Electronic Notes in Theoretical Computer Science 50 No. 4 (2001) – Proc. VEPAS 2001
URL: http://www.elsevier.nl/locate/entcs/volume50.html 15 pages

Analyzing Automata with Presburger Arithmetic and Uninterpreted Function Symbols

Vlad Rusu [1]

IRISA / INRIA Rennes,
Campus de Beaulieu,
Rennes, France

Elena Zinovieva [2,3]

IRISA / INRIA Rennes,
Campus de Beaulieu,
Rennes, France

Abstract

We study a class of extended automata defined by guarded commands over Presburger arithmetic with uninterpreted functions. On the theoretical side, we show that the bounded reachability problem is decidable in this model. On the practical side, the class is useful for modeling programs with unbounded data structures, and the reachability procedure can be used for symbolic simulation, testing, and verification.

1 Introduction

Modern research in automated verification can be divided into two categories. The first category, which includes languages and tools such as SPIN [12], SMV [3] and CADP [8], focuses on using well-established techniques such as state enumeration or binary decision diagrams, and pushing them to the limits by optimizing the algorithms in every possible way.

The second category deals with exploration of new techniques, whose impact for practical applications, although in quantitative progress, still has to be assessed. Two representative examples for this category are Presburger

[1] Email: rusu@irisa.fr
[2] Email: lenaz@irisa.fr
[3] Supported in part by DYADE action FormalCard, a joint project of INRIA and Bull/CP-8

arithmetic [19] and the theory of uninterpreted functions with equality [1]. These relatively rich theories allow more accurate modeling of real-life systems, avoiding the rough simplifications imposed by finite-state methods. The theories are decidable (although their combination is not) and are efficiently implemented in state-of-the-art verification tools such as PVS [18] and Omega [15].

Motivated by these observations, this paper investigates a class of extended automata that consist of transitions over a finite control structure, with guards and assignments in a decidable fragment of the theory of Presburger arithmetic with uninterpreted functions. The formalism is expressive (*e.g.,* one transition can modify an unbounded number of function values). It can model quite naturally programs with unbounded data structures such as parametric-sized arrays, or communication protocols with unbounded channels carrying unbounded data.

We show that reachability in a bounded number of steps is decidable in this model. The result gives a direct procedure for symbolic simulation. The procedure computes the weakest constraints on a program's input data for a given finite sequence of transitions to be executed. Thus, we obtain an automatic solution to the *test input problem* [16] from software testing: given a finite path in a program, find whether it is executable, and if this is the case, obtain input data for executing it. Then, given a coverage criterion [21] defined by a finite set of finite paths in the program (*e.g.,* executing every instruction at least once, or checking each condition at least once), it is possible, using our procedure, to select the paths that are executable and to synthesize input data for executing them. This results in a complete coverage for the chosen coverage criterion. Finally, symbolic analysis can also be used as a semi-decision procedure for verification of safety properties: if a safety property does not hold, the procedure will detect this in a finite number of steps.

The rest of the paper is organized as follows. In Section 2 we present the theory of Presburger arithmetic with uninterpreted function symbols. In Section 3 we define a class of extended automata (called PF-automata) with guards and assignments in a decidable fragment of this theory. The initial condition may constrain an unbounded number of function values, and assignments may modify an unbounded number of such values. In Section 4 we present a decision procedure for the bounded reachability problem in PF-automata, based on symbolic analysis techniques. The procedure is implemented using the ICS decision procedure package [9] from SRI International. Section 5 presents conclusions, related work, and future work.

2 Background

In this section, we briefly review the theory of Presburger arithmetic and its extension with uninterpreted function symbols.

2.1 Presburger Arithmetic

Let $\mathbb{Z}$ denote the set of integers, and V be a set of integer variables. A *term* is a finite, affine combination on the variables. An *inequality* is a comparison $(<, >, \leq, \geq, =)$ between terms. A *quantifier-free Presburger formula* is a finite Boolean combination of inequalities. A *Presburger formula* is a finite Boolean combination of inequalities, in which some variables can be quantified. Thus, if x, y, z, u are variables, then $x + 2y + 1$ is a term, $x \leq y \wedge x + 2y + 1 > 0$ is a quantifier-free Presburger formula, and $\forall z.x \leq y + z \wedge \exists u.x + 2y + u > y$ is a Presburger formula. Satisfiability in Presburger arithmetic is decidable [19], with time complexity triple exponential in the size of the formula [17] (but simple exponential in the quantifier-free fragment [2]).

2.2 Presburger Arithmetic with Uninterpreted Function Symbols

Again, let V be a set of integer variables, and F be a set of function symbols. For each function $f \in F$, we only know its *arity* (a natural number n), and the fact that it is a function from $\mathbb{Z}^n$ to $\mathbb{Z}$. A *term with function symbols* is either a variable, or an application of a function symbol to n terms, or an affine combination of those. *Inequalities* (*resp. quantifier-free Presburger formulas*, *resp. Presburger formulas*) *with function symbols* are defined as in Section 2.1, except that they are built over terms with function symbols. Quantification over function symbols is not allowed. Thus, if x, y, z, u are variables and f, g are functions of arity one, $x + 2y + f(g(z))$ is a term with function symbols, $x \leq y \wedge x + 2y + f(g(z)) > 0$ is a quantifier-free Presburger formula with function symbols, and $\forall x.x \leq y + f(g(z)) \wedge \exists u.x + 2y + f(z) > g(u)$ is a Presburger formula with function symbols. We denote by PF the theory of Presburger arithmetic with function symbols. Satisfiability in PF is Σ_1^1-complete [13].

