Omega-Regular Model Checking*

Bernard Boigelot, Axel Legay, and Pierre Wolper

Université de Liège,
Institut Montefiore, B28,
4000 Liège, Belgium
{boigelot,legay,pw}@montefiore.ulg.ac.be,
http://www.montefiore.ulg.ac.be/~{boigelot,legay,pw}/

Abstract. "Regular model checking" is the name of a family of techniques for analyzing infinite-state systems in which states are represented by words or trees, sets of states by finite automata on these objects, and transitions by finite automata operating on pairs of state encodings, i.e. finite-state transducers. In this context, the central problem is then to compute the iterative closure of a finite-state transducer. This paper addresses the use of regular model-checking like techniques for systems whose states are represented by infinite (omega) words. Its main motivation is to show the feasibility and usefulness of this approach through a combination of the necessary theoretical developments, implementation, and experimentation. The iteration technique that is used is adapted from recent work of the authors on the iteration of finite-word transducers. It proceeds by comparing successive elements of a sequence of approximations of the iteration, detecting an "increment" that is added to move from one approximation to the next, and extrapolating the sequence by allowing arbitrary repetitions of this increment. By restricting oneself to weak deterministic Büchi automata, and using a number of implementation optimizations, examples of significant size can be handled. The proposed transducer iteration technique can just as well be exploited to compute the closure of a given set of states by the transducer iteration, which has proven to be a very effective way of using the technique. Examples such as a leaking gas burner in which time is modeled by real variables have been handled completely within the automata-theoretic setting.

1 Introduction

At the heart of all the techniques that have been proposed for exploring infinite state spaces, is a symbolic representation that can finitely represent infinite sets of states. In early work on the subject, this representation was domain specific, for example linear constraints for sets of real vectors. For several years now, the idea that a generic finite-automaton based representation could

* This work was partially funded by a grant of the "Communauté française de Belgique - Direction de la recherche scientifique - Actions de recherche concertées" and by the European IST-FET project ADVANCE (IST-1999-29082).

be used in many settings has gained ground, starting with systems manipulating queues and integers [WB95,BGWW97,WB98,BRW98], then moving to parametric systems [KMM+97], and, recently, reaching systems using real variables [BBR97,BJW01,BHJ03].

Beyond the necessary symbolic representation, there is also a need to "accelerate" the search through the state space in order to reach, in a finite amount of time, states at unbounded depths. In acceleration techniques, the move has again been from the specific to the generic, the latter approach being often referred to as regular model checking. In regular model checking (see e.g. [BJNT00,DLS01]), the transition relation is represented by a finite-state transducer and acceleration techniques aim at computing the iterative closure of this transducer algorithmically, though necessarily foregoing totality or preciseness, or even both. The advantages of using a generic technique are of course that there is only one method to implement independently of the domain considered, that multidomain situations can potentially be handled transparently, and that the scope of the technique can include cases not handled by specific approaches. Beyond these concrete arguments, one should not forget the elegance of the generic approach, which can be viewed as an indication of its potential, thus justifying a thorough investigation.

So far, generic acceleration techniques have been developed and mostly used for parametric systems, though in [JN00,BLW03] it is shown that they can also be applied to systems with integer variables. Quite naturally, in these cases system states are described by finite words or, for some parametric systems, by finite trees ([AJMd02,BT02]), and the corresponding types of automata are used. On the other hand, the regular model checking approach has not been developed for infinite words, though these are used to represent reals in [BJW01] and [BHJ03], but with domain-specific accelerations. Besides reals, using infinite words to describe states can be useful while checking liveness properties of parametric systems since behaviors violating such properties are necessarily infinite and thus might involve an infinite number of processes.

This paper addresses the problem of using regular model-checking like techniques for systems whose states are represented by infinite (omega) words, hence its title. Its main motivation is to show the feasibility of the approach through a combination of the necessary theoretical developments, implementation, and experimentation. The main features of the techniques developed in this paper are the following. First, to avoid the hard to implement algorithms needed for some operations on infinite-word automata, only omega-regular sets that can be defined by weak deterministic Büchi automata will be considered. This is of course restrictive, but as is shown in [BJW01], it is sufficient to handle sets of reals defined in the first-order theory of linear constraints, leads to algorithms that are very similar to the ones used in the finite-word case, and allows us to work with reduced deterministic automata as a normal form.

Second, taking advantage of this, we lift to weak deterministic Büchi automata the techniques developed in [BLW03], but consider the problem of extrapolating an arbitrary sequence of automata, not just the iterations of a transducer.

This generalization is immediate and moving to omega-words involves only minor technical problems, but has been an opportunity to fine-tune the method. Basically, a sequence of automata is extrapolated by comparing its successive elements, and attempting to identify an "increment" that keeps being added when moving from one element to the next. The acceleration is then obtained by allowing arbitrary repetitions of this increment. The issue of the preciseness of this acceleration is handled with an adapted version of the criterion presented in [BLW03].

