General Decidability Theorems for Infinite-State Systems

(Extended Abstract)

Parosh Aziz Abdulla* — Uppsala University

Kārlis Čerāns — University of Latvia

Bengt Jonsson — Uppsala University

Yih-Kuen Tsay — National Taiwan University

Abstract

Over the last few years there has been an increasing research effort directed towards the automatic verification of infinite state systems. For different classes of such systems (e.g., hybrid automata, data-independent systems, relational automata, Petri nets, and lossy channel systems) this research has resulted in numerous highly nontrivial algorithms. As the interest in this area increases, it will be important to extract common principles that underly these and related results. This paper is concerned with identifying general mathematical structures which could serve as sufficient conditions for achieving decidability.

We present decidability results for systems which consist of a finite control part operating on an infinite data domain. The data domain is equipped with a well-ordered and well-founded preorder such that the transition system becomes "monotonic" with respect to the preorder, i.e. transitions from larger states lead to larger states. We call the class of systems satisfying these properties *well-structured systems*.

We show that the following properties are decidable for well-structured systems:

- *Reachability:* whether a certain set of control states is reachable. Other properties can be reduced to the reachability problem, notably invariant properties and safety properties represented by the prefix-closed set of traces of a finite automaton.
- *Eventuality:* whether all executions eventually reach a given set of control states (represented as AFp in CTL).
- *Simulation:* The problem of deciding whether there exists a simulation between a finite automaton and a well-structured system. The simulation problem will be shown to be decidable in both directions.

We also describe how these general principles subsume several decidability results from the literature.

*Department of Computer Systems, Uppsala University, P.O.Box 325, 751 05 Uppsala, Sweden, tel: +46 - 18 - 18 31 63, fax: +46 - 18 - 55 02 25, email: `parosh@docs.uu.se`.

1 Introduction

Over the last few years there has been an increasing research effort directed towards the automatic verification of infinite state systems. This has resulted in numerous highly non-trivial algorithms for the verification of different classes of such systems. Examples include timed automata [ACD90, AH89, Č92a], hybrid automata [Hen95], data-independent systems [JP93, Wol86], relational automata ([JJA77, Č92b, Č94]), Petri nets ([Jan90, JM95]), and lossy channel systems [AJ93, AK95]. As the interest in this area increases, it will be important to extract common principles that underly these and related results.

Our goal is to develop general mathematical structures which could serve as sufficient conditions for achieving decidability. Our objective is twofold. We aim on one hand to give a unified explanation of existing decidability results including those mentioned above, and on the other hand to provide guidelines for discovering similar decidability results for other novel classes of systems.

Existing work on general principles for deciding properties of infinite-state systems is rather limited. Most existing methods are based on finding a finite-state system, which is in some simulation or bisimulation relation with the original system. This could be achieved e.g. by partitioning the state space of the original system into equivalence classes under bisimulation [JP93, ACD90, Yi91, Hen95]. However, the requirement of having an appropriate finite partitioning of the state space is very restrictive since it implies that the system under consideration is "essentially finite state".

In this paper we present substantially more general conditions for decidability of several verification problems. We work with a *well-founded preorder* on states instead of an *equivalence*. We consider systems which consist of a finite control part operating on an infinite data domain. The data domain is equipped with a well-founded preorder such that the following properties (which are generalization of those required by finite partitioning methods) hold: (i) the transition system is "monotonic" with respect to the preorder, i.e. transitions from larger states lead to larger states; and (ii) the preorder on the data domain should be *well-ordered*, which means that each ascending chain contains an element which is larger than or equivalent to an earlier element in the chain. We call the class of systems satisfying these properties *well-structured systems*.

In this paper, we show that the following properties are decidable for well-structured systems:

- The problem of whether a certain set of control states is reachable. Other properties can be reduced to the reachability problem, notably invariant properties and safety properties represented by the prefix-closed set of traces of a finite automaton.
- The problem of checking whether all executions eventually reach a given set of control states (represented as AFp in CTL).
- The problem of deciding whether there is a simulation between a finite automaton and a well-structured system. The simulation problem can be decided in both directions.