2.3 Decidable Fragments

Satisfiability is decidable in the quantifier-free fragment of PF [23]. In this section we build on this result to define a larger decidable fragment.

2.3.1 Decidability of the quantifier-free fragment.

The result [23] is based on a simple observation. In a quantifier-free formula with uninterpreted function symbols, the only relevant property about functions is that they map equals to equals, and, by instantiating this property to finitely many terms, it is possible to obtain an equivalent Presburger arithmetic formula. Let φ be a formula of the quantifier-free fragment of PF. For simplicity, we suppose that in φ there is only one unary function symbol f, which is only applied to two terms, t_1 and t_2. Then, φ is satisfiable if and only

if the following Presburger formula is satisfiable:

$$\varphi' : \quad \varphi[f(t_1)/f_1, f(t_2)/f_2] \wedge (t_1 = t_2 \supset f_1 = f_2) \tag{1}$$

That is, in φ', the function applications $f(t_i)$ from φ are replaced by new integer variables f_i, and the general property that function f maps equals to equals is instantiated to "equality of the terms t_i implies equality of the variables f_i". Indeed, a model for φ trivially induces a model for φ' by choosing $f_1 = f(t_1)$ and $f_2 = f(t_2)$. Conversely, if there exists a model for φ', then a model for φ can be obtained by choosing the value for f such that $f(t_1) = f_1$, $f(t_2) = f_2$, and by letting $f(i)$ be an arbitrary value at any other position than t_1, t_2.

2.3.2 Decidability of the existential fragment.
Consider now the existential fragment of PF (*i.e.,* only existential quantifiers are allowed, and under the scope of an even number of negations). Modulo variable renaming, it is possible to move all quantifiers to the outermost level. Then, a formula $\exists x.\varphi$ has a model if and only if φ also has one. Indeed, if there exist values of the functions and free variables that satisfy φ, then the same values satisfy $\exists x.\varphi$, and if there exists a model for $\exists x.\varphi$, then this model, augmented with the "witness" value of x for the existential quantifier, is a model for φ. Thus, the fragment is decidable. This reasoning can be pushed further: a PF formula φ is satisfiable if and only if $\exists f.\varphi$, where f is a function, is satisfiable. Note that such a formula is not in PF. We use this result in Section 4.

2.3.3 Decidability of the semi-universal fragment.
The universal fragment of PF consists of formulas in which only universal quantifiers are allowed, under an even number of negations. This fragment is highly undecidable [4]. Nevertheless, universal formulas are useful: for example, specifying a property of *all* the elements of a parametric-sized vector requires a universal quantifier. In the remainder of this section we define a class of PF formulas with universal quantifiers for which satisfiability is decidable. This result will be used in the subsequent sections, where we define extended automata with guards and assignments in the class.

Definition 2.1 A *shielded* PF *formula* is a PF formula of the form $\psi : \quad \forall i.\psi'$ where ψ' is a quantifier-free formula with the property that function symbols f may only appear within terms of the form $f(i)$. □

That is, the formula $\forall i.(f(i) > j \supset j > g(i))$ is shielded, but $\forall i.(f(i+1) = 0)$ and $\forall i.(f(i) = f(y))$ are not.

Also, let the *function depth* of a formula ψ be the deepest nesting of function symbols in ψ. That is, both formulas above have function depth 1.

[4] This can be shown, e.g., by encoding the recurrence problem for 2-counter automata.

Definition 2.2 A PF formula ϑ is *semi-universal* if it is a conjunction $\vartheta : \varphi \wedge \psi_1 \wedge \ldots \wedge \psi_k$ of function depth 1, where φ is quantifier-free, every ψ_j $(j = 1, \ldots, k)$ is shielded, and the following semantic property holds: for every model $\mathcal{M}_\varphi$ of φ there exists a model $\mathcal{M}_\psi$ of $\psi : \psi_1 \wedge \ldots \wedge \psi_k$ such that $\mathcal{M}_\varphi$, $\mathcal{M}_\psi$ agree on the values of the free variables that occur in both φ, ψ. □

Example 2.3 Consider the following formula

$$\vartheta : \quad \underbrace{f(x+1) \leq f(y)+1}_{\varphi} \wedge \underbrace{\forall i.(f(i) = y+1)}_{\psi}$$

It is a semi-universal formula, as it satisfies the syntactic conditions of Definition 2.2, *i.e.* has function depth 1, φ is quantifier-free, and ψ is shielded. For the semantic condition, note that for every model $\mathcal{M}_\varphi$ (*e.g.*, Equation 2) there exists also a model $\mathcal{M}_\psi$ (*e.g.*, Equation 3) such that $\mathcal{M}_\varphi$, $\mathcal{M}_\psi$ agree on the value of y, *i.e.*, the free variable that occurs in both φ and ψ.

$$\mathcal{M}_\varphi : \quad \{x = 0, y = 2, f(1) = 3, f(2) = 3, \text{and for all } k \notin \{1, 2\} : \ f(k) = 0\} \tag{2}$$

$$\mathcal{M}_\psi : \quad \{y = 2, \text{for all } k : \ f(k) = 3\} \tag{3}$$

□

Definition 2.2 may be hard to use in practice because of the semantic property that is involved. However, it turns out to be the exact the definition we need in the following sections. A sufficient (and easier to check) condition for a formula to be semi-universal is presented at the end of this section.