Third, we turn to the pragmatics of computing reachable states. Taking advantage of the fact that our extrapolation technique works on automata, not just on transducers, we consider computing reachable states both by computing the closure of the transducer representing the transition relation, and by repeatedly applying the transducer to a set of initial states. The first approach yields a more general object and is essential if one wishes to extend the method to the verification of liveness properties ([BJNT00]), but the second is often less demanding from a computational point of view and can handle cases that are out of reach for the first. Preciseness is not always possible to check when working with state sets rather than transducers, but this just amounts to saying that what is computed is possibly an overapproximation of the set of reachable states, a situation which is known to be pragmatically unproblematic.

Fourth, by implementing and working with examples, we have tuned our approach so that it can handle transducers of substantial size. The most costly step in the computations that are performed is the combination of a transducer with itself, or with an automaton representing a set of states, and the determinization step that is needed after projecting out the intermediate variables. Two tactics have been used to improve the efficiency of this step: using a dominance relation between states as described in [BLW03] and simply doing a bisimulation reduction before applying determinization. The effect of theses tactics can be very substantial. Finally, even though we technically need to work with the reflexive closure of the transducers, some steps can be computed with the nonreflexive transducer. By considering various strategies that exploit this, substantial performance improvements have also been obtained.

Our case studies and experiments include simple linear relations defined over the reals as well as models of hybrid systems (see e.g. [ACH+95]), including a leaking gas burner and an alternating bit protocol with timers. The transducers that are handled contain up to over a thousand states.

2 Preliminaries

An *infinite word* (or *ω-word*) w over an alphabet Σ is a mapping from the natural numbers to Σ. The set of infinite words over Σ is denoted Σ^ω. A *Büchi automaton* is a tuple $(Q, \Sigma, \delta, q_0, F)$, where Q is a set of *states*, $q_0 \in Q$ is the *initial state*, $\delta : Q \times \Sigma \to 2^Q$ is the *transition function* ($\delta : Q \times \Sigma \to Q$ if the automaton is deterministic), and $F \subseteq Q$ is the set of *accepting* states.

A *run* π of a Büchi automaton $A = (Q, \Sigma, \delta, q_0, F)$ on an ω-word w is a mapping $\pi : \mathbb{N} \to Q$ such that $\pi(0) = q_0$, and for all $i \geq 0$, $\pi(i+1) \in \delta(\pi(i), w(i))$ (nondeterministic automaton) or $\pi(i+1) = \delta(\pi(i), w(i))$ (deterministic automaton).

Let $inf(\pi)$ denote the set of states that occur infinitely often in a run π. A run π is said to be *accepting* if $inf(\pi) \cap F \neq \emptyset$. An ω-word w is *accepted* by a Büchi automaton if that automaton admits at least one accepting run on w. The language $L_\omega(A)$ *accepted* by a Büchi automaton A is the set of ω-words it accepts. A language $L \subseteq \Sigma^\omega$ is *omega-regular* if it can be accepted by a Büchi automaton. It is well known that the union and intersection of Büchi automata can be computed efficiently. However, the complement operation requires intricate algorithms that not only are worst-case exponential, but are also hard to implement and optimize.

We now introduce the notion of *weak* automaton [MSS86]. For a Büchi automaton $A = (Q, \Sigma, \delta, q_0, F)$ to be weak, there has to be partition of its state set Q into disjoint subsets $Q_1, \ldots, Q_m$ such that for each of the Q_i, either $Q_i \subseteq F$, or $Q_i \cap F = \emptyset$, and there is a partial order $\leq$ on the sets $Q_1, \ldots, Q_m$ such that for every $q \in Q_i$ and $q' \in Q_j$ for which, for some $a \in \Sigma$, $q' \in \delta(q, a)$ ($q' = \delta(q, a)$ in the deterministic case), $Q_j \leq Q_i$. A weak automaton is thus a Büchi automaton such that each of the strongly connected components of its graph contains either only accepting or only non-accepting states, and transitions leaving a component always move to a lower one.

Not all omega-regular languages can be accepted by weak deterministic Büchi automata, nor even by weak nondeterministic automata. However, there are algorithmic advantages to working with weak automata: weak deterministic automata can be complemented simply by inverting their accepting and non-accepting states; and there exists a simple determinization procedure for weak automata [Saf92], which produces Büchi automata that are deterministic, but generally not weak. Nevertheless, if the represented language can be accepted by a weak deterministic automaton, the result of the determinization procedure will be *inherently weak* according to the definition below [BJW01] and thus easily transformed into a weak automaton.