The reachability problem is solved by a backwards reachability analysis. Starting from a set I of states, the reachability of which is to be decided, we generate the set of states from which I can be reached by a sequence of transitions of length $\leq j$ for successively larger j. The sets that are successively generated in this way are upwards closed wrp. to the preorder and form an ascending chain (under set-inclusion). Since the preorder is well-founded and well-ordered, each set can be represented by a finite set of minimal states, and the chain stabilizes after a finite number of iterations. The problem of whether a well-structured system is simulated by a finite automaton is solved using similar principles. Eventuality properties and the problem of whether a finite automaton is simulated by a well-structured system system are checked by a standard tableau method. Again the tableau construction terminates by the well-ordering property.

The iteration method in this paper can also be viewed as an abstract interpretation of the infinite state space. One contribution is that we show, for well-structured systems, that our abstraction is *exact* for the analysis of the above problems.

Related Work The idea of verifying a system by analyzing a property for an abstraction or simpler approximation of the system has been considered by several authors [CGL92, LGS+95, DGG94]. These works present conditions such that if the property is satisfied by the abstract programs then it will be satisfied by the original program. Sufficient conditions are given for an abstraction to preserve e.g. the branching time logic CTL^* or fragments thereof. However, these works do not give general methods for constructing exact abstractions, and are not concerned primarily with constructing decision procedures for verification as we do.

Compared to finite partitioning methods, we observe that an equivalence is a preorder relation which in addition is symmetric. This means that, apart from systems whose state spaces can be finitely partitioned, e.g. timed automata [ACD90, Č92a], various classes of hybrid automata [Hen95], data-independent systems [JP93], and rational relational automata [Č94], our methods can be used to analyze systems which do not allow for finite partitioning, such as Petri nets [JM95], lossy channel systems [AJ93], and integral relational automata [Č94].

Finkel [Fin90] shows that, for well-structured systems, it is decidable whether a system has a finite reachability tree. He also considers a restricted class of well-structured systems, namely those with *strict monotonicity*. This means that transitions from *strictly* larger states lead to *strictly* larger states. For this class it is shown that the coverability problem, and the problem whether the set of reachable states is finite, are both decidable. The coverability problem is solved in [Fin90] using a generalization of the Karp-Miller algorithm [KM69]. This algorithm depends on strict monotonicity, which does not hold in general for well-structured systems, and hence the Karp-Miller algorithm cannot be applied in this case.

Outline This extended abstract is structured as follows. In the following section, we define infinite-state systems as systems with a finite-state control part which operates on a potentially infinitely large domain of data values. In Section 3 we define well-structured systems. Section 4 presents the method for deciding reachability, Section 5 treats eventuality properties, and Section 6 shows how to check the existence of simulations. In Section 7 we give examples of several classes of transition systems whose elements are well-structured systems.

2 Infinite-State Systems

In this section we give the basic definitions for infinite-state systems. As a general model of such systems, we adopt labeled transition systems. We assume a finite set Λ of *labels*. Each label $\lambda \in \Lambda$ represents an observable interaction with the environment.

Definition 2.1. A (labeled) transition system $\mathcal{L}$ is a pair $\langle S, \delta \rangle$, where

- S is a set of states, formed by the cartesian product of a finite set Q of *control states* and a possibly infinite set D of *data values*, and
- $\delta \subseteq S \times \Lambda \times S$ is the set of allowed transitions.

We use $\langle q, d \rangle$ to denote the state whose control part is q and whose data part is d, and $s \xrightarrow{\lambda} s'$ to denote that $\langle s, \lambda, s' \rangle \in \delta$. Intuitively, $s \xrightarrow{\lambda} s'$ means that the system can move from state s to state s' while performing the observable action λ. We let $s \longrightarrow s'$ denote that there is a λ such that $s \xrightarrow{\lambda} s'$, and let $\xrightarrow{*}$ denote the reflexive transitive closure of $\longrightarrow$.

For $q \in Q$ and $D' \subseteq D$, we use $\langle q, D' \rangle$ to denote the set $\{\langle q, d \rangle \mid d \in D'\}$. For $s \in S$ and $T \subseteq S$ we say that T *is reachable from* s (written $s \xrightarrow{*} T$) if $\exists s' \in T$ such that $s \xrightarrow{*} s'$.