Lemma 2.4 *Satisfiability in the class of semi-universal formulas is decidable.*

Proof. Let ϑ be a semi-universal formula. For simplicity, we assume without loss of generality that the quantified part of ϑ consists only of one conjunct. Thus,

$$\vartheta : \quad \varphi \wedge \underbrace{\forall i.\psi_1(i, f_1(i), \ldots, f_n(i))}_{\psi} \tag{4}$$

where $f_1(i), \ldots, f_n(i)$ are all the function applications in ψ. Similarly, let $f(t_1), \ldots, f(t_m)$ be all the function applications in φ. Then, ϑ can be equivalently written as

$$\vartheta' : \quad \underbrace{\varphi \wedge \bigwedge_{i \in \{t_1, \ldots, t_m\}} \psi_1(i, f_1(i), \ldots, f_n(i))}_{\varphi'} \wedge \underbrace{\forall i \notin \{t_1, \ldots, t_m\}.\psi_1(i, f_1(i), \ldots, f_n(i))}_{\psi'} \tag{5}$$

That is, formula ψ is split into two conjuncts: in the first one (which is itself a conjunction), the quantified variable i is instantiated to all of $t_1, \ldots, t_m$, and in the second one, i is required to be different from all these terms. We show that the satisfiability of Formula (5) is equivalent to the satisfiability of its first two conjuncts, *i.e.* φ', which make a quantifier-free formula, whose satisfiability is decidable.

Indeed, any model of Formula (5) is also a model for φ'. Conversely, suppose φ' has a model $\mathcal{M}_{\varphi'}$. Then, $\mathcal{M}_{\varphi'}$ clearly induces a model $\mathcal{M}_{\varphi}$ for φ. As ϑ is semi-universal, this means there exists also a model $\mathcal{M}_{\psi}$ for ψ such that $\mathcal{M}_{\varphi}$ and $\mathcal{M}_{\psi}$ agree on the values of the free variables common to φ and ψ. Now, $\mathcal{M}_{\psi}$ can be extended into a model $\mathcal{M}_{\psi'}$ of ψ': the latter may have more free variables than ψ because of the terms $i \notin \{t_1, \ldots t_m\}$, and we evaluate these supplementary variables according to $\mathcal{M}_{\varphi'}$.

We build a valuation $\mathcal{M}_{\vartheta'}$ for the free variables and functions of Formula (5) as follows. The values of the free variables of φ' (*resp.* ψ') are taken from $\mathcal{M}_{\varphi'}$ (*resp.* $\mathcal{M}_{\psi'}$). The values of the functions f_j $(j = 1 \ldots n)$ are defined as follows: at all positions i defined by the values of $t_1, \ldots, t_m$ in $\mathcal{M}_{\varphi'}$ we let $f_j(i)$ evaluate according to $\mathcal{M}_{\varphi'}$, and at all positions i *different* from the values of $t_1, \ldots, t_m$ in $\mathcal{M}_{\varphi'}$ we let $f_j(i)$ evaluate according to $\mathcal{M}_{\psi'}$.

The construction of $\mathcal{M}_{\vartheta'}$ is sound because we have assumed that φ has function depth 1, thus, $t_1, \ldots, t_m$ depend only on the free variables, and $\mathcal{M}_{\varphi'}$, $\mathcal{M}_{\psi'}$ agree on the values of the free variables common to φ and ψ. We now prove that $\mathcal{M}_{\vartheta'}$ is a model for Formula (5). By construction, $\mathcal{M}_{\vartheta'}$ evaluates the first two conjuncts of (5) just as $\mathcal{M}_{\varphi'}$ does, thus, $\mathcal{M}_{\vartheta'}$ satisfies the first two conjuncts. Similarly, $\mathcal{M}_{\vartheta'}$ evaluates the third conjunct of (5) just as $\mathcal{M}_{\psi'}$ does, hence, $\mathcal{M}_{\vartheta'}$ also satisfies the third conjunct, and the proof is done. □

Let us illustrate the proof of Lemma 2.4 by the following example.

Example 2.5 Consider the semi-universal formula ϑ from Example 2.3. The formula ϑ is equivalently rewritten as:

$$\vartheta' : \quad \underbrace{f(x+1) \leq f(y)+1 \wedge f(x+1) = y+1 \wedge f(y) = y+1}_{\varphi'} \wedge$$
$$\wedge \underbrace{\forall i \notin \{x+1, y\}.(f(i) = y+1)}_{\psi'}$$

We show that satisfiability of ϑ' is equivalent to satisfiability of its quantifier-free part, *i.e.* φ'. Let

$$\mathcal{M}_{\varphi'} : \quad \{x = 0, y = 2, f(1) = 3, f(2) = 3, \text{and for all } k \notin \{1, 2\} : \ f(k) = 0\}$$

be a model for φ'. Here, $\mathcal{M}_{\varphi'}$ is also a model for φ. A corresponding model for ψ, which agrees with $\mathcal{M}_{\varphi'}$ on the values of the common free variables, is

$$\mathcal{M}_{\psi} : \quad \{y = 2, \text{for all } k : \ f(k) = 3\}$$

We extend $\mathcal{M}_\psi$ into a model for $\mathcal{M}_{\psi'}$ of ψ' by choosing $x = 0$ as in $\mathcal{M}_{\varphi'}$, and build a valuation $\mathcal{M}_{\vartheta'}$ as shown in the proof of Lemma 2.4. The values of free variables x and y are chosen from $\mathcal{M}_{\varphi'}$: $x = 0$, $y = 2$. The values of $f(i)$ such that $i = x + 1$ or $i = y$, that is, for $i \in \{1, 2\}$, are also chosen from $\mathcal{M}_{\varphi'}$: $f(1) = 3,\ f(2) = 3$. The values of $f(i)$ such that $i \notin \{1, 2\}$ are chosen from $\mathcal{M}_{\psi'}$, that is, for all $i \notin \{1, 2\} :\ f(i) = 3$. Thus, the valuation $\mathcal{M}_{\vartheta'}$, which is also a model for ϑ, is

$$\mathcal{M}_{\vartheta'} :\quad \{x = 0, y = 2, \text{for all } k :\ f(k) = 3\}$$

□

We now give a simple condition for a formula to be semi-universal.

