $m\mathbf{p} + \mathbf{x}$ are both in $\mathbf{V}_j$. In particular the difference of this two vectors is in $\mathbf{V}_j$. Since this difference is $(m-n)\mathbf{p}$ and $\mathbf{p} \notin \mathbf{V}_{k+1}$ we get $j \in \{1, \dots, k\}$. Observe that $n(m\mathbf{p} + \mathbf{x}) - m(n\mathbf{p} + \mathbf{x})$ is the difference of two vectors in $\mathbf{V}_j$. Thus this vector is in $\mathbf{V}_j$ and we deduce that $\mathbf{x} \in \mathbf{V}_j$. We have shown that $\mathbf{P} \subseteq \bigcup_{j=1}^{k} \mathbf{V}_j$. By induction there exists $j \in \{1, \dots, k\}$ such that $\mathbf{P} \subseteq \mathbf{V}_j$. We have proved the induction.

Finally, assume that $\mathbf{P} \subseteq \mathbb{Z}^d$ is a periodic set included in $\bigcup_{j=1}^{k} \mathbf{b}_j + \mathbf{V}_j$ where $k \in \mathbb{N}_{>0}$, $\mathbf{b}_j \in \mathbb{Z}^d$ and $\mathbf{V}_j \subseteq \mathbb{Q}^d$ is a vector space. Let J be the set of $j \in \{1, \dots, k\}$ such that $\mathbf{b}_j \in \mathbf{V}_j$ and let us prove that $\mathbf{P} \subseteq \bigcup_{j \in J} \mathbf{V}_j$. Let $\mathbf{p} \in \mathbf{P}$. Since $n\mathbf{p} \in \mathbf{P}$ for every $n \in \mathbb{N}$, there exist $j \in \{1, \dots, k\}$ and $n < m$ such that $n\mathbf{p}$ and $m\mathbf{p}$ are both in $\mathbf{b}_j + \mathbf{V}_j$. The difference of these two vectors shows that $(m-n)\mathbf{p}$ is in $\mathbf{V}_j$. From $\mathbf{b}_j \in n\mathbf{p} - \mathbf{V}_j \subseteq \mathbf{V}_j$ we deduce that $j \in J$. Thus $\mathbf{P} \subseteq \bigcup_{j \in J} \mathbf{V}_j$. As $\mathbf{0} \in \mathbf{P}$ we deduce that $J \neq \emptyset$ and from the previous paragraph, there exists $j \in J$ such that $\mathbf{P} \subseteq \mathbf{V}_j$. □

Lemma 7.6.4. *We have $\dim(\mathbf{P}) = \text{rank}(\mathbf{V})$ for every periodic set $\mathbf{P}$ where $\mathbf{V}$ is the vector space generated by $\mathbf{P}$.*

Proof. Since $\mathbf{P} \subseteq \mathbf{V}$ we deduce that $\dim(\mathbf{P}) \leq \text{rank}(\mathbf{V})$. For the converse inequality, there exist $k \in \mathbb{N}$, $(\mathbf{b}_j)_{1\leq j\leq k}$ a sequence of vectors $\mathbf{b}_j \in \mathbb{Z}^d$ and a sequence $(\mathbf{V}_j)_{1\leq j\leq k}$ of vector spaces $\mathbf{V}_j \subseteq \mathbb{Q}^d$ such that $\mathbf{P} \subseteq \bigcup_{j=1}^{k} \mathbf{b}_j + \mathbf{V}_j$ and such that $\text{rank}(\mathbf{V}_j) \leq \dim(\mathbf{P})$ for every j. Since $\mathbf{P}$ is non empty we deduce that $k \in \mathbb{N}_{>0}$. Lemma 7.6.3 proves that there exists $j \in \{1, \ldots, k\}$ such that $\mathbf{P} \subseteq \mathbf{V}_j$ and $\mathbf{b}_j \in \mathbf{V}_j$. By minimality of the vector space generated by $\mathbf{P}$ we get $\mathbf{V} \subseteq \mathbf{V}_j$. Hence $\text{rank}(\mathbf{V}) \leq \text{rank}(\mathbf{V}_j)$. From $\text{rank}(\mathbf{V}_j) \leq \dim(\mathbf{P})$ we get $\text{rank}(\mathbf{V}) \leq \dim(\mathbf{P})$. □

7.7 Linearizations

A *linearization* of an almost semilinear set $\mathbf{X}$ is a set $\bigcup_{j=1}^{k} \mathbf{b}_j + (\mathbf{P}_j - \mathbf{P}_j) \cap \mathbb{Q}_{\geq 0}\mathbf{P}_j$ where $\mathbf{b}_j \in \mathbb{Z}^d$ and $\mathbf{P}_j \subseteq \mathbb{Z}^d$ is an asymptotically definable periodic set such that $\mathbf{X} = \bigcup_{j=1}^{k} \mathbf{b}_j + \mathbf{P}_j$. Let us recall that every *subgroup* of $(\mathbb{Z}^d, +)$ is finitely generated[Schrijver 1987]. Moreover, since $\text{FO}(\mathbb{Q}, +, \leq, 0)$ admits a quantifier elimination algorithm, we deduce that linearizations are definable in the Presburger arithmetic.

Remark 7.7.1. *Almost semilinear sets can have multiple linearizations.*

In this section we show that if $\mathbf{X}, \mathbf{Y} \subseteq \mathbb{Z}^d$ are two non-empty almost semilinear sets with an empty intersection then every linearizations $\mathbf{S}, \mathbf{T}$ of $\mathbf{X}, \mathbf{Y}$ satisfy:

$$\dim(\mathbf{S} \cap \mathbf{T}) < \dim(\mathbf{X} \cup \mathbf{Y})$$

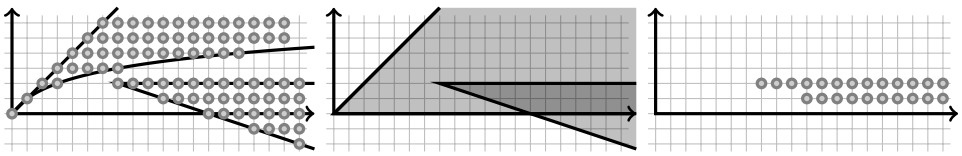


Figure 7.6: From left to right : sets $\mathbf{X}$ and $\mathbf{Y}$, sets $\mathbf{u} + \mathbb{Q}_{\geq 0}\mathbf{P}$ and $\mathbf{v} + \mathbb{Q}_{\geq 0}\mathbf{Q}$, and set $\mathbf{S} \cap \mathbf{T}$.

Example 7.7.2. *Sets introduced in this example are depicted in Figure 7.6. Let us introduce the asymptotically definable periodic set* $\mathbf{P} = \{\mathbf{p} \in \mathbb{N}^2 \mid \mathbf{p}(2) \leq \mathbf{p}(1) \leq 2^{\mathbf{p}(2)} - 1\}$ *and the finitely generated periodic set* $\mathbf{Q} = \mathbb{N}(1, 0) + \mathbb{N}(3, -1)$. *We introduce the almost semilinear sets* $\mathbf{X} = \mathbf{u} + \mathbf{P}$ *and* $\mathbf{Y} = \mathbf{v} + \mathbf{Q}$ *where* $\mathbf{u} = (0, 0)$ *and* $\mathbf{v} = (7, 2)$. *Observe that* $\mathbf{X} \cap \mathbf{Y}$ *is empty and* $\dim(\mathbf{X} \cup \mathbf{Y}) = 2$. *Let us consider linearizations* $\mathbf{S}, \mathbf{T}$ *of* $\mathbf{X}, \mathbf{Y}$ *defined by* $\mathbf{S} = \mathbf{u} + \mathbf{P}'$ *and* $\mathbf{T} = \mathbf{v} + \mathbf{Q}'$ *where* $\mathbf{P}' = (\mathbf{P} - \mathbf{P}) \cap \mathbb{Q}_{\geq 0}\mathbf{P}$ *and* $\mathbf{Q}' = (\mathbf{Q} - \mathbf{Q}) \cap \mathbb{Q}_{\geq 0}\mathbf{Q}$. *Observe that* $\mathbf{P}' = \{(0, 0)\} \cup \{\mathbf{p} \in \mathbb{N}^2_{>0} \mid \mathbf{p}(2) \leq \mathbf{p}(1)\}$ *and* $\mathbf{Q}' = \mathbf{Q}$. *Note that the intersection* $\mathbf{S} \cap \mathbf{T}$ *is non empty since it is equal to* $\{(7, 2), (10, 1)\} + \mathbb{N}(1, 0)$. *In particular* $\dim(\mathbf{S} \cap \mathbf{T}) \leq 1$ *and we get* $\dim(\mathbf{S} \cap \mathbf{T}) < \dim(\mathbf{X} \cup \mathbf{Y})$.