For $T \subseteq S$ and $\lambda \in \Lambda$, we define $pre_\lambda(T)$ to be the set $\left\{ s' \mid \exists s \in T.\ s' \xrightarrow{\lambda} s \right\}$. Analogously, we define $post_\lambda(T)$ to be the set $\left\{ s' \mid \exists s \in T.\ s \xrightarrow{\lambda} s' \right\}$. By $pre(T)$ ($post(T)$) we mean $\cup_{\lambda \in \Lambda} pre_\lambda(T)$ ($\cup_{\lambda \in \Lambda} post_\lambda(T)$). Sometimes we write $pre(s)$ ($post(s)$) instead of $pre(\{s\})$ ($post(\{s\})$).

3 Well-structured Systems

In this section, we define a class of transition systems which we call *well-structured systems*. First, we recall the notion of *preorders*.

3.1 Preliminaries

A *preorder* $\preceq$ is a reflexive and transitive (binary) relation on a set D. By $a \prec b$ we mean $a \preceq b$ and $b \not\preceq a$. We say that $\preceq$ is *decidable* if there is a procedure which, given $a, b \in D$, decides whether $a \preceq b$. We say that $\preceq$ is *well-founded* if there is no infinite sequence $a_1 \succ a_2 \succ a_3 \succ \cdots$. A set M is said to be *canonical* if $a, b \in M$ implies that $a \not\preceq b$. We say that $M \subseteq A$ is a *minor set* of A, if (i) for all $a \in A$ there exists $b \in M$ such that $b \preceq a$, and (ii) M is canonical. It is easy to establish that if $\preceq$ is well-founded, then for each set $A \subseteq D$ there exists at least one (possibly infinite) minor set of A. We use *min* to denote a function which, given a set A, returns a minor set of A.

A set $I \subseteq D$ is an *ideal (in D)* if $a \in I$, $b \in D$, and $a \preceq b$ imply $b \in I$. We define the *(upward) closure* of A, denoted $\mathcal{C}(A)$, as the ideal $\{b \in D \mid \exists a \in A.\ a \preceq b\}$ which is generated by A.

For sets A and B, we say that $A \equiv B$ if $\mathcal{C}(A) = \mathcal{C}(B)$. Observe that $A \equiv B$ if and only if for all $a \in A$ there is a $b \in B$ such that $b \preceq a$, and vice versa.

3.2 Well-structured Systems

To set up our framework we require that the set D of data values is equipped with a decidable well-founded preorder $\preceq$, and assume that we are given a minor set of D which we henceforth call D_{min}. The relation $\preceq$ induces a decidable well-founded preorder on the set S of states defined by $\langle q, d\rangle \preceq \langle q', d'\rangle$ if and only if $q = q'$ and $d \preceq d'$.

A transition system $\langle S, \delta\rangle$ is *monotonic* (with respect to $\preceq$) if for each $s_1, s_2, s_3 \in S$ and $\lambda \in \Lambda$

$$(s_1 \preceq s_2) \ \wedge \ (s_1 \xrightarrow{\lambda} s_3) \quad \supset \quad \exists s_4.\,(s_3 \preceq s_4) \ \wedge \ (s_2 \xrightarrow{\lambda} s_4)$$

A preorder $\preceq$ is said to be a *well-ordering*, if there is no infinite sequence $s_0, s_1, s_2, \ldots$, in which $s_i \not\preceq s_j$ for all $i < j$.

Definition 3.1. A transition system $\mathcal{L} = \langle S, \delta\rangle$, assuming a decidable well-founded relation $\preceq$ on the set D of data values, is said to be *well-structured* if

1. it is monotonic;
2. $\preceq$ is a *well-ordering*; and
3. for each state $s \in S$ and $\lambda \in \Lambda$, the set $min(pre_\lambda(\mathcal{C}(\{s\})))$ is computable.