Definition 2.6 The *universal closure* of a formula ψ is obtained by universally quantifying every free variable in ψ. A formula is *universally satisfiable* if its universal closure is satisfiable. □

Definition 2.7 A *simple semi-universal formula* is a conjunction $\varphi \wedge \forall i_1.\psi_1 \wedge \ldots \wedge \forall i_k.\psi_k$ satisfying the syntactic conditions of Definition 2.2 and the semantic condition that $\psi : \forall i_1.\psi_1 \wedge \ldots \wedge \forall i_k.\psi_k$ is universally satisfiable. □

In a simple semi-universal formula, there exist values of the functions that, together with any values of the variables, constitute a model for the quantified part ψ. In particular, the values of the variables can be chosen from a model $\mathcal{M}$ of the unquantified part φ, thus, the semantic constraints from Definition 2.2 are met. Checking Definition 2.7 is easier than checking Definition 2.2. It can be done, *e.g.,* using theorem proving by providing witness values for the function symbols.

Example 2.8 Consider the following formula

$$\vartheta :\quad \underbrace{f(x+1) \leq f(y) + 1}_{\varphi} \wedge \underbrace{\forall i.(f(i) = i + 1)}_{\psi}$$

It is a simple semi-universal formula as it satisfies the syntactic conditions of Definition 2.7, *i.e.* has functions depth 1, φ is quantifier-free, and ψ is shielded. For the semantic condition, note that ψ coincides with its universal closure and is satisfiable. □

3 PF-Automata

In this section we define the syntax and semantics of a class of extended automata with guards and assignments in the semi-universal fragment of PF.

Definition 3.1 A *variable assignment* to variable x is an expression of the form $x' = t$, where t is a term of PF. □

Definition 3.2 A *function assignment* to function f is a shielded PF formula of function of depth 1, which has the form $\forall i.(e_1(i) \supset f'(i) = e_2(i))$, where e_1 is a quantifier-free formula, e_2 is a term, and f' does not occur in e_1, e_2. □

Definition 3.3 A PF-*automaton* is a tuple $\langle Q, q^0, V, F, \Theta, \mathcal{T} \rangle$:
- $Q = \{1, 2, \ldots, |Q|\}$ is a finite set of *locations*,
- $q^0 \in Q$ is the *initial location*,
- V is a finite set of *integer variables*,
- F is a finite set of unary *function symbols*,
- Θ is a simple semi-universal PF formula, called the *initial condition*,
- $\mathcal{T}$ is a finite set of *transitions*. Each transition is a tuple $\langle q, \gamma, \nu, \phi, q' \rangle$ where
  - $q \in Q$ is the *origin* of the transition,
  - γ is a quantifier-free PF formula of function depth at most 1, called the *guard* of the transition,
  - ν is a finite set of *variable assignments* (*cf.* Definition 3.1). For each variable $v \in V$, there is at most one assignment to v in ν,
  - ϕ is a finite set of *function assignments* (*cf.* Definition 3.2). For each function $f \in F$, there is at most one assignment to f in ϕ,
  - $q' \in Q$ is a location called the *destination* of the transition.

□

Note that the initial condition Θ is required to be a simple semi-universal formula. As membership in this class is not decidable, other techniques (*e.g.,* theorem proving, *cf.* end of Section 2.3) may be needed to establish that a given structure is a PF-automaton. We expect that PF-automata which model "real" programs will have rather simple initial conditions, whose satisfiability is not hard to assess.

PF-automata are useful for modeling programs with unbounded data structures such as files, buffers, and arrays of parametric size. In Definition 3.3, we have assumed that the only basic type is integer, but, of course, other ground types (Booleans, enumerations, records, subranges) can be encoded using integers. The restriction that there is at most one assignment for each variable and function application is useful for avoiding semantic complications (*i.e.,* situations where a function gets two different values simultaneously). It can be dealt with in practice by introducing new transitions to sequentialize the assignments.

Figure 1 is an example of PF-automaton, which models the sorting algorithm of a vector g of parametric size m using a bubble-sorting procedure. Initially, the actual parameters g and m are equal to the formal parameters of the procedure, f and n. Then, f is sorted and copied back into g at the end of the procedure.

In Figure 1, expressions such as $f'(i) = f(j)$ are abbreviations for function assignments of the form, *e.g.,* $\forall k.(k = i \supset f'(k) = f(j))$. The meaning of

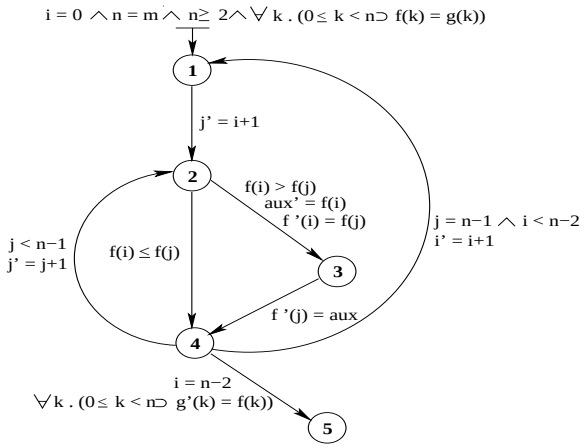

Fig. 1. Example of PF-automaton: Vector Sorting

such an assignment is that the next value of $f(i)$ will be current value of $f(j)$, and the next value of $f(w)$ for $w \neq j$ will remain equal to its current value. By convention, variables that do not appear in assignments also remain unchanged.