Lemma 7.7.3. *Assume that* $\mathbf{b}+\mathbf{M} \subseteq (\mathbf{P}-\mathbf{P})\cap\mathbb{Q}_{\geq 0}\mathbf{P}$ *where* $\mathbf{b} \in \mathbb{Z}^d$ *and* $\mathbf{M}, \mathbf{P} \subseteq \mathbb{Z}^d$ *are two periodic sets. Let* $\mathbf{a}$ *be a vector of the form* $\mathbf{m}_1+\cdots+\mathbf{m}_k$ *where* $(\mathbf{m}_j)_{1\leq j\leq k}$ *is a sequence of vectors* $\mathbf{m}_j \in \mathbf{M}$ *that generates a vector space that contains* $\mathbf{P}$*. There exists* $k \in \mathbb{N}_{>0}$ *such that* $\mathbf{b} + k\mathbb{N}_{>0}\mathbf{a} \subseteq \mathbf{P}$*.*

Proof. Since $\mathbf{b} \in \mathbf{P} - \mathbf{P}$ there exists $\mathbf{p}_+, \mathbf{p}_- \in \mathbf{P}$ such that $\mathbf{b} = \mathbf{p}_+ - \mathbf{p}_-$. As the sequence $(\mathbf{m}_j)_{1\leq j\leq k}$ generates a vector space that contains $\mathbf{P}$, we get $\mathbf{p}_+ \in \sum_{j=1}^k \mathbb{Q}\mathbf{m}_j$. Hence there exists $z \in \mathbb{N}_{>0}$ such that $-z\mathbf{p}_+ \in \sum_{j=1}^k \mathbb{Z}\mathbf{m}_j$. By definition of $\mathbf{a}$, there exists $n \in \mathbb{N}_{>0}$ such that $-z\mathbf{p}_+ + n\mathbf{a} \in \sum_{j=1}^k \mathbb{N}\mathbf{m}_j$. Hence $\mathbf{b} - z\mathbf{p}_+ + n\mathbf{a} \in \mathbf{b}+\sum_{j=1}^k \mathbb{N}\mathbf{m}_j$. Since this set is included in $\mathbb{Q}_{\geq 0}\mathbf{P}$ and $(z-1)\mathbf{p}_+ \in \mathbf{P}$ we deduce that $(\mathbf{b} - z\mathbf{p}_+ + n\mathbf{a}) + (z-1)\mathbf{p}_+$ is in $\mathbb{Q}_{\geq 0}\mathbf{P}$. Note that this vector is equal to $-\mathbf{p}_- + n\mathbf{a}$ since $\mathbf{b} = \mathbf{p}_+ - \mathbf{p}_-$. Hence, there exists $s \in \mathbb{N}_{>0}$ such that $s(-\mathbf{p}_- + n\mathbf{a}) \in \mathbf{P}$. Let $k = sn$ and observe that $-\mathbf{p}_- + k\mathbf{a} = s(-\mathbf{p}_- + n\mathbf{a}) + (s-1)\mathbf{p}_-$. Hence $-\mathbf{p}_- + k\mathbf{a} \in \mathbf{P}$. Since $\mathbf{b} + k\mathbf{a} = (-\mathbf{p}_- + k\mathbf{a}) + \mathbf{p}_+$ and $k\mathbf{a} = (-\mathbf{p}_- + k\mathbf{a}) + \mathbf{p}_-$ we deduce that $\mathbf{b} + k\mathbf{a}$ and $k\mathbf{a}$ are both in $\mathbf{P}$. In particular $\mathbf{b} + k\mathbb{N}_{>0}\mathbf{a} \subseteq \mathbf{P}$. □

Corollary 7.7.4. *Let* $\mathbf{X}, \mathbf{Y} \subseteq \mathbb{Z}^d$ *be two non-empty almost semilinear sets with an empty intersection. For every linearizations* $\mathbf{S}, \mathbf{T}$ *of* $\mathbf{X}, \mathbf{Y}$ *we have:*

$$\dim(\mathbf{S} \cap \mathbf{T}) < \dim(\mathbf{X} \cup \mathbf{Y})$$

Proof. We can assume that $\mathbf{X} = \mathbf{u} + \mathbf{P}$, $\mathbf{Y} = \mathbf{v} + \mathbf{Q}$ where $\mathbf{u}, \mathbf{v} \in \mathbb{Z}^d$ and $\mathbf{P}, \mathbf{Q} \subseteq \mathbb{Z}^d$ are two asymptotically definable periodic sets such that $\mathbf{X} \cap \mathbf{Y} = \emptyset$ and we can assume that $\mathbf{S} = \mathbf{u} + \mathbf{P}'$ where $\mathbf{P}' = (\mathbf{P} - \mathbf{P}) \cap \mathbb{Q}_{\geq 0}\mathbf{P}$ and $\mathbf{T} = \mathbf{v} + \mathbf{Q}'$ where $\mathbf{Q}' = (\mathbf{Q} - \mathbf{Q}) \cap \mathbb{Q}_{\geq 0}\mathbf{Q}$. Let $\mathbf{U}$ and $\mathbf{V}$ be the vector spaces generated by $\mathbf{P}$ and $\mathbf{Q}$. Lemma 7.6.4 shows that $\dim(\mathbf{X}) = \operatorname{rank}(\mathbf{U})$ and $\dim(\mathbf{Y}) = \operatorname{rank}(\mathbf{V})$. Note that $\mathbf{S} \cap \mathbf{T}$ is a Presburger set and in particular a finite union of linear sets. If this set is empty the corollary is proved. Otherwise there exists $\mathbf{b} \in \mathbb{Z}^d$ and a finitely generated periodic set $\mathbf{M} \subseteq \mathbb{Z}^d$ such that $\mathbf{b} + \mathbf{M} \subseteq \mathbf{S} \cap \mathbf{T}$ and such that $\dim(\mathbf{S} \cap \mathbf{T}) = \dim(\mathbf{b} + \mathbf{M})$. Let $\mathbf{W}$ be the vector space generated by $\mathbf{M}$. Observe that $\mathbf{b} + \mathbf{M} \subseteq (\mathbf{u} + \mathbf{U}) \cap (\mathbf{v} + \mathbf{V})$. Hence for every $\mathbf{m} \in \mathbf{M}$ since $\mathbf{b} + \mathbf{m} - \mathbf{u}$ and $\mathbf{b} + 2\mathbf{m} - \mathbf{u}$ are both in $\mathbf{U}$ the difference is also in $\mathbf{U}$. Hence $\mathbf{m} \in \mathbf{U}$. We deduce that $\mathbf{M} \subseteq \mathbf{U}$ and symmetrically $\mathbf{M} \subseteq \mathbf{V}$. As $\mathbf{M}$ is included in the vector space $\mathbf{U} \cap \mathbf{V}$, by minimality of $\mathbf{W}$, we get $\mathbf{W} \subseteq \mathbf{U} \cap \mathbf{V}$. Assume by contradiction that $\mathbf{W} = \mathbf{U}$ and $\mathbf{W} = \mathbf{V}$. Since $\mathbf{M}$ is finitely generated, there exists a sequence $(\mathbf{m}_j)_{1\leq j\leq k}$ of vectors $\mathbf{m}_j \in \mathbf{M}$ such that $\mathbf{M} = \mathbb{N}\mathbf{m}_1 + \cdots + \mathbb{N}\mathbf{m}_k$. Let $\mathbf{a} = \mathbf{m}_1 + \cdots + \mathbf{m}_k$. From $\mathbf{b} - \mathbf{u} + \mathbf{M} \subseteq (\mathbf{P} - \mathbf{P}) \cap \mathbb{Q}_{\geq 0}\mathbf{P}$ and Lemma 7.7.3 we deduce that there exists $k \in \mathbb{N}_{>0}$ such that $\mathbf{b} - \mathbf{u} + k\mathbb{N}_{>0}\mathbf{a} \subseteq \mathbf{P}$. From $\mathbf{b} - \mathbf{v} + \mathbf{M} \subseteq (\mathbf{Q} - \mathbf{Q}) \cap \mathbb{Q}_{\geq 0}\mathbf{Q}$ and Lemma 7.7.3 we deduce that there exists $k' \in \mathbb{N}_{>0}$ such that $\mathbf{b} - \mathbf{v} + k'\mathbb{N}_{>0}\mathbf{a} \subseteq \mathbf{Q}$. In particular $\mathbf{b} + kk'\mathbf{a} \in (\mathbf{u} + \mathbf{P}) \cap (\mathbf{v} + \mathbf{Q})$ and we get a contradiction since this intersection is empty. Thus $\mathbf{W} \neq \mathbf{U}$ or $\mathbf{W} \neq \mathbf{V}$. Since $\mathbf{W} \subseteq \mathbf{U} \cap \mathbf{V}$ we deduce that $\mathbf{W}$ is strictly included in $\mathbf{U}$ or in $\mathbf{V}$. Hence $\operatorname{rank}(\mathbf{W}) < \max\{\operatorname{rank}(\mathbf{U}), \operatorname{rank}(\mathbf{V})\} = \dim(\mathbf{X} \cup \mathbf{Y})$. From Lemma 7.6.4 we get $\dim(\mathbf{M}) = \operatorname{rank}(\mathbf{W})$ and since $\dim(\mathbf{M}) = \dim(\mathbf{S} \cap \mathbf{T})$ the corollary is proved. □

7.8 Presburger Invariants

We introduce the notion of *separators*. A *separator* is a pair $(\mathbf{X}, \mathbf{Y})$ of Presburger sets $\mathbf{X}, \mathbf{Y} \subseteq \mathbb{N}^d$ such that there does not exist a run from a configuration in $\mathbf{X}$ to a configuration in $\mathbf{Y}$. In particular $\mathbf{X} \cap \mathbf{Y} = \emptyset$. The Presburger set $\mathbf{D} = \mathbb{N}^d \backslash (\mathbf{X} \cup \mathbf{Y})$ is called the *domain* of $(\mathbf{X}, \mathbf{Y})$. We observe that a separator $(\mathbf{X}, \mathbf{Y})$ with an empty domain is a partition of $\mathbb{N}^d$ such that $\mathbf{X}$ is a Presburger forward inductive invariant and $\mathbf{Y}$ is a Presburger backward inductive invariant.