Note that $min(pre_\lambda(\mathcal{C}(\{s\})))$ is always a finite set if $\preceq$ is well-ordered. We define $premin_\lambda(s)$ as notation for $min(pre_\lambda(\mathcal{C}(\{s\})))$. On the concrete models where we shall apply our theory (Section 7) the establishment of this property will be rather straightforward given the explicit syntactic representations of the transition relations.

Lemma 3.2. *If $\langle S, \delta\rangle$ is a monotonic transition system, and I is an ideal in S, then $pre_\lambda(I)$ and $pre(I)$ are also ideals in S.*

Lemma 3.3. *If $\preceq$ is well-ordered, then each canonical set is finite.*

It follows that every minor set of a set is finite. Notice that there may still be infinitely many minor sets of T.

Lemma 3.4. *For a well-ordered transition system and an infinite sequence $I_0 \subseteq I_1 \subseteq I_2 \subseteq \cdots$ of ideals there is a k such that $I_k = I_{k+1}$*

Proof. Suppose that no such a k exists. It follows that there is a sequence $s_0, s_1, s_2, \ldots$ of states such that $s_k \in I_k$ and $s_k \notin I_j$ for all $j < k$. This means $s_k \not\succeq s_j$ for $j < k$, otherwise $s_k \in I_j$, since I_j is an ideal. This is a contradiction since the sequence $s_0, s_1, s_2, \ldots$ will then violate the well-orderedness assumption. □

4 Control State Reachability

In this section we describe an algorithm to solve the control state reachability problem for well-structured transition systems. More precisely, given a state s and a control state q, we want to check whether $\langle q, D\rangle$ is reachable from s. Our algorithm actually solves the more general problem of deciding whether an ideal I is reachable from a given state s. Since $\langle q, D\rangle$ is an ideal, the control state reachability problem is a special case of the reachability problem for ideals.

To check the reachability of an ideal I, we perform a reachability analysis backwards. Starting from I we define the sequence $I_0, I_1, I_2, \ldots$ of sets by $I_0 = I$ and $I_{j+1} = I \cup pre(I_j)$. Intuitively, I_j denotes the set of states from which I is reachable in j or less steps. Thus, if we define $pre^*(I)$, to be $\cup_{j\geq 0} I_j$, then I is reachable from s if and only if $s \in pre^*(I)$. Notice that $pre^*(I)$ is the least fixpoint $\mu X.\, I \cup pre(X)$. By Lemma 3.2 each I_j is an ideal in S. We know that $I_0 \subseteq I_1 \subseteq I_2 \subseteq \cdots$, and hence from Lemma 3.4 it follows that there is a k such that $I_k = I_{k+1}$. It can easily be seen that $I_\ell = I_k$ for all $\ell \geq k$ implying that $pre^*(I) = I_k$.

Our method for deciding whether I is reachable is based on generating the above sequence $I_0, I_1, I_2, \ldots$ of ideals, and checking for convergence. This cannot be carried out directly since I_j is an infinite set. Instead, we represent each I_j by a canonical set $M_j = min(I_j)$. By Lemma 3.3 each minor set M_j is finite. It is straightforward to show that $M_{j+1} \equiv min(min(I) \cup premin(M_j))$, which is computable as

$$M_{j+1} = min\left(min(I) \cup \bigcup_{s\in M_j} min(pre(\mathcal{C}(\{s\})))\right)$$

since, by the definition of well-structured transition systems, each set $min(pre(\mathcal{C}(\{s\})))$ is computable, and the union is taken over a finite set of sets.

From the above discussion we conclude that if we define $premin^*(min(I))$ to be $\cup_{j\geq 0} M_j$, then there is a k such that $M_{k+1} \equiv M_k$, and $premin^*(min(I)) \equiv M_k$. This implies that $premin^*(M)$ is computable for any minor set M of I and in fact $\mathcal{C}(premin^*(M)) = pre^*(I)$.

Our approach can be explained in terms of abstract interpretation as follows. In the abstract domain of canonical sets, we approximate each ideal I of states by $min(I)$. Conversely, each canonical set M represents the ideal $\mathcal{C}(M)$. The function pre on ideals is translated to the function $premin$ in the abstract domain. This abstraction is now exact in the sense that $\mathcal{C}(premin(M)) = pre(\mathcal{C}(M))$ for each canonical set. Moreover, well-orderedness ensures that the computation of the fixpoint $\mu X.\, I \cup pre(X)$ in the abstract domain converges after a finite number of steps.