Semantics of PF*-automata.*
A *valuation* is a mapping that assigns, to each free variable appearing in the automaton, a value in $\mathbb{Z}$, and to each function symbol, a function from $\mathbb{Z}$ to $\mathbb{Z}$. We denote by $\mathcal{V}$ the set of all valuations. A *state* is a pair (q, v) consisting of a location $q \in Q$ and a valuation $v \in \mathcal{V}$. Note that, for a PF-automaton with at least one function symbol, there is an uncountably infinite set of states. An *initial state* is a state of the form (q^0, v^0) such that $v^0 \models \Theta$, that is, the location is initial and the values of the variables and functions satisfy the initial condition Θ. The set of states is denoted by $\mathcal{S}$, and the set of initial states is denoted by $\mathcal{S}^0$. Each transition $\tau \in \mathcal{T}$ defines a *transition relation* $\varrho_\tau \subseteq \mathcal{S} \times \mathcal{S}$, in the following way. Intuitively, s and s' are in the relation ϱ_τ if the location of s (*resp.* of s') is the origin (*resp.* destination) of τ, and the variables and functions in s satisfy the guard γ of τ. Moreover, the variables and functions get new values according to the assignments of τ.

Formally, for a PF formula φ and a valuation $v \in \mathcal{V}$, let $\varphi[v]$ be the truth value of φ when the free variables and the function symbols of φ evaluate according to v. For a term t, we denote by $t[v]$ the integer value obtained by evaluating variables and function symbols according to v. We now define how valuations are modified by assignments.

According to Definition 3.3, there are two kinds of assignments: variable assignments and function assignments. Let $v' \in \mathcal{V}$ be the valuation obtained from v after an assignment, then, v' is obtained in the following way. If the assignment is of the form $x' = t$, where x is a variable, then v' is the valuation such that for all $u \in F \cup (V \setminus \{x\})$, $v'(u) = v(u)$, and $v'(x) = t[v]$. Otherwise, the assignment is of the form $\forall i.(e_1 \supset f'(i) = e_2)$, where e_1 is a

quantifier-free formula and e_2 is a term, both over the variables $V \cup \{i\}$ and functions F. Then, v' is the valuation such that for all $u \in V \cup (F \setminus \{f\})$, $v'(u) = v(u)$, and $v'(f)$ is defined as follows: for any $i_0 \in \mathbb{Z}$, let $e_1[v, i/i_0]$ (*resp.* $e_2[v, i/i_0]$) denote the value of e_1 (*resp.* e_2) when the free variables V evaluate according to v, and i equals i_0. Then, for any $i_0 \in \mathbb{Z}$ such that $e_1[v, i/i_0]$ holds, $v'(f)(i_0) = e_2[v, i/i_0]$, and $v'(f)(i_0) = v(f)(i_0)$ otherwise.

Finally, for a transition $\tau = \langle q, \gamma, \nu, \phi, q' \rangle$, we denote by $v[\nu, \phi]$ the valuation obtained by successively transforming v according to the assignments of τ. (Note that the order in which this is done is not important, because there is at most one assignment per variable or function). Then, the transition relation ϱ_τ of transition $\tau = \langle q, \gamma, \nu, \phi, q' \rangle$ is the smallest relation defined by the rule

$$\frac{s, s' \in \mathcal{S},\ s = (q, v),\ s' = (q', v'),\ \gamma[v] = true,\ v' = v[\nu, \phi]}{\varrho_\tau(s, s')}$$

We denote the transition relation of the PF-automaton by $\varrho = \bigcup_{\tau \in \mathcal{T}} \varrho_\tau$. A *run* is a sequence of states $\rho : \quad s_0, s_1, \ldots, s_n$ such that $s_0 \in \mathcal{S}^0$, and for $i = 0, \ldots, n-1$, $\varrho(s_i, s_{i+1})$ holds. The *length* of a run $\rho : \quad s_0, s_1, \ldots, s_n$ is n.

4 Symbolic Analysis

We show how to decide reachability of a set of states by a bounded-length run. For a transition τ and an arbitrary predicate Ψ on states, the predicate $post_\tau(\Psi)$ characterizes the states s' that can be reached by taking transition τ from some state s satisfying Ψ :

$$post_\tau(\Psi): \quad \exists s. \varrho_\tau(s, s') \wedge \Psi(s)$$

For a sequence of transitions $\sigma : \quad \tau_1, \ldots, \tau_n$ and a predicate Ψ, the predicate $post_\sigma(\Psi)$ is defined as $post_\sigma(\Psi)$: $post_{\tau_n}(post_{\tau_{n-1}}(\ldots post_{\tau_1}(\Psi)))$. We identify sets of states with the formulas that characterize them, and let the set of states Ω be a quantifier-free PF formula. Clearly, Ω is reachable in at most m steps if and only if there exists a sequence σ of transitions, of length at most m, and starting in the initial location, such that $post_\sigma(\mathcal{S}^0) \cap \Omega \neq \emptyset$, where $\mathcal{S}^0$ is the set of all initial states. Since there are finitely many sequences of transitions up to a given length, it is enough to show that, for any finite sequence of transitions σ starting in the initial location, the formula $post_\sigma(\mathcal{S}^0) \wedge \Omega$ is in a class where satisfiability is decidable.