Lemma 7.8.1. *Let $(\mathbf{X}_0, \mathbf{Y}_0)$ be a separator with a non-empty domain $\mathbf{D}_0$. There exists a separator $(\mathbf{X}, \mathbf{Y})$ with a domain $\mathbf{D}$ such that $\mathbf{X}_0 \subseteq \mathbf{X}$, $\mathbf{Y}_0 \subseteq \mathbf{Y}$ and $\dim(\mathbf{D}) < \dim(\mathbf{D}_0)$.*

Proof. As $\mathbf{X}_0, \mathbf{D}_0$ are Presburger sets, Corollary 7.5.3 shows that $\mathbf{H} = \mathrm{Post}^*(\mathbf{X}_0) \cap \mathbf{D}_0$ is an almost semilinear set. We introduce a linearization $\mathbf{S}$ of this set. Since $(\mathbf{X}_0, \mathbf{Y}_0)$ is a separator, the intersection $\mathrm{Post}^*(\mathbf{X}_0) \cap \mathbf{Y}_0$ is empty. Moreover, as $\mathrm{Post}^*(\mathbf{X}_0) \cap \mathbf{D}_0 \subseteq \mathbf{S}$, we deduce that the set $\mathbf{Y} = \mathbf{Y}_0 \cup (\mathbf{D}_0 \backslash \mathbf{S})$ is such that $\mathrm{Post}^*(\mathbf{X}_0) \cap \mathbf{Y} = \emptyset$. Hence $(\mathbf{X}_0, \mathbf{Y})$ is a separator. Symmetrically, as $\mathbf{D}_0, \mathbf{Y}$ are Presburger sets, Corollary 7.5.3 shows that $\mathbf{K} = \mathrm{Pre}^*(\mathbf{Y}) \cap \mathbf{D}_0$ is an almost semilinear set. We introduce a linearization $\mathbf{T}$ of this set. Since $(\mathbf{X}_0, \mathbf{Y})$ is a separator, the intersection $\mathrm{Pre}^*(\mathbf{Y}) \cap \mathbf{X}_0$ is empty. Moreover, as $\mathrm{Pre}^*(\mathbf{Y}) \cap \mathbf{D}_0 \subseteq \mathbf{T}$, we deduce that the set $\mathbf{X} = \mathbf{X}_0 \cup (\mathbf{D}_0 \backslash \mathbf{T})$ is such that $\mathrm{Pre}^*(\mathbf{Y}) \cap \mathbf{X} = \emptyset$. Hence $(\mathbf{X}, \mathbf{Y})$ is a separator.

Let us introduce the domain $\mathbf{D}$ of $(\mathbf{X}, \mathbf{Y})$ and observe that $\mathbf{D} = \mathbf{D}_0 \cap \mathbf{S} \cap \mathbf{T}$. If $\mathbf{H}$ or $\mathbf{K}$ is empty then $\mathbf{S}$ or $\mathbf{T}$ is empty and in particular $\mathbf{D}$ is empty and the lemma is proved. So we can assume that $\mathbf{H}$ and $\mathbf{K}$ are non empty. Since $\mathbf{H} \subseteq \mathrm{Post}^*(\mathbf{X}_0) \subseteq \mathrm{Post}^*(\mathbf{X})$ and $\mathbf{K} \subseteq \mathrm{Pre}^*(\mathbf{Y})$ and $(\mathbf{X}, \mathbf{Y})$ is a separator, we deduce that $\mathbf{H} \cap \mathbf{K} = \emptyset$. Moreover as $\mathbf{H}, \mathbf{K} \subseteq \mathbf{D}_0$ we deduce that $\dim(\mathbf{H} \cup \mathbf{K}) \leq \dim(\mathbf{D}_0)$. As $\mathbf{S}$ and $\mathbf{T}$ are linearizations of the non-empty almost semilinear sets $\mathbf{H}$, $\mathbf{K}$ and $\mathbf{H} \cap \mathbf{K} = \emptyset$, Corollary 7.7.4 shows that $\dim(\mathbf{S} \cap \mathbf{T}) < \dim(\mathbf{H} \cup \mathbf{K})$. Therefore $\dim(\mathbf{D}) < \dim(\mathbf{D}_0)$. □

We deduce the main theorem of this paper.

Theorem 7.8.2. *For every $\mathbf{x}, \mathbf{y} \in \mathbb{N}^d$ such that there does not exist a run from $\mathbf{x}$ to $\mathbf{y}$, then there exists a pair $(\mathbf{X}, \mathbf{Y})$ of disjoint Presburger sets $\mathbf{X}, \mathbf{Y} \subseteq \mathbb{N}^d$ such that $\mathbf{X}$ is a forward inductive invariant that contains $\mathbf{x}$ and $\mathbf{Y}$ is a backward inductive invariant that contains $\mathbf{y}$.*

Proof. Observe that $(\{\mathbf{x}\}, \{\mathbf{y}\})$ is a separator.Thanks to Lemma 7.8.1 with an immediate induction over the dimension of the domains we deduce that there exists a separator $(\mathbf{X}, \mathbf{Y})$ with an empty domain such that $\mathbf{x} \in \mathbf{X}$ and $\mathbf{y} \in \mathbf{Y}$. □

7.9 Conclusion

The reachability problem for vector addition systems can be solved with a simple algorithm based on inductive invariants definable in the Presburger arithmetic. This

algorithm does not require the classical KLMST decomposition. Note however that the complexity of this algorithm is still open. In fact, the complexity depends on the minimal length of a run from $\mathbf{x}$ to $\mathbf{y}$ when such a run exists, or the minimal length of a Presburger formula denoting a forward inductive invariant $\mathbf{X}$ such that $\mathbf{x} \in \mathbf{X}$ and $\mathbf{y} \notin \mathbf{X}$ when such a formula exists. We left as an open question the problem of computing lower and upper bounds for these lengths. Note that the VAS exhibiting a large (Ackermann size) but finite reachability set given in [Mayr 1981c] does not directly provide an Ackermann lower-bound for these sizes since Presburger forward invariants can over-approximate reachability sets. Note that the existence of a primitive recursive upper bound of complexity for the reachability problem is still open since Zakaria Bouziane's paper[Bouziane 1998] introducing such a bound was proved to be incorrect by Petr Jančar[Jančar 2008].

As already mentioned, from a practical viewpoint, an algorithm that enumerates the Presburger formulas until it discovers one that denotes a precise enough inductive invariant is useless since it will never terminate in a reasonable amount of time. Nevertheless, the previous result shows that the Presburger arithmetic is expressive enough for separating unreachable configurations by inductive invariants definable in the Presburger arithmetic. In the future, we are interested in extending known semi-algorithms like the lazy interpolation model-checking or the accelerated symbolic model-checking (or even its abstract counterpart) in such a way that we get algorithms with termination guaranty that works well in practice for VASS.

Part III

Conclusion

Chapter 8

Conclusion

We presented in this document frameworks based on *good semialgorithms* for deciding the reachability problem for Presburger counter machines. Since the problem is in general undecidable even for the restricted class of Minsky machines[Minsky 1967], these semialgorithms do not have termination guaranty.

In the future we are interested in exploring *efficient algorithms for deciding the vector addition systems reachability problem.* This problem is central in the verification of infinite state systems and some other applications[Bojańczyk 2006, Demri 2009, Figueira 2009]. The reachability problem is difficult and its decidability remained open during 20 years. Nowadays, no decision procedure are implemented. In fact the known algorithm is conceptually difficult (to be understood and to be implemented) and with high computational complexity bound (at least Ackermann). Naturally, the good semialgorithms presented in this document can be applied for deciding some instances of the reachability problems. For instance, we can use our model-checker Fast, but we do not have any termination guaranty.

Our recent results on the reachability problem for Petri nets open research directions to solve this problem. The reminder of this chapter is divided in two sections corresponding respectively to research directions based on (1) the mathematical properties satisfied by the almost semilinear sets, and (2) the reachability problem for Petri nets.