Theorem 4.1. *The control state reachability problem is decidable for well-structured systems.*

Proof. Given a state s and a control state q we compute $premin^*(\langle q, D_{min}\rangle)$. We then check whether there is an $s' \in premin^*(\langle q, D_{min}\rangle)$ such that $s' \preceq s$. □

5 Eventuality Properties

In this section we describe an algorithm for deciding whether each execution starting from an initial state eventually reaches a certain control state satisfying a predicate p over control states. In CTL, these properties are of the form AFp. We present an algorithm for the dual property EGp from which an algorithm for AFp can easily be derived. The property EGp is true in a state s_0 iff there is an infinite path from s_0 in which all states have a control part that satisfies p. Our algorithm will actually solve the more general problem of whether s_0 satisfies a property of the form EGI for an ideal I. We write this property as $s_0 \models \mathrm{EG}I$.

The algorithm assumes that the transition system is *recursively finite branching*, meaning that for each state s the set $post_\lambda(s)$ is finite and can be effectively computed from s. In the algorithm, we build a tree labeled by properties of the form $s \models \mathrm{EG}I$. The root node is labeled by $s_0 \models \mathrm{EG}I$. A node labeled by $s \models \mathrm{EG}I$ is a leaf if either

- $s \notin I$. In this case, the node is considered *unsuccessful*, or
- the node has an ancestor labeled $s' \models \mathrm{EG}I$ for some s' with $s' \preceq s$. In this case, the node is considered *successful*.

From a non-leaf node labeled $s \models \mathrm{EG}I$ we create a child labeled $s' \models \mathrm{EG}I$ for each transition $s \longrightarrow s'$ from s. The algorithm answers "yes" if a successful node is encountered, otherwise it answers "no".

The correctness of the algorithm follows from the fact that when a successful node is encountered, we can, by monotonicity, construct an infinite path where all states are in I by continuing from the ancestor node. The construction of the tree terminates, since all branches are finite by well-orderedness of $\preceq$ if the tree is finitely branching (using König's lemma). We have thus proved the following theorem:

Theorem 5.1. *The eventuality problem for control states is decidable for well-structured and recursively finite branching systems.*

6 Simulations between Infinite Systems and Finite Systems

In this section we consider the problem of whether a well-structured system is simulated by a finite transition system. A transition system is said to be *finite* if it has a finite set of states. In our algorithms we assume that a finite transition system is described by finite sets representing states and transitions.

Definition 6.1. Given two transition systems $\mathcal{L}_1 = \langle S_1, \delta_1 \rangle$ and $\mathcal{L}_2 = \langle S_2, \delta_2 \rangle$, we say that a relation $\mathcal{R} \subseteq S_1 \times S_2$ is a *simulation* (of $\mathcal{L}_1$ by $\mathcal{L}_2$) if for each $\langle s_1, s_2 \rangle \in \mathcal{R}$, $s_1' \in S_1$, and $\lambda \in \Lambda$, if $s_1 \xrightarrow{\lambda} s_1'$ then there exists $s_2' \in S_2$ such that $s_2 \xrightarrow{\lambda} s_2'$ and $\langle s_1', s_2' \rangle \in \mathcal{R}$.

For $s_1 \in S_1$ and $s_2 \in S_2$, we say that s_1 is *simulated* by s_2, denoted $s_1 \sqsubseteq s_2$, if there is a simulation $\mathcal{R}$ of $\mathcal{L}_1$ by $\mathcal{L}_2$ such that $\langle s_1, s_2 \rangle \in \mathcal{R}$.

A transition system is said to be *intersection effective* if $min(\mathcal{C}(s_1) \cap \mathcal{C}(s_1))$ is computable for any states s_1 and s_2.