Lemma 4.1 *For a* PF*-automaton with initial set of states $\mathcal{S}^0$ and any sequence σ of contiguous transitions of the automaton, $post_\sigma(\mathcal{S}^0)$ is of the form*

$$\exists \overline{x}\_, \overline{x}_{2\_}, \ldots, \overline{x}_{(k-1)\_} \exists \overline{f}\_, \overline{f}_{2\_}, \ldots, \overline{f}_{(k-1)\_}. \vartheta(\ \overline{x}, \overline{x}\_, \overline{x}_{2\_}, \ldots, \overline{x}_{(k-1)\_}, \ \overline{f}, \overline{f}\_, \overline{f}_{1\_}, \ldots, \overline{f}_{(k-1)\_}) \tag{6}$$

where ϑ is a semi-universal formula.

Here, $\overline{x}_{\_}, \overline{x}_{2\_}, \ldots, \overline{x}_{(k-1)\_}$ and $\overline{f}_{\_}, \overline{f}_{2\_}, \ldots, \overline{f}_{(k-1)\_}$ (resp. $\overline{x}$ and $\overline{f}$) are the values of free variables and functions on previous steps (resp. on current step) of the calculation of $post_\sigma(\mathcal{S}^0)$.

Proof. We show by induction on the length k of σ that $post_\sigma(\mathcal{S}^0)$ has the form (6).

The base step is obvious: by Definition 3.3 of PF-automata, the initial states $\mathcal{S}^0$ are described by the formula $pc = q_0 \wedge \Theta$, where the initial condition Θ is a semi-universal formula. Thus, the post-image by an empty sequence of the initial states is a formula of the desired form (6).

For the inductive step, consider a transition $\tau = \langle q, \gamma, \nu, \phi, q' \rangle$ of the PF-automaton. For simplicity, we assume without loss of generality that the PF-automaton has only one function symbol f. This means that the function assignments ϕ of transition τ consists of one element, of the form $\forall i.(e_1(i, \overline{x}, f) \supset f'(i) = e_2(i, \overline{x}, f))$. Then, for a state predicate Ψ of the form (6), the predicate $post_\tau(\Psi)$ can be written as Formula (7)

$$\begin{aligned} \exists \overline{x}_{\_}, \ldots, \overline{x}_{k\_} \exists f_{\_}, \ldots, f_{k\_}.(\; & pc = q \wedge \gamma(\overline{x}_{\_}, f_{\_}) \wedge \overline{x} = \nu(\overline{x}_{\_}, f_{\_}) \wedge \\ & \vartheta(\overline{x}_{\_}, \ldots, \overline{x}_{k\_}, f_{\_}, \ldots, f_{k\_}) \wedge \\ & \forall i.(e_1(i, \overline{x}_{\_}, f_{\_}) \supset f(i) = e_2(i, \overline{x}_{\_}, f_{\_}) \wedge \\ & \forall i.(\neg e_1(i, \overline{x}_{\_}, f_{\_}) \supset f(i) = f_{\_}(i))) \end{aligned} \tag{7}$$

(See Example 4.2 for an illustration.) The meaning of Formula (7) is that the next control is at location q, that γ must hold on the values of the variables and the function at the previous step, *i.e.* on $\overline{x}_{\_}$ and $f_{\_}$, and that Ψ must hold on the values of the variables and the function at all previous steps, *i.e.* on $\overline{x}_{\_}, \ldots, \overline{x}_{k\_}$ and $f_{\_}, \ldots, f_{k\_}$. Moreover, the variables are modified by the assignments ν, depending on their previous values and that of the function, and the function is updated at all positions i where e_1 holds of i, $\overline{x}_{\_}$, and $f_{\_}$.

What we still have to show is that the formula obtained from (7) after removing the existential quantifiers is a semi-universal formula. By induction hypothesis, ϑ is a semi-universal formula. Thus, ϑ is a formula of function depth 1, of the form $\varphi \wedge \forall i_1 \psi_1 \wedge \ldots \wedge \forall i_m.\psi_m$, where φ is a quantifier-free PF formula, all formulas $\psi_1, \ldots, \psi_m$ are quantifier-free, and the semantic property of Definition 2.2 holds: for every model $\mathcal{M}_\varphi$ of φ, there exists a model $\mathcal{M}_\psi$ of $\psi : \quad \forall i_1.\psi_1 \wedge \ldots \wedge \forall i_m.\psi_m$ such that the free variables among $\overline{x}_{\_}, \ldots, \overline{x}_{k\_}$ that are common to φ and ψ have the same values in $\mathcal{M}_\varphi$, $\mathcal{M}_\psi$.

Now, it is possible to write the formula obtained from (7) after removing all existential quantifiers, as the conjunction $\varphi' \wedge \psi'$, where $\varphi' : \quad \overline{x} = \nu(\overline{x}_{\_}, f_{\_}) \wedge \varphi$, and $\psi' : \; \forall i_1 \psi_1 \wedge \ldots \wedge \forall i_m.\psi_m \; \wedge \forall i.(e_1(i, \overline{x}_{\_}, f_{\_}) \supset f'(i) = e_2(i, \overline{x}_{\_}, f_{\_}) \wedge \forall i.(\neg e_1(i, \overline{x}_{\_}, f_{\_}) \supset f'(i) = f_{\_}(i)))$. Clearly, φ' is a quantifier-free PF formula, all conjuncts of ψ' are shielded, and all are of function depth 1. To complete the proof, we just have to show that the semantic property of Definition 2.2 holds. By induction hypothesis, assume that $\mathcal{M}_\varphi$ is a model of φ. We build a model $\mathcal{M}_{\varphi'}$ for φ' as follows: the "old" variables $\overline{x}_{\_}, \ldots, \overline{x}_{k\_}$ take their values from $\mathcal{M}_\varphi$, and the "new" variables $\overline{x}$ are defined by the equality $\overline{x} = \nu(\overline{x}_{\_}, f_{\_})$.