8.1 Almost Semilinear Sets

An interesting approach for deciding the reachability problem consists in denoting reachability sets with formulas in a decidable formalism. The Presburger arithmetic seems to be a good candidate for vector addition systems with states (VASS). Hopcroft and Pansiot proved in [Hopcroft 1979] that reachability sets of VASS with 2 counters are definable in the Presburger arithmetic, but with 3 counters this property is no longer true. Hence the characterization of VASS reachability sets by formulas in a decidable logic seems to be a difficult problem. This intuition is confirmed by the undecidability of checking if two VASS reachability sets are equal[Hack 1976]. This result shows in particular that it is not possible to effectively compute formulas in a decidable logic denoting reachability sets.

Recently we proved in [Leroux 2011a] that VASS reachability sets are *almost semilinear*. This class is inspired by the geometrical characterization of the Presburger sets by *semilinear sets*[Ginsburg 1966]. Intuitively, semilinear sets and almost semilinear sets are both defined as finite union of "simple submonoids" of

$(\mathbb{Z}^d, +)$ translated by some vectors. Whereas these "simple submonoids" are assumed to be finitely generated for the semilinear sets, we simply require that the conic sets generated by these "simple submonoids" are definable in the decidable logic $\mathrm{FO}(\mathbb{Q}, +, \leq)$ in the case of almost semilinear sets.

Decidable Formalism

We proved that From the characterization of VASS reachability relations by almost semilinear sets, we deduced that unreachable configurations can be separated by inductive invariants definable in the Presburger arithmetic. Since the class of VASS reachability relations is stable by composition and transitive closure, it seems reasonable to check these stability properties over the class of almost semilinear relations. Such a result is not trivial since reachability relations of Presburger counter machines are transitive closure of binary relations definable in the Presburger arithmetic. However the VASS reachability relations satisfy and additional monotonic property. We say that a binary relation R over $\mathbb{N}^d$ is *monotonic* if $(\mathbf{v}, \mathbf{w}) \in R \Rightarrow (\mathbf{v} + \mathbf{z}, \mathbf{w} + \mathbf{z}) \in R$ for every $\mathbf{z} \in \mathbb{N}^d$. We conjecture that the transitive closure of monotonic almost semilinear relations are stable almost semilinear (the intersection of the relation with any semilinear relation should be assumed almost semilinear). With such a result we deduce a uniform method for solving the reachability problem for Petri nets extensions like the ones with zero-tests[Bonnet 2011, Reinhardt 2005].

Define a decidable formalism for manipulating monotonic binary relations over $\mathbb{N}^d$ by composition, intersection, union, transitive closure.

Semilinearity detection

In general VASS reachability sets are not definable in the Presburger arithmetic. In particular, the construction of an inductive invariant in the Presburger arithmetic separating some unreachable configurations requires to over-approximate the reachability set. We are interested in detecting which parts of the VASS reachability sets must be over-approximated in order to obtain inductive invariants definable in the Presburger arithmetic. This problem reduces to check if the reachability set of an initialized VASS is definable in the Presburger arithmetic. Such an initialized VASS is said to be *semilinear*.

Hauschildt and Lambert proved independently in the early 90's that the class of semilinear initialized VASS is recursive : we can effectively decide if the reachability set of an initialized VASS is definable in the Presburger arithmetic, moreover in the positive case a Presburger formula denoting this set is effectively computable. These two results are not well-known by the community. In fact, they are unpublished and the proofs are based on mathematical structures used by Kosaraju and Mayr for deciding the reachability problem. These structures suffer from complexity problems and implementation difficulties.

Find out a simple criterion for deciding the semilinearity of initialized VASS.

There exists a rich literature of classes of Petri nets with reachability sets definable in the Presburger arithmetic : Petri nets with 4 counters, conflict-free, persistent, regular, BPP-nets, reversible Petri nets. With Grégoire Sutre (LaBRI, Bordeaux), we proved in [Leroux 2005b] that all these classes are *flat*, that means their reachability sets can be computed with acceleration techniques. Whereas the symbolic model-checker FAST had been implemented in the objective of computing reachability sets of general Presburger counter machines, for these subclasses we have a termination guaranty. We are interested by the link between *flat* and *semilinear* for initialized VASS. Whereas the flat property clearly implies the semilinear one, the converse implication is open.

Show that flat is equivalent to semilinear for initialized VASS.

8.2 The Reachability Problem

The reachability problem for Petri nets was solved 30 years ago. Today, the complexity of this problem is still open. The complexity gap between the best known lower and upper bounds is huge. In 1976, the problem was proved to be EXPSPACE-hard[Cardoza 1976]. Concerning upper-bound, the algorithms introduced by Mayr and Kosaraju are non-primitive recursive. This complexity gap is very surprising in computer science. By comparison, the coverability problem is easier. The coverability problem is a variant of the reachability problem that consists in deciding if there exists a reachable configuration larger or equal (component-wise) than a given final configuration. This problem was proved to be EXPSPACE-complete [Cardoza 1976, Rackoff 1978].

Complexity upper bound

The best complexity lower bound for the reachability problem is EXPSPACE. This bound comes from a log-space reduction of the coverability problem. We do not know if this bound is tight since the coverability problem seems to be an easier problem compared to the reachability problem. Concerning the upper bound, we are interested by Ackermann functions. Such a bound is motivated by the Ackermann-complete problem that consists in checking the equality of two finite reachability sets[Mayr 1981c]. We are confident that such a bound can be obtained from the recent results presented in [Leroux 2011a]. Such a result should be a great theoretical progress for the reachability problem.

An Ackermann complexity upper bound for the reachability problem.

Good semi-algorithms à la Karp-Miller

The Karp and Miller algorithm [Karp 1969] provides a simple way for deciding the coverability problem. This algorithm logs a state space exploration of the reachability set with a finite tree. In order to enforce the termination of this algorithm, some reachable configurations are abstracted away by replacing integral components with a special symbol ∞, which intuitively denotes a very large number. Whereas this abstraction is fine for deciding the coverability problem, it is no longer sufficient for deciding the reachability problem. In order to overcome this limitation, the preciseness of this algorithm must be improved. The way we would like to explore consists in replacing the previous abstraction with a finer one. Inspired by the class of asymptotically definable periodic sets [Leroux 2011a] we are interested in equipping reachable markings with (1) a conic set definable in $\mathrm{FO}(\mathbb{Q}, +, \leq)$ denoting asymptotic directions of reachable markings, and (2) a subgroup of $(\mathbb{Z}^d, +)$ of the form $\mathbb{Z}\mathbf{v}_1 + \cdots + \mathbb{Z}\mathbf{v}_k$ denoting periodicities of reachable markings. We think that symmetrically to the coverability problem even if this algorithm is Ackermann in the worst case, in practice it should works well on interesting instances of the reachability problem.

A Karp and Miller algorithm for the reachability problem.

Bibliography

[Allouche 2003] Jean-Paul Allouche and Jeffrey O. Shallit. Automatic sequences - theory, applications, generalizations. Cambridge University Press, 2003.

[Alur 1995] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, Pei-Hsin Ho, X. Nicollin, A. Olivero, J. Sifakis and S. Yovine. *The algorithmic analysis of hybrid systems.* Theoretical Computer Science, vol. 138, no. 1, pages 3–34, 1995.

[Annichini 2001] A. Annichini, A. Bouajjani and M. Sighireanu. *TReX: A Tool for Reachability Analysis of Complex Systems.* In Proc. 13th Int. Conf. Computer Aided Verification (CAV'2001), Paris, France, July 2001, volume 2102 of *LNCS*, pages 368–372. Springer, 2001.

[Araki 1977] T. Araki and T. Kasami. *Decidable Problems on the Strong Connectivity of Petri Net Reachability Sets.* Theoretical Computer Science, vol. 4, no. 1, pages 99–119, 1977.

[Bagnara 2005] R. Bagnara, P. M. Hill, E. Ricci and E. Zaffanella. *Precise widening operators for convex polyhedra.* Science of Computer Programming, vol. 58, no. 1–2, pages 28–56, 2005.

[Bardin 2003] S. Bardin, A. Finkel, J. Leroux and L. Petrucci. *FAST: Fast Acceleration of Symbolic Transition systems.* In Proc. 15th Int. Conf. Computer Aided Verification (CAV'2003), Boulder, CO, USA, July 2003, volume 2725 of *LNCS*, pages 118–121. Springer, 2003.

[Bardin 2004] Sébastien Bardin, Alain Finkel and Jérôme Leroux. *FASTer Acceleration of Counter Automata in Practice.* In Tools and Algorithms for the Construction and Analysis of Systems, 10th International Conference, TACAS 2004, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2004, Barcelona, Spain, March 29 - April 2, 2004, Proceedings, volume 2988 of *Lecture Notes in Computer Science*, pages 576–590. Springer, 2004.

[Bardin 2005] S. Bardin, A. Finkel, J. Leroux and P. Schnoebelen. *Flat acceleration in symbolic model checking.* In Proc. 3rd Int. Symp. Automated Technology for Verification and Analysis (ATVA'05), Taipei, Taiwan, Oct. 2005, volume 3707 of *LNCS*, pages 474–488. Springer, 2005.