Theorem 6.2. *For a state s in an intersection effective well-structured transition system and a state q in a finite transition system, it is decidable whether $s \sqsubseteq q$.*

Proof. (Sketch) The idea is to calculate the set of pairs $\langle s, q\rangle$ of states such that $s \not\sqsubseteq q$. We observe that for each q, the set $\{s \mid\ s \not\sqsubseteq q\}$ is an ideal. This allows us to compute the set by a fixpoint iteration analogous to that used for the reachability problem. For each state q of the finite transition system, we define a sequence $I_0^q, I_1^q, I_2^q, \ldots$, where $I_0^q = \emptyset$, and $s \in I_{j+1}^q$ iff $s \in I_j^q$, or there are λ and s' such that $s \xrightarrow{\lambda} s'$ and for all q' if $q \xrightarrow{\lambda} q'$ then $s' \in I_j^{q'}$. It is clear that I_j^q is an ideal and that $I_0^q \subseteq I_1^q \subseteq I_2^q \subseteq \cdots$. It follows that there is a k such that $I_{k+1}^q = I_k^q$ for all q, and $s \not\sqsubseteq q$ iff $s \in I_k^q$.

We represent I_j^q by the canonical set $M_j^q = min(I_j^q)$, where $M_0^q = \emptyset$, and

$$M_{j+1}^q = \bigcup_{\lambda} premin_{\lambda}\left(\bigcap_{q' \in post_{\lambda}(q)} M_j^{q'}\right)$$

Note that M_{j+1}^q can be computed from M_j^q for intersection effective well-structured transition systems. We iterate until we reach a k such that $M_{k+1}^q \equiv M_k^q$. To decide whether $s \not\sqsubseteq q$ we check if $\exists s' \preceq s$ such that $s' \in M_k^q$. □

The above result can easily be extended to the case of weak simulation.

We next consider the problem of whether a finite transition system is simulated by a well-structured system. For this we must assume that the well-structured system is recursively finite branching. To determine whether $q_0 \sqsubseteq s_0$, we construct an and-or tree (in analogy with the eventuality problem). The root is labeled by $q_0 \sqsubseteq s_0$. A node labeled by $q \sqsubseteq s$ is a leaf if either

- there is a transition $q \xrightarrow{\lambda} q'$ such that no transition from s is labeled by λ, in which case the node is *unsuccessful*, or
- it has an ancestor labeled $q \sqsubseteq s'$ for some s' with $s \preceq s'$, in which case the node is *successful.*

Each non-leaf node labeled $q \sqsubseteq s$ is an and-node, which for each transition $q \xrightarrow{\lambda} q'$ has a descendant labeled $q' \ll_{\lambda} s$. Each node labeled $q' \ll_{\lambda} s$ is an or-node, which for each transition $s \xrightarrow{\lambda} s'$ has a descendant labeled $q' \sqsubseteq s'$.

By arguments similar to those used in the eventuality algorithm, the and-or tree is always finite. It can be shown that $q_0 \sqsubseteq s_0$ if and only if the root in the and-or tree generated from $q_0 \sqsubseteq s_0$ evaluates to *true* (where unsuccessful leaves evaluate to *false* and successful leaves evaluate to *true*). We have thus proved the following theorem:

Theorem 6.3. *The problem whether $q_0 \sqsubseteq s_0$ for a state q_0 of a finite state system is decidable when s_0 is a state of a well-structured and recursively finite branching system.*

7 Examples

In this section, we survey various classes of well-structured transition systems from the literature. Each system in these classes consists of a finite-state transition system (a *control part*) operating on an infinite domain D. A transition in the control part has a label $\lambda \in \Lambda$ and an operation on D. A transition may also have a condition, which is a predicate on D, and which must hold for the transition to be executed. A state of such a system can be represented as a pair $\langle q, d \rangle$, where q is the state of the control part and $d \in D$.

Lossy Channel Systems *Lossy Channel Systems* (LCSs) are systems of finite-state processes that communicate messages (from a finite alphabet M) over unreliable unbounded FIFO channels. The channels are unreliable in the sense that they may lose messages at any time. LCSs have been used to model and verify data transfer protocols (e.g., sliding window protocols) that are designed to tolerate message losses in channels. The data domain D of an LCS is the set of mappings from channel names to finite sequences of messages. The operations associated with a transition are those of appending a message at the end of a channel, or removing a message from the head of a channel. A transition which receives a message may be executed only if the message is in the head of the channel. Furthermore, any number of messages may be lost inside the channels between executions of transitions.