We also build a model $\mathcal{M}_{\psi'}$ for ψ' as follows: "old" variables $\overline{x}_\_, \ldots, \overline{x}_{k\_}$ and functions $f_\_, \ldots, f_{k\_}$ take their values from the model $\mathcal{M}_\psi$. Formula ψ' does not refer to the values of the "new" variables $\overline{x}$, and the value of the "new" function f is built as follows: for all values of i, if $e_1(i, \overline{x}_\_, f_\_)$ holds, then the value of $f(i)$ equals that of $e_2(i, \overline{x}_\_, f_\_)$), otherwise the value of $f(i)$ equals that of $f_\_(i)$.

We still have to show that $\mathcal{M}_{\varphi'}$ and $\mathcal{M}_{\psi'}$ agree on the values of their common free variables. But these are just the variables among $\overline{x}_\_, \ldots, \overline{x}_{k\_}$ that were also common to φ and ψ. By induction hypothesis, $\mathcal{M}_\varphi$ and $\mathcal{M}_\psi$ agree on the values of these variables, and, by construction, $\mathcal{M}_{\varphi'}$ agrees with $\mathcal{M}_\varphi$ on all these values, and $\mathcal{M}_{\psi'}$ agrees with $\mathcal{M}_{\varphi'}$. The proof is done. □

Example 4.2 For example, if $\Psi \equiv true$, the predicate $post_\tau(\Psi)$ for the transition τ from location 2 to location 3 of the PF-automaton depicted in Fig 1 is:

$$\begin{aligned}\exists f\_ \exists i\_, j\_, aux\_.(\ & pc = 3 \wedge f\_(i\_) > f\_(j\_) \wedge aux = f\_(i\_) \wedge i = i\_ \wedge \\ & j = j\_ \wedge true \wedge \forall k.(k = i\_ \supset f(k) = f\_(j\_)) \wedge \\ & \forall k.(k \neq i\_ \supset f(k) = f\_(k))\end{aligned}$$

Here, pc is a new integer variable used to encode presence at a given location. For simplicity, we did not include variables m, n and function g, which are not modified by the transition. □

Discussion.
The proof of Lemma 4.1 shows that checking reachability in m steps involves checking satisfiability of a semi-universal formula ϑ with m copies of each variable and function symbol. This, in turn, involves instantiating every universal quantifier from the universal part of ϑ to all the terms in its quantifier-free part (*cf.* Lemma 2.4). Finally, a decision procedure for quantifier-free Presburger arithmetic with uninterpreted function symbols is used to decide the resulting quantifier-free formula. We use the ICS decision procedure package from SRI International [9].

Preliminary results with our symbolic analysis prototype are encouraging: for example, a symbolic simulation of a path of about ten thousand steps in a vector-sorting algorithm was completed in about twenty hours. This means ten thousand calls to the decision procedures for checking formulas with thousands of variables and function applications. As optimizations in both ICS and our prototype are still being developed, we expect to be able in the future to perform symbolic simulation on real-size programs and specifications.

Finally, it worth noting that a simple extension of PF-automata which consists in letting the guards be universal a PF formulas, is too expressive for symbolic simulation, as reachability even in one step becomes highly undecidable (*cf.* Section 2.3).

5 Conclusion, Related Work, and Future Work

In search of new infinite-state models for which some verification problems are still solvable, we investigate in this paper a class of extended automata, called PF-automata, with guards and assignments in a decidable fragment of Presburger arithmetic with uninterpreted function symbols. This formalism allows us to model quite naturally programs with unbounded data structures such as parametric-sized vectors and arrays, or communication protocols with unbounded channels that carry unbounded data. The model is expressive: the initial condition may constrain an unbounded number of function values, and assignments may modify an unbounded number of such values. The latter can be viewed as meta-transitions, which encode in one step the execution of an unbounded number of transitions. We present a decision procedure for the bounded reachability problem in this model. The procedure works by symbolically simulating the initial states over finite sequences of transitions of the automaton. It is implemented in `caml` and uses the ICS decision procedures from SRI International [9]. Symbolic simulation has a number of practical applications:

- it is a useful technique for understanding and debugging programs by interactive execution,
- it can be employed for performing accurate structural testing: given a coverage criterion, which is a finite set of finite paths in the program (as computed by, *e.g.,* a commercial tool [6]), our procedure computes input data for executing those paths that are executable and discards those that are not, resulting in complete coverage for the given criterion, finally,
- it can also be used as a semi-decision procedure for formal verification of safety properties: if a safety property does not hold, our reachability procedure detects this, otherwise, it will loop forever.

In another related work [22] we study the deductive verification of infinite-state systems modeled by PF-automata. We introduce techniques for automatically verifying that a predicate is an inductive invariant in a given context, and identify a class of systems and a logic for expressing invariants and contexts for which the problem is decidable.