[Bardin 2006] Sébastien Bardin, Jérôme Leroux and Gérald Point. *FAST Extended Release.* In Proc. 18th Int. Conf. Computer Aided Verification (CAV'2006), Seattle, Washington, USA, August 2006, volume 4144 of *LNCS*, pages 63–66. Springer, 2006.

[Bardin 2008] Sébastien Bardin, Alain Finkel, Jérôme Leroux and Laure Petrucci. *FAST: acceleration from theory to practice.* International Journal on Software Tools for Technology Transfer (STTT), vol. 10, no. 5, pages 401–424, 2008.

[Bartzis 2004] Constantinos Bartzis and Tevfik Bultan. *Widening Arithmetic Automata.* In Proc. 16th Int. Conf. Computer Aided Verification (CAV 2004), , Boston, Massachussetts , July 2004, volume 3114 of *Lecture Notes in Computer Science*, pages 321–333. Springer, 2004.

[Becker 2007] Bernd Becker, Christian Dax, Jochen Eisinger and Felix Klaedtke. *LIRA: Handling Constraints of Linear Arithmetics over the Integers and the Reals.* In CAV, volume 4590 of *LNCS*, pages 307–310. Springer, 2007.

[Berman 1977] Leonard Berman. *Precise Bounds for Presburger Arithmetic and the Reals with Addition: Preliminary Report.* In Proc. 18th IEEE Symp. Foundations of Computer Science (FOCS'77), Providence, RI, USA, Oct.-Nov. 1977, pages 95–99, Providence, Rhode Island, 31 October–2 November 1977. IEEE.

[Beyer 2008] Dirk Beyer, Damien Zufferey and Rupak Majumdar. *CSIsat: Interpolation for LA+EUF.* In Computer Aided Verification, 20th International Conference, CAV 2008, Princeton, NJ, USA, July 7-14, 2008, Proceedings, volume 5123 of *Lecture Notes in Computer Science*, pages 304–308. Springer, 2008.

[Blumensath 2000] Achim Blumensath and Erich Grädel. *Automatic Structures.* In LICS, pages 51–62, 2000.

[Boigelot 1994] B. Boigelot and P. Wolper. *Symbolic Verification with Periodic Sets.* In Proc. 6th Int. Conf. Computer Aided Verification (CAV'94), Stanford, CA, USA, June 1994, volume 818 of *LNCS*, pages 55–67. Springer, 1994.

[Boigelot 1997] B. Boigelot, P. Godefroid, B. Willems and P. Wolper. *The Power of QDDs.* In Proc. Static Analysis 4th Int. Symp. (SAS'97), Paris, France, Sep. 1997, volume 1302 of *LNCS*, pages 172–186. Springer, 1997.

[Boigelot 1998] Bernard Boigelot. *Symbolic Methods for Exploring Infinite State Spaces.* PhD thesis, Université de Liège, 1998. Dans cette thèse, deux représentations symboliques sont présentées: les NDD pour représenter des vecteurs d'entier et les QDD pour représenter des contenus de file. L'acceleration de circuits de controles pour ces deux représentations sont étudiés.

[Boigelot 2009] Bernard Boigelot, Julien Brusten and Jérôme Leroux. *A Generalization of Semenov's Theorem to Automata over Real Numbers.* In Renate A. Schmidt, editeur, Automated Deduction, 22nd International Conference, CADE 2009, McGill University, Montreal,August 2 - 7, 2009 Pro-

ceedings, volume 5663 of *Lecture Notes in Computer Science*, pages 469–484. Springer, 2009.

[Bojańczyk 2006] M. Bojańczyk, A. Muscholl, Th. Schwentick, L. Segoufin and C. David. *Two-variable logic on words with data.* In LICS 2006, pages 7–16. IEEE, 2006. To appear in *ACM T. Comput. Log.*

[Bonnet 2011] Rémi Bonnet. *The Reachability Problem for Vector Addition System with One Zero-Test.* In Mathematical Foundations of Computer Science 2011 - 36th International Symposium, MFCS 2011, Warsaw, Poland, August 22-26, 2011. Proceedings, volume 6907 of *Lecture Notes in Computer Science*, pages 145–157. Springer, 2011.

[Bouajjani 1999a] A. Bouajjani and P. Habermehl. *Symbolic reachability analysis of FIFO-channel systems with nonregular sets of configurations.* Theoretical Computer Science, vol. 221, no. 1–2, pages 211–250, 1999.

[Bouajjani 1999b] A. Bouajjani and R. Mayr. *Model Checking Lossy Vector Addition Systems.* In Proc. 16th Ann. Symp. Theoretical Aspects of Computer Science (STACS'99), Trier, Germany, Mar. 1999, volume 1563 of *LNCS*, pages 323–333. Springer, 1999.

[Bouchy 2008] Florent Bouchy, Jérôme Leroux and Alain Finkel. *Decomposition of Decidable First-Order Logics over Integers and Reals.* In Temporal Representation and Reasoning, 15th International Symposium, TIME 2008, Montréal, Canada, June 16 - 18, 2008, Proceedings, pages 147–155. IEEE Computer Society Press, 2008.

[Boudet 1996] Alexandre Boudet and Hubert Comon. *Diophantine equations, Presburger arithmetic and finite automata.* In Proc. 21st Int. Coll. on Trees in Algebra and Programming (CAAP'96), Linköping, Sweden, Apr. 1996, volume 1059 of *Lecture Notes in Computer Science*, pages 30–43. Springer, 1996.

[Bouziane 1998] Z. Bouziane. *A primitive recursive algorithm for the general Petri net reachability problem.* In FOCS 1998, pages 130 –136, nov 1998.

[Bozzelli 2011] Laura Bozzelli and Pierre Ganty. *Complexity Analysis of the Backward Coverability Algorithm for VASS.* In Reachability Problems - 5th International Workshop, RP 2011, Genoa, Italy, September 28-30, 2011. Proceedings, volume 6945 of *Lecture Notes in Computer Science*, pages 96–109. Springer, 2011.

[Brillout 2010] Angelo Brillout, Daniel Kroening, Philipp Rümmer and Thomas Wahl. *An Interpolating Sequent Calculus for Quantifier-Free Presburger Arithmetic.* In IJCAR, volume 6173 of *LNCS*. Springer, 2010.

[Bruttomesso 2010] Roberto Bruttomesso, Edgar Pek, Natasha Sharygina and Aliaksei Tsitovich. *The OpenSMT Solver.* In Javier Esparza and Rupak Majumdar, editeurs, TACAS, volume 6015 of *LNCS*, pages 150–153. Springer, 2010.

[Bruyère 1994] Véronique Bruyère, Georges Hansel, Christian Michaux and Roger Villemaire. *Logic and p-recognizable Sets of Integers.* Bull. Belg. Math. Soc., vol. 1, no. 2, pages 191–238, March 1994.

[Bultan 2001] Tevfik Bultan and Tuba Yavuz-Kahveci. *Action Language Verifier.* In Proc. 16th IEEE Int. Conf. Automated Software Engineering (ASE 2001), 26-29 November 2001, Coronado Island, San Diego, CA, USA, pages 382–386. IEEE Computer Society, 2001.

[Burch 1990] Jerry R. Burch, Edmund M. Clarke, Kenneth L. McMillan, David L. Dill and L. J. Hwang. *Symbolic Model Checking: $10\hat{2}0$ States and Beyond.* In Proceedings, Fifth Annual IEEE Symposium on Logic in Computer Science, 4-7 June 1990, Philadelphia, Pennsylvania, USA, pages 428–439. IEEE Computer Society, 1990.

[Büchi 1962] J. R. Büchi. *On a Decision Method in Restricted Second Order Arithmetic.* In Proc. International Congress on Logic, Methodoloy and Philosophy of Science, pages 1–12, Stanford, 1962. Stanford University Press.

[Cardoza 1976] E. Cardoza, Richard J. Lipton and Albert R. Meyer. *Exponential Space Complete Problems for Petri Nets and Commutative Semigroups: Preliminary Report.* In Proceedings of the 8th Annual ACM Symposium on Theory of Computing, May 3-5, 1976, Hershey, Pennsylvania, USA, pages 50–54. ACM, 1976.

[Cimatti 2009] Alessandro Cimatti, Alberto Griggio and Roberto Sebastiani. *Interpolant Generation for UTVPI.* In Renate A. Schmidt, editeur, CADE, volume 5663 of *LNCS*, pages 167–182. Springer, 2009.

[Clarke 2003] E. M. Clarke, O. Grumberg, S. Jha, Y. Lu and H. Veith. *CounterExample-Guided Abstraction Refinement for Symbolic Model Checking.* Journal of the ACM, vol. 50, no. 5, pages 752–794, 2003.

[Cobham 1969] A. Cobham. *On the Base-dependence of Sets of Numbers Recognizable by Finite Automata.* Mathematical Systems Theory, vol. 3, pages 186–192, 1969.