The preorder $\preceq$ on D is given by $d_1 \preceq d_2$ iff for each channel c, the content of c in d_1 is a (not necessarily contiguous) substring of the content of c in d_1. Monotonicity follows from the fact that a state may lose any number of messages transforming to a smaller state and then perform all the transitions of the smaller state. Well-orderedness holds since the substring relation among strings over a finite alphabet is a well-ordering [Hig52, Lot83]. The rules for computing $premin_\lambda$ may be found in [AJ93]. Intersection effectiveness and recursive finite branching are obvious.

The decidability of control state reachability and eventuality properties is shown in [AJ93], while the decidability of simulation with finite transition systems in both directions is shown in [AK95].

Vector Addition Systems with States (Petri Nets) In a *Vector Addition System with States* (abbreviated *VASS*), the data domain D is the set of mappings from a finite set of variables to the natural numbers. VASSs are very natural extensions of Petri Nets, and have been used for modelling algorithms for resource handling, communication protocols, etc. The operations performed when executing transitions are those of adding integers to the values of the variables. A transition may be executed only if it does not cause any variable to get a negative value.

The preorder $\preceq$ on D is the pointwise ordering on values of variables. For a VASS, monotonicity, intersection effectiveness, and recursive finite branching are obvious. Well-orderedness and the rules for computing $premin_\lambda$ are special cases of those for lossy channel systems.

Control state reachability and eventuality properties for VASSs can also be decided by the Karp-Miller algorithm [KM69]. The control state reachability algorithm we present in this paper performs backward reachability analysis, and can be considered as an alternative to the Karp-Miller algorithm which uses forward reachability analysis. The decidability of simulation with finite transition systems in both directions is shown in [JM95].

Real-Time Automata The data part of a *real-time automaton* consists of the set of mappings from a finite set of *clocks* to the set of nonnegative real numbers. Each transition has a condition, which is a boolean expression built from comparisons between clocks and integer constants. When performed, a transition may reset some of the clocks to 0. Between transitions, the clocks advance at uniform speed. Formally, this is modeled by discrete transitions that advance each clock by the same amount of time. Real-time automata have in recent years become important for modelling and analysis of time-dependent systems.

As our preorder we take the usual equivalence relation on real-time automata, introduced in [ACD90]. This is the largest congruence under predicates of the form $x \geq n$, $x \leq n$, and $x_1 - x_2 \geq n$ for sufficiently small n, where x are clocks.

Monotonicity follows from the fact that transitions from equivalent states lead to equivalent states [ACD90]. The system is well-ordered since there are finitely many equivalence classes. The computation rules for $premin_\lambda$ and intersection effectiveness are obvious.

Relational Automata The data domain of a *relational automaton* operating over the rational numbers (QRA), is the set of mappings from a finite set of variables to the set of rational numbers. The allowed operations of a QRA are: testing the relative orderings of variables and constants, assignments of constants and values of variables to other variables, and input operations. An input operation can be regarded as unbounded choice of an arbitrary new value to be assigned to the corresponding variable. For a QRA, the preorder is defined to be the equivalence relation which equates mappings with the same orderings of the values of the variables and constants.

An IRA is defined in a similar manner to a QRA except that it operates over the set of integers instead of the rational numbers. For an IRA, we define a partial order such that $d_1 \preceq d_2$ iff the relative orderings of the values of the variables and the constants coincide in d_1 and d_2, while the relative differences are smaller in d_1 than d_2. It can be proved [Č94] that an IRA is a well-structured system.

The decidability of the control state reachability and eventuality problems for IRA is shown in [Č94]. The decidability of simulation of an IRA by a finite transition system has not been published before.

References

[ACD90] R. Alur, C. Courcoubetis, and D. Dill. Model-checking for real-time systems. In *Proc. 5^{th} IEEE Int. Symp. on Logic in Computer Science*, pages 414–425, Philadelphia, 1990.