Related Work.
There is a large amount of work on the analysis of automata extended with integer variables. See, *e.g.,* [4] for new results and as a good stating point. Concerning automata with uninterpreted function symbols, a recent result [7] shows that simulation is decidable for certain classes of such automata. The only results we are aware of about automata extended with Presburger arithmetic *and* function symbols are reported in [15]. Where, the Omega tool is used to analyze graphs whose edges are annotated by quantifier-free formulas in this logic. A *reachable* operation computes an over-approximation

of the reachable states. The operation will work for graphs with either cycles *or* functions, but not both.

The other main area of related work is structural testing. Most commercial tools (*e.g.,* [6]) can measure the coverage with respect to a given coverage criterion for input data provided by the user. Moreover, the paths in the criterion are not always executable. Research on synthesizing input data using symbolic simulation was started in the seventies [20,5] and has received renewed attention recently [11]. However, these techniques are currently limited to programs with scalar data types (vectors and arrays are not treated). Some techniques for dealing with vectors and arrays have been proposed [14,10]. To our knowledge, our model, which allows an unbounded number of values to be constrained by the initial condition and the assignments, is among the most expressive for which symbolic simulation techniques exist.

Future Work.
The main direction of future work consists in optimizing the prototype symbolic analysis tool and adding new features to it, such as structural path computation for several coverage criteria, analysis of variable dependencies, and slicing. We are also interested in using symbolic analysis techniques for obtaining coverage measures in conformance testing.

References

[1] W. Ackerman. *Solvable Cases of the Decision Problem.* North-Holland Publishing Company, Amsterdam, 1954.

[2] W. Bledsoe. A new method for proving certain Presburger formulas. *In 4th International Joint Conference on Artificial Intelligence*, Tbilissi (USSR), pp. 15–21.

[3] J.R. Burch, E.M. Clarke, K.L. McMillan, D.L. Dill, and J. Hwang. Symbolic model checking: 10^{20} states and beyond. *Information and Computation*, 98(2):142–170, 1992.

[4] T. Bultan, R. Gerber, and W. Pugh. Model checking concurrent systems with unbounded integer variables: symbolic representations, approximations, and experimental results. *ACM Transactions on Programming Languages and Systems*, 21(4):747–789, 1999.

[5] L.A. Clarke. A system to generate test data and symbolically execute programs. *IEEE Transactions on Software Engineering*, 2(3):215–22, 1976.

[6] S. Cornett. Code coverage analysis. Available at `http://www.bullseye.com/coverage.html`

[7] W. Damm, A. Pnueli, and S. Ruah. Herbrand automata for hardware verification. *Conference on Concurrency Theory (CONCUR'98), LNCS 1466*, pp. 67–83, 1998.

[8] J-C. Fernandez, H. Garavel, A. Kerbrat, R. Mateescu, L. Mounier, and M. Sighireanu. CADP: A protocol validation and verification toolbox. *Computer-Aided Verification, CAV'96, LNCS 1102*, pp. 437–440, 1996.

[9] J-C. Filliâtre, S. Owre, H. Rueß, and N. Shankar. ICS: Integrated Canonizer and Solver. *To be presented at CAV'2001.* Available at `http://www.csl.sri.com/papers/cav01/`

[10] A. Goldberg, T-C. Wang, and D. Zimmerman. Applications of feasible path analysis to program testing. *International Symposium on Software Testing and Analysis, ISSTA'94*, pp. 80–94, 1994.

[11] E. Gunter, and D. Peled. Path exploration tool. *Tools and Algorithms for the Construction and Analysis of Systems, TACAS'99, LNCS 1579*, pp. 405–479, 1999.

[12] G.J. Holzmann. *Design and Validation of Communication Protocols.* Prentice Hall, 1991.

[13] J. Halpern. Presburger arithmetic with uninterpreted function symbols is Π_1^1-complete. *Journal of Symbolic Logic*, 56:637–642, 1991.

[14] R. Jasper, M. Brennan, K. Williamson, B. Currier, and D. Zimmerman. Test data generation using feasible path analysis. *International Symposium on Software Testing and Analysis ISSTA'94* pp. 95–107, 1994.

[15] W. Kelly, V. Maslov, W. Pugh, E. Rosser, T. Shpiesman, and D. Wonnacott. The Omega library interface guide. Available at `http://www.cs.umd.edu/projects/omega`.

[16] G.J. Myers. *The Art of Software Testing.* John Wiley and Sons, 1979.

[17] D. Oppen. A $2^{2^{2^{pn}}}$ upper bound on the complexity of Presburger arithmetic. *Journal of Computer and System Sciences*, 16(3):323–332, 1978.

[18] S. Owre, J. Rushby, N. Shankar, and F. von Henke. Formal verification for fault-tolerant architectures: prolegomena to the design of PVS. *IEEE Transactions on Software Engineering*, 21(2):107–125, 1995.

[19] M. Presburger. Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zählen in Welchem die Addition als einzige Operation hervortritt. *Sprawozdanie z I Kongresu Matematykow Krajow Slowcanskich Warszawa*, Poland, pp. 92–101, 1929.

[20] C. Ramomoorthy, S. Ho, and W. Chen. On the automated generation of program test data. *IEEE Transactions on Software Engineering*, 2(4):293–300, 1976.

[21] S. Rapps, and E. Weyuker. Selecting software test data using data flow information. *IEEE Transactions on Software Engineering*, 11(4):367–375, 1985.

[22] V. Rusu. Verifying that invariants are context-inductive. Submitted for publication.

[23] R. Shostak. A practical decision procedure for arithmetic with uninterpreted function symbols. *Journal of the ACM*, 26(2):351–360, 1979.