[Comon 1998] H. Comon and Y. Jurski. *Multiple counters automata, safety analysis and Presburger arithmetic.* In Proc. 10th Int. Conf. Computer Aided Verification (CAV'98), Vancouver, BC, Canada, June-July 1998, volume 1427 of *LNCS*, pages 268–279. Springer, 1998.

[Cormen 1989] Thomas H. Cormen, Charles E. Leiserson and Ronald L. Rivest. Introduction to algorithms. The MIT Press and McGraw-Hill Book Company, 1989.

[Costan 2005] A. Costan, S. Gaubert, E. Goubault, M. Martel and S. Putot. *A Policy Iteration Algorithm for Computing Fixed Points in Static Analysis of Programs.* In In Proc. 7th Int. Conf. on Computer Aided Verification (CAV'05), Edinburgh, Scotland, UK, July 2005, LNCS, pages 462–475. Springer, 2005.

[Cousot 1977] P. Cousot and R. Cousot. *Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints.* In Proc. 4th ACM Symp. Principles of Programming Languages, Los Angeles, CA, USA, pages 238–252. ACM Press, 1977.

[Cousot 1978] Patrick Cousot and Nicolas Halbwachs. *Automatic Discovery of Linear Restraints Among Variables of a Program.* In POPL, pages 84–96, 1978.

[Cousot 1992] P. Cousot and R. Cousot. *Comparing the Galois Connection and Widening/Narrowing Approaches to Abstract Interpretation.* In Proc. 4th Int. Symp. Programming Language Implementation and Logic Programming (PLILP'92), Leuven, Belgium, Aug. 1992, volume 631 of *LNCS*, pages 269–295. Springer, 1992.

[Couvreur 2004] Jean-Michel Couvreur. *A bdd-like implementation of an automata package.* In Proc. 9th Int. Conf. Implementation and Application of Automata (CIAA'2004), Kingston, Canada, July 2004, volume 3317 of *LNCS*, pages 310–311. Springer, 2004.

[Delzanno 2004] G. Delzanno, J.-F. Raskin and L. Van Begin. *Covering sharing trees: a compact data structure for parameterized verification.* Journal of Software Tools for Technology Transfer, vol. 5, no. 2–3, pages 268–297, 2004.

[Demri 2009] S. Demri and R. Lazić. *LTL with the freeze quantifier and register automata.* ACM Trans. Computational Logic, vol. 10, no. 3, art. 16, 2009.

[Dickson 1913] L. E. Dickson. *Finiteness of the odd perfect and primitive abundant numbers with n distinct prime factors.* Amer. Journal Math., pages 413–422, 1913.

[Durand-Gasselin 2010] Antoine Durand-Gasselin and Peter Habermehl. *On the Use of Non-deterministic Automata for Presburger Arithmetic.* In CONCUR 2010 - Concurrency Theory, 21th International Conference, CONCUR 2010, Paris, France, August 31-September 3, 2010. Proceedings, volume 6269 of *Lecture Notes in Computer Science*, pages 373–387. Springer, 2010.

[Dutertre 2006] Bruno Dutertre and Leonardo Mendonça de Moura. *A Fast Linear-Arithmetic Solver for DPLL(T).* In CAV, volume 4144 of *LNCS*, pages 81–94. Springer, 2006.

[Esparza 1994] J. Esparza and M. Nielsen. *Decidability Issues for Petri Nets - a Survey.* Bulletin of the European Association for Theoretical Computer Science, vol. 52, pages 245–262, 1994.

[Esparza 1997] J. Esparza. *Petri Nets, Commutative Context-Free Grammars, and Basic Parallel Processes.* Fundamenta Informaticae, vol. 31, no. 1, pages 13–25, 1997.

[Feret 2004] Jérôme Feret. *Static Analysis of Digital Filters.* In Programming Languages and Systems, 13th European Symposium on Programming, ESOP 2004, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2004, Barcelona, Spain, March 29 - April 2, 2004, Proceedings, volume 2986 of *Lecture Notes in Computer Science*, pages 33–48. Springer, 2004.

[Figueira 2009] D. Figueira and L. Segoufin. *Future-looking logics on data words and trees.* In MFCS 2009, volume 5734 of *LNCS*, pages 331–343. Springer, 2009.

[Finkel 2000a] A. Finkel and G. Sutre. *An algorithm constructing the semilinear post* for 2-dim Reset/Transfer VASS.* In Proc. 25th Int. Symp. Math. Found. Comp. Sci. (MFCS'2000), Bratislava, Slovakia, Aug. 2000, volume 1893 of *LNCS*, pages 353–362. Springer, 2000.

[Finkel 2000b] A. Finkel and G. Sutre. *Decidability of reachability problems for classes of two counters automata.* In Proc. 17th Ann. Symp. Theoretical Aspects of Computer Science (STACS'2000), Lille, France, Feb. 2000, volume 1770 of *LNCS*, pages 346–357. Springer, 2000.

[Finkel 2002] A. Finkel and J. Leroux. *How To Compose Presburger-Accelerations: Applications to Broadcast Protocols.* In Proc. 22nd Conf. Found. of Software Technology and Theor. Comp. Sci. (FST&TCS'2002), Kanpur, India, Dec. 2002, volume 2556 of *LNCS*, pages 145–156. Springer, 2002.

[Finkel 2003] A. Finkel, S. P. Iyer and G. Sutre. *Well-Abstracted Transition Systems: Application to FIFO automata.* Information and Computation, vol. 181, no. 1, pages 1–31, 2003.

[Finkel 2005] Alain Finkel and Jérôme Leroux. *The convex hull of a regular set of integer vectors is polyhedral and effectively computable.* Information Processing Letters, vol. 96, no. 1, pages 30–35, 2005.

[Fribourg 1997a] L. Fribourg and H. Olsén. *A Decompositional Approach for Computing Least Fixed-Points of Datalog Programs with Z-counters.* Constraints, vol. 2, no. 3/4, pages 305–335, 1997.

[Fribourg 1997b] L. Fribourg and H. Olsén. *Proving Safety Properties of Infinite State Systems by Compilation into Presburger Arithmetic.* In Proc. 8th Int.

Conf. Concurrency Theory (CONCUR'97), Warsaw, Poland, Jul. 1997, volume 1243 of *LNCS*, pages 213–227. Springer, 1997.

[Geeraerts 2005] Gilles Geeraerts, Jean-François Raskin, and Laurent Van Begin. *Expand, Enlarge and Check... Made efficient.* In S. K. Rajjamani and K. Etessami, editeurs, Proceedings of 17th International Conference on Computer Aided Verification – (CAV 2005), numéro 3576 de LNCS, pages 394–404. Springer, 2005.

[Ginsburg 1966] S. Ginsburg and E. H. Spanier. *Semigroups, Presburger formulas and languages.* Pacific J. Math., vol. 16, no. 2, pages 285–296, 1966.

[Gonnord 2006] L. Gonnord and N. Halbwachs. *Combining Widening and Acceleration in Linear Relation Analysis.* In Proc. Static Analysis, 13th Int. Symp. (SAS'06), Seoul, Korea, Aug. 2006, volume 4134 of *LNCS*, pages 144–160. Springer, 2006.

[Graf 1997] S. Graf and H. Saïdi. *Construction of Abstract State Graphs with PVS.* In Proc. of 9th Conf. on Computer Aided Verification (CAV'97), volume 1254 of *LNCS*, pages 72–83, 1997.

[Gulavani 2006] B. Gulavani, T. A. Henzinger, Y. Kannan, A. Nori and S. K. Rajamani. *Synergy: A New Algorithm for Property Checking.* In Proc. of 14th Symp. on Foundations of Software Engineering (FSE'06), pages 117–127. ACM Press, 2006.

[Hack 1976] Michel Hack. *The Equality Problem for Vector Addition Systems is Undecidable.* Theoretical Computer Science, vol. 2, no. 1, pages 77–95, 1976.

[Hauschildt 1990] Dirk Hauschildt. *Semilinearity of the Reachability Set is Decidable for Petri Nets.* PhD thesis, University of Hamburg, 1990.

[Henzinger 2002] T. A. Henzinger, R. Jhala, R. Majumbar and G. Sutre. *Lazy Abstraction.* In Proc. of 29th Symp. on Principles of Programming Languages (POPL'02), pages 58–70, 2002.

[Hopcroft 1979] J. E. Hopcroft and J.-J. Pansiot. *On the reachability problem for 5-dimensional vector addition systems.* Theoretical Computer Science, vol. 8, no. 2, pages 135–159, 1979.

[Ibarra 1978] O.H. Ibarra. *Reversal-Bounded Multicounter Machines and Their Decision Problems.* Journal of the ACM, vol. 25, no. 1, pages 116–133, 1978.

[Jain 2008] Himanshu Jain, Edmund M. Clarke and Orna Grumberg. *Efficient Craig Interpolation for Linear Diophantine (Dis)Equations and Linear Modular Equations.* In CAV, LNCS. Springer, 2008.