[AH89] R. Alur and T. Henzinger. A really temporal logic. In *Proc. 30^{th} Annual Symp. Foundations of Computer Science*, pages 164–169, 1989.

[AJ93] Parosh Aziz Abdulla and Bengt Jonsson. Verifying programs with unreliable channels. In *Proc. 8^{th} IEEE Int. Symp. on Logic in Computer Science*, 1993. Extended Abstract.

[AK95] Parosh Aziz Abdulla and Mats Kindahl. Decidability of simulation and bisimulation between lossy channel systems and finite state systems. In Lee and Smolka, editors, *Proc.*

CONCUR '95, 6^{th} Int. Conf. on Concurrency Theory, volume 962 of *Lecture Notes in Computer Science*, pages 333 – 347. Springer Verlag, 1995. Extended abstract.

[CGL92] E. M. Clarke, O. Grumberg, and D.E. Long. Model checking and abstraction. In *Proc. 19^{th} ACM Symp. on Principles of Programming Languages*, 1992.

[Č92a] K. Čerāns. Decidability of bisimulation equivalence for parallel timer processes. In *Proc. Workshop on Computer Aided Verification*, volume 663 of *Lecture Notes in Computer Science*, pages 302–315, 1992.

[Č92b] K. Čerāns. Feasibility of finite and infinite paths in data dependent programs. In *LFCS'92*, volume 620 of *Lecture Notes in Computer Science*, pages 69–80, 1992.

[Č94] K. Čerāns. Deciding properties of integral relational automata. In Abiteboul and Shamir, editors, *Proc. ICALP '94*, volume 820 of *Lecture Notes in Computer Science*, pages 35–46. Springer Verlag, 1994.

[DGG94] D. Dams, O. Grumberg, and R. Gerth. Abstract interpretation of reactive systems: Abstractions preserving $\forall$CTL*, $\exists$CTL* and CTL*. In *Proc. IFIP working conference on Programming Concepts, Methods and Calculi (PROCOMET'94)*, 1994.

[Fin90] A. Finkel. Reduction and covering of infinite reachability trees. *Information and Computation*, (89):144–179, 1990.

[Hen95] T.A. Henzinger. Hybrid automata with finite bisimulations. In *Proc. ICALP '95*, 1995.

[Hig52] G. Higman. Ordering by divisibility in abstract algebras. *Proc. London Math. Soc.*, 2:326–336, 1952.

[Jan90] P. Jančar. Decidability of a temporal logic problem for petri nets. *Theoretical Computer Science*, 74:71–93, 1990.

[JJA77] J.M.Barzdin, J.J.Bicevskis, and A.A.Kalninsh. Automatic construction of complete sample systems for program testing. In *IFIP Congress, 1977*, 1977.

[JM95] P. Jančar and F. Moller. Checking regular properties of petri nets. In *Proc. CONCUR '95, 6^{th} Int. Conf. on Concurrency Theory*, pages 348–362, 1995.

[JP93] B. Jonsson and J. Parrow. Deciding bisimulation equivalences for a class of non-finite-state programs. *Information and Computation*, 107(2):272–302, Dec. 1993.

[KM69] R.M. Karp and R.E. Miller. Parallel program schemata. *Journal of Computer and Systems Sciences*, 3(2):147–195, May 1969.

[LGS+95] C. Loiseaux, S. Graf, J. Sifakis, A. Boujjani, and S. Bensalem. Property preserving abstractions for the verification of concurrent systems. *Formal Methods in System Design*, (6):11–44, 1995.

[Lot83] M. Lothaire. *Combinatorics on Words*, volume 17 of *Encyclopedia of Mathematics and its Applications*. Addison-Wesley, 1983.

[Wol86] Pierre Wolper. Expressing interesting properties of programs in propositional temporal logic (extended abstract). In *Proc. 13^{th} ACM Symp. on Principles of Programming Languages*, pages 184–193, Jan. 1986.

[Yi91] Wang Yi. CCS + Time = an interleaving model for real time systems. In Leach Albert, Monien, and Rodriguez Artalejo, editors, *Proc. ICALP '91*, volume 510 of *Lecture Notes in Computer Science*. Springer Verlag, 1991.