[Jančar 1990a] P. Jančar. *Decidability of a Temporal Logic Problem for Petri Nets.* Theoretical Computer Science, vol. 74, no. 1, pages 71–93, 1990.

[Jančar 1990b] Petr Jančar. *Decidability of a temporal logic problem for Petri nets.* Theoretical Computer Science, vol. 74, no. 1, pages 71 – 93, 1990.

[Jančar 2001] Petr Jančar. *Nonprimitive recursive complexity and undecidability for Petri net equivalences.* Theoretical Computer Science, vol. 256, no. 1–2, pages 23–30, 2001.

[Jančar 2008] Petr Jančar. *Bouziane's transformation of the Petri net reachability problem and incorrectness of the related algorithm.* Inf. Comput., vol. 206, pages 1259–1263, November 2008.

[Jhala 2006] R. Jhala and K. L. McMillan. *A Practical and Complete Approach to Predicate Refinement.* In Proc. of 12th Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'06), volume 3920 of *LNCS*, pages 459–473. Springer, 2006.

[Karp 1969] R. M. Karp and R. E. Miller. *Parallel Program Schemata.* Journal of Computer and System Sciences, vol. 3, no. 2, pages 147–195, 1969.

[Karr 1976] M. Karr. *Affine Relationship Among Variables of a Program.* Acta Informatica, vol. 6, pages 133–141, 1976.

[Kelly 1995] Wayne Kelly, William Pugh, Evan Rosser and Tatiana Shpeisman. *Transitive Closure of Infinite Graphs and Its Applications.* In 8th Int. Wor. Languages and Compilers for Parallel Computing (LCPC'95), Columbus, Ohio, USA, August 10-12, 1995, volume 1033 of *Lecture Notes in Computer Science*, pages 126–140. Springer, 1995.

[Klaedtke 2004] Felix Klaedtke. *On the Automata Size for Presburger Arithmetic.* In Proc. 19th Annual IEEE Symposium on Logic in Computer Science (LICS'04), Turku, Finland July 2004, pages 110–119. IEEE Comp. Soc. Press, 2004.

[Kopetz 1994] Hermann Kopetz and Günter Grünsteidl. *TTP - A Protocol for Fault-Tolerant Real-Time Systems.* IEEE Computer, vol. 27, no. 1, pages 14–23, 1994.

[Kosaraju 1982] S. R. Kosaraju. *Decidability of reachability in vector addition systems.* In Proc. 14th ACM Symp. Theory of Computing (STOC'82), San Francisco, CA, May 1982, pages 267–281, 1982.

[Kroening 2010] Daniel Kroening, Jérôme Leroux and Philipp Rümmer. *Interpolating Quantifier-Free Presburger Arithmetic.* In Christian G. Fermüller and Andrei Voronkov, editeurs, Logic for Programming, Artificial Intelligence, and Reasoning - 17th International Conference, LPAR-17, Yogyakarta, Indonesia, October 10-15, 2010. Proceedings, volume 6397 of *Lecture Notes in Computer Science*, pages 489–503. Springer, 2010.

[Lambert 1992] Jean Luc Lambert. *A structure to decide reachability in Petri nets.* Theoretical Computer Science, vol. 99, no. 1, pages 79–104, 1992.

[Landweber 1978] L.H. Landweber and E.L. Robertson. *Properties of Conflict-Free and Persistent Petri Nets.* Journal of the ACM, vol. 25, no. 3, pages 352–364, 1978.

[Latour 2004] L. Latour. *From Automata to Formulas: Convex Integer Polyhedra.* In Proc. 19th Annual IEEE Symposium on Logic in Computer Science (LICS'04), Turku, Finland July 2004, pages 120–129. IEEE Comp. Soc. Press, 2004.

[Leroux 2003] Jérôme Leroux. *Algorithmique de la vérification des systèmes à compteurs. Approximation et accélération. Implémentation de l'outil Fast.* PhD thesis, Ecole Normale Supérieure de Cachan, Laboratoire Spécification et Vérification. CNRS UMR 8643, décembre 2003.

[Leroux 2005a] J. Leroux. *A Polynomial Time Presburger Criterion and Synthesis for Number Decision Diagrams.* In 20th IEEE Symposium on Logic in Computer Science (LICS 2005), 26-29 June 2005, Chicago, IL, USA, Proceedings, pages 147–156. IEEE Computer Society, 2005.

[Leroux 2005b] Jérôme Leroux and Grégoire Sutre. *Flat Counter Automata Almost Everywhere!* In Automated Technology for Verification and Analysis, Third International Symposium, ATVA 2005, Taipei, Taiwan, October 4-7, 2005, Proceedings, volume 3707 of *Lecture Notes in Computer Science*, pages 489–503. Springer, 2005.

[Leroux 2006] Jérôme Leroux. *Least Significant Digit First Presburger Automata.* CoRR, vol. abs/cs/0612037, 2006.

[Leroux 2007a] Jérôme Leroux and Grégoire Sutre. *Accelerated Data-Flow Analysis.* In Static Analysis, 14th International Symposium, SAS 2007, Kongens Lyngby, Denmark, August 22-24, 2007, Proceedings, volume 4634 of *Lecture Notes in Computer Science*, pages 184–199. Springer, 2007.

[Leroux 2007b] Jérôme Leroux and Grégoire Sutre. *Acceleration in Convex Data-Flow Analysis.* In Foundations of Software Technology and Theoretical Computer Science, 27th International Conference, FSTTCS 2007, New Delhi, India, December 12-14, 2007, Proceedings, volume 4855 of *Lecture Notes in Computer Science*, pages 520–531. Springer, 2007.

[Leroux 2008a] Jérôme Leroux. *Convex Hull of Arithmetic Automata.* In Static Analysis, 15th International Symposium, SAS 2008, Valencia, Spain, Jully 16-18, 2008, Proceedings, volume 5079 of *Lecture Notes in Computer Science*, pages 47–61. Springer, 2008.

[Leroux 2008b] Jérôme Leroux. *Structural Presburger Digit Vector Automata.* Theoretical Computer Science, vol. 409, no. 3, pages 549–556, 2008.

[Leroux 2009a] Jérôme Leroux. *The General Vector Addition System Reachability Problem by Presburger Inductive Invariants.* In 24th IEEE Symposium on Logic in Computer Science (LICS 2009), 11-14 August 2009, Los Angeles, California, USA, Proceedings, pages 4–13. IEEE Computer Society, 2009.

[Leroux 2009b] Jérôme Leroux and Gérald Point. *TaPAS : The Talence Presburger Arithmetic Suite.* In Stefan Kowalewski and Anna Philippou, editeurs, Tools and Algorithms for the Construction and Analysis of Systems, 15th International Conference, TACAS 2009, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2009, York, UK, March 22-29, 2009. Proceedings, volume 5505 of *Lecture Notes in Computer Science*, pages 182–185. Springer, 2009.

[Leroux 2011a] Jérôme Leroux. *Vector addition system reachability problem: a short self-contained proof.* In Proceedings of the 38th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2011, Austin, Texas, January 26-28, 2011, POPL '11, pages 307–316. ACM, 2011.

[Leroux 2011b] Jérôme Leroux. *The Vector Addition System Reversible Reachability Problem.* In CONCUR 2011 - Concurrency Theory, 22th International Conference, Aachen, Germany, 5-10 September, 2011, Proceedings, Lecture Notes in Computer Science. Springer, 2011. to appear.

[Lugiez 2004] Denis Lugiez. *From Automata to semilinear sets: a solution for polyhedra and even more general sets.* In Proc. 9th. Int. Conf. on Implementation and Application of Automata (CIAA'04), Queen's University, Kingston, Ontario, Canada, Jul. 2004, volume 3317 of *Lecture Notes in Computer Science*, pages 321–322. Springer, 2004.

[Lynch 2008] Christopher Lynch and Yuefeng Tang. *Interpolants for Linear Arithmetic in SMT.* In ATVA, LNCS. Springer, 2008.

[Mayr 1981a] E. W. Mayr. *Persistence of Vector Replacement Systems is Decidable.* Acta Informatica, vol. 15, pages 309–318, 1981.

[Mayr 1981b] Ernst W. Mayr. *An Algorithm for the General Petri Net Reachability Problem.* In Conference Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computation, (STOC 1981), 11-13 May 1981, Milwaukee, Wisconsin, USA, pages 238–246. ACM, 1981.

[Mayr 1981c] Ernst W. Mayr and Albert R. Meyer. *The Complexity of the Finite Containment Problem for Petri Nets.* J. ACM, vol. 28, no. 3, pages 561–576, 1981.

[Mayr 2000] Richard Mayr. *Undecidable Problems in Unreliable Computations.* In LATIN 2000: Theoretical Informatics, 4th Latin American Symposium, Punta del Este, Uruguay, April 10-14, 2000, Proceedings, volume 1776 of *Lecture Notes in Computer Science*, pages 377–386. Springer, 2000.

[Mayr 2003] R. Mayr. *Undecidable problems in unreliable computations.* Theoretical Computer Science, vol. 297, no. 1–3, pages 337–354, 2003.

[McMillan 2003] Kenneth L. McMillan. *Interpolation and SAT-Based Model Checking.* In Computer Aided Verification, 15th International Conference, CAV 2003, Boulder, CO, USA, July 8-12, 2003, Proceedings, volume 2725 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 2003.

[McMillan 2005] K. L. McMillan. *An Interpolating Theorem Prover.* Journal of Theoritical Computer Science, vol. 345, no. 1, pages 101–121, 2005.

[McMillan 2006] K. L. McMillan. *Lazy Abstraction with Interpolants.* In Proc. of 18th Conf. on Computer Aided Verification (CAV'06), volume 4144 of *LNCS*, pages 123–136. Springer, 2006.

[Miné 2001] A. Miné. *A New Numerical Abstract Domain Based on Difference-Bound Matrices.* In Proc. 2nd Symp. Programs as Data Objects (PADO'01), Aarhus, Denmark, May 2001, volume 2053 of *LNCS*, pages 155–172. Springer, 2001.

[Minsky 1967] M. Minsky. Computation, finite and infinite machines. Prentice Hall, 1967.

[Muchnik 1991] A. Muchnik. Definable criterion for definability in presburger arithmetic and its applications. (in russian), preprint, Institute of new technologies, 1991.

[Muchnik 2003] A. A. Muchnik. *The Definable Criterion for Definability in Presburger Arithmetic and its Applications.* Theoretical Computer Science, vol. 290, no. 3, pages 1433–1444, 2003.

[Müller-Olm 2004] M. Müller-Olm and H. Seidl. *A Note on Karr's Algorithm.* In Proc. 31st Int. Coll. on Automata, Languages and Programming (ICALP'04), Turku, Finland, July 2004, LNCS, pages 1016 – 1028. Springer, 2004.

[Oppen 1978] Derek C. Oppen. *A 2^ 2^ 2^ pn Upper Bound on the Complexity of Presburger Arithmetic.* J. Comput. Syst. Sci., vol. 16, no. 3, pages 323–332, 1978.

[Péron 2007] Mathias Péron and Nicolas Halbwachs. *An Abstract Domain Extending Difference-Bound Matrices with Disequality Constraints.* In Verification, Model Checking, and Abstract Interpretation, 8th International Conference, VMCAI 2007, Nice, France, January 14-16, 2007, Proceedings, volume 4349 of *Lecture Notes in Computer Science*, pages 268–282. Springer, 2007.

[Pin 1996] Jean-Eric Pin. *Logic, Semigroups and Automata on Words.* Ann. Math. Artif. Intell., vol. 16, pages 343–384, 1996.

[Presburger 1929] M. Presburger. *Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt.* In Comptes Rendus du Premier Congrès des Mathématiciens des Pays Slaves, pages 92–101, Warsaw, 1929.

[Pudlák 1995] P. Pudlák. *Lower Bounds for Resolution and Cutting Planes Proofs and Monotone Computations.* Journal of Symbolic Logic, vol. 62, no. 3, pages 981–998, 1995.

[Pugh 1992a] William Pugh. *The Omega test: a fast and practical integer programming algorithm for dependence analysis.* Communications of the ACM, vol. 8, pages 102–114, 1992.

[Pugh 1992b] William Pugh and David Wonnacott. *Eliminating False Data Dependences using the Omega Test.* In PLDI, pages 140–151, 1992.

[Rackoff 1978] Charles Rackoff. *The Covering and Boundedness Problems for Vector Addition Systems.* Theor. Comput. Sci., vol. 6, pages 223–231, 1978.

[Reinhardt 2005] K. Reinhardt. *Counting as method, model and task in theoretical computer science.* Habilitation thesis, 2005.

[Reutenauer 1990] Christophe Reutenauer. The mathematics of petri nets. Prentice Hall/Masson, 1990.

[Rybalchenko 2007] Andrey Rybalchenko and Viorica Sofronie-Stokkermans. *Constraint Solving for Interpolation.* In VMCAI, volume 4349 of *LNCS*, pages 346–362. Springer, 2007.

[Rybina 2002] Tatiana Rybina and Andrei Voronkov. *BRAIN: Backward Reachability Analysis with Integers.* In Proc. 9th Int. Conf. Algebraic Methodology and Software Technology (AMAST'2002), Saint-Gilles-les-Bains, Reunion Island, France, Sep. 2002, volume 2422 of *Lecture Notes in Computer Science*, pages 489–494. Springer, 2002.

[Sacerdote 1977] George S. Sacerdote and Richard L. Tenney. *The Decidability of the Reachability Problem for Vector Addition Systems (Preliminary Version).* In Conference Record of the Ninth Annual ACM Symposium on Theory of Computing, 2-4 May 1977, Boulder, Colorado, USA, pages 61–76. ACM, 1977.

[Schnoebelen 2010a] Philippe Schnoebelen. *Lossy Counter Machines Decidability Cheat Sheet.* In Reachability Problems, 4th International Workshop, RP 2010, Brno, Czech Republic, August 28-29, 2010. Proceedings, volume 6227 of *Lecture Notes in Computer Science*, pages 51–75. Springer, 2010.

[Schnoebelen 2010b] Philippe Schnoebelen. *Revisiting Ackermann-Hardness for Lossy Counter Machines and Reset Petri Nets.* In Mathematical Foundations of Computer Science 2010, 35th International Symposium, MFCS 2010, Brno, Czech Republic, August 23-27, 2010. Proceedings, volume 6281 of *Lecture Notes in Computer Science*, pages 616–628. Springer, 2010.

[Schrijver 1987] Alexander Schrijver. Theory of linear and integer programming. John Wiley and Sons, New York, 1987.

[Semenov 1977] A.L. Semenov. *Presburgerness of Predicates Regular in Two Number Systems.* Siberian Mathematical Journal, vol. 18, pages 289–299, 1977.

[Stallings 1983] J. Stallings. *The topology of graphs.* Invent. Math., no. 71, pages 551–565, 1983.

[Su 2004] Z. Su and D. Wagner. *A Class of Polynomially Solvable Range Constraints for Interval Analysis without Widenings and Narrowings.* In Proc. 10th Int. Conf. Tools and Algorithms for the Construction and Analysis of Systems (TACAS'04), Barcelona, Spain, Mar.-Apr. 2004, volume 2988 of *LNCS*, pages 280–295. Springer, 2004.

[T. Gawlitza 2007] H. Seidl T. Gawlitza. *Precise Fixpoint computation Through Strategy Iteration.* In Proc. 16th European Symp. on Programming (ESOP'2007), Braga, Portugal, April 2007, volume 4421 of *LNCS*, pages 300–315. Springer, 2007.

[Taiclin 1968] M.A. Taiclin. *Algorithmic Problems for Commutative Semigroups.* Soviet Math. Doklady, vol. 9, no. 1, pages 201–204, 1968.

[Touikan 2006] Nicholas W. M. Touikan. *A Fast Algorithm for Stallings' Folding Process.* IJAC, vol. 16, no. 6, pages 1031–1046, 2006.

[Valk 1981] R. Valk and G. Vidal-Naquet. *Petri Nets and Regular Languages.* Journal of Computer and System Sciences, vol. 23, no. 3, pages 299–325, 1981.

[Weispfenning 1997] Volker Weispfenning. *Complexity and Uniformity of Elimination in Presburger Arithmetic.* In ISSAC, pages 48–53, 1997.

[Wolper 2000] Pierre Wolper and Bernard Boigelot. *On the Construction of Automata from Linear Arithmetic Constraints.* In Proc. 6th Int. Conf. Tools and Algorithms for the Construction and Analysis of Systems (TACAS'2000), Berlin, Germany, Mar.-Apr. 2000, volume 1785 of *LNCS*, pages 1–19. Springer, 2000.

[Yavuz-Kahveci 2005] Tuba Yavuz-Kahveci, Constantinos Bartzis and Tevfik Bultan. *Action Language Verifier, Extended.* In Proc. 17th Int. Conf. Computer Aided Verification (CAV 2005), Edinburgh, Scotland, UK, July 6-10, 2005, volume 3576 of *Lecture Notes in Computer Science*, pages 413–417. Springer, 2005.

Presburger Counter Machines

Abstract: Critical, embedded or real-time systems have many applications from the control (automotive and avionic industry), to signal processing (audio or video data, GPS navigation systems) and communication (cell phones, internet). Usually these systems are *infinite* since they are based on potentially *unbounded variables*: integer variables (program counters, number of processes connected to a remote server) or real numbers (clocks modeling elapsing time), communication channels, stacks, memory heap, and so on. Whereas the verification of finite state systems is algorithmically decidable (and there exist efficient tools), the verification of infinite state systems is not a sinecure. In this document we introduce frameworks for deciding reachability problems of infinite state systems. The frameworks are illustrated by the Presburger counter machines, a class of machines manipulating a finite set of counters with semantics definable in the first order logic over the natural numbers with the addition, also known as the *Presburger arithmetic*.

Keywords: Verification; Infinite state system; Presburger arithmetic; Automaton; First order logic.