Definition 1. *A Büchi automaton is* inherently weak *if none of the reachable strongly connected components of its transition graph contain both accepting (visiting at least one accepting state) and non-accepting (not visiting any accepting state) cycles.*

This gives us a pragmatic way of staying within the realm of weak deterministic Büchi automata. We start with sets represented by such automata. This is preserved by union, intersection and complementation operations. If a projection is needed, the result is determinized by the known simple procedure. Then, either the result is inherently weak and we can proceed, or it is not and we are forced to stop. The latter cases might never occur, for instance if we are working with automata representing sets of reals definable in the first-order theory of linear constraints [BJW01]. A final advantage of weak deterministic Büchi automata is that they admit a normal form, which is unique up to isomorphism [Löd01].

3 The Omega Regular Model Checking Framework

We use the following modeling framework:

Definition 2. *A* program *is a triple* $P = (\Sigma, \phi_I, R)$ *where*

- Σ *is a finite alphabet, over which the program configurations are encoded as infinite words;*
- ϕ_I *is a set of initial configurations represented by a weak deterministic automaton over* Σ*;*
- R *is a transition relation represented by a weak deterministic transducer over* Σ *(i.e., a weak automaton over* $\Sigma \times \Sigma$*).*

Using the encoding of reals by infinite words presented in [BBR97,BJW01], this class of programs includes systems using variables ranging over $\mathbb{R}$ and $\mathbb{Z}$ and for which the data operations involving these variables are definable in $\langle \mathbb{R}, \mathbb{Z}, +, \leq \rangle$. Note that, in particular, linear hybrid systems [ACH+95] fall within this category.

Given a program (Σ, ϕ_I, R), we consider two verification problems:

- *Computing the transitive closure of* R*:* The goal is to compute a infinite-word transducer representing the reflexive and transitive closure R^* of R. Such a closure can be used for computing the reachability set $R^*(\phi_I)$ of the program, or for finding cycles between reachable program configurations [BJNT00].
- *Computing the reachable states:* The goal is to compute a Büchi automaton representing $R^*(\phi_I)$, which can be used for checking omega-regular safety properties of the program.

In this paper, we tackle both of these problems by the same approach, which consists in constructing successive approximations of R^* or $R^*(\phi_I)$, and trying to algorithmically construct an approximation of their limit. In certain cases, we will be able to prove that our approximation is exact. In order to show that the approximations to be considered can be selected quite freely, we need the following lemma [BLW03].

Lemma 1. *Let* R *be a relation, and* $R_0 = R \cup Id$*, where* Id *is the identity relation. If* $s_1, s_2, \ldots$ *is an infinite subsequence of the natural numbers, then* $R^* = \bigcup_{k \geq 0} (R_0)^{s_k}$ *and, similarly,* $R^*(\phi_I) = \bigcup_{k \geq 0} (R_0)^{s_k}(\phi_I)$*.*

Thus, given a program (Σ, ϕ_I, R), we compute a transducer T_0 representing $R \cup Id$, and select a *sampling sequence* $s = s_1, s_2, \ldots$ as discussed in [BLW03]. Computing R^* (resp. $R^*(\phi_I)$) then amounts to computing the limit of the sequence of automata $(T_0)^{s_1}$, $(T_0)^{s_2}$, $(T_0)^{s_3}$, ... (resp. $(T_0)^{s_1}(A_{\phi_I})$, $(T_0)^{s_2}(A_{\phi_I})$, $(T_0)^{s_3}(A_{\phi_I})$, ..., where A_{ϕ_I} is an automaton representing ϕ_I). The problem of computing the limit of a sequence of automata is addressed in the next section.

4 Detecting Increments

Consider a sequence $A^1, A^2, A^3, \ldots$ of infinite-word automata, which are assumed to be weak and deterministic. Our goal is to determine whether, for sufficiently large i, the automaton A^{i+1} differs from A^i by some additional constant finite-state structure. Our strategy, borrowed from [BLW03,Leg03], consists in comparing a finite number of successive automata until a suitable increment can be detected.

For each $i > 0$, let $A^i = (Q^i, \Sigma, q_0^i, \delta^i, F^i)$. In order to identify common parts between A^i and A^{i+1}, we consider two equivalence relations between their states:

- The *forward equivalence relation* $E_f^i \subseteq Q^i \times Q^{i+1}$ is such that $(q, q') \in E_f^i$ iff the language accepted from q in A^i is identical to the language accepted from q' in A^{i+1};
- The *backward equivalence relation* $E_b^i \subseteq Q^i \times Q^{i+1}$ is such that $(q, q') \in E_b^i$ iff the finite-word language accepted by A^i with q as final state is identical to the finite-word language accepted by A^{i+1} with q' as final state.

The relations E_f^i and E_b^i can be computed by procedures similar to those developed for finite-word transducers in [BLW03,Leg03], replacing Hopcroft's minimization algorithm for finite-word automata [Hop71] by a variant suited for weak deterministic automata [Löd01].

The relations E_f^i and E_b^i enable us to define our notion of finite-state "increment" between two successive automata.

Definition 3. *The automaton A^{i+1} is* incrementally larger *than A^i if the relations E_f^i and E_b^i cover all the states of A^i. In other words, for each $q \in Q^i$, there must exist $q' \in Q^{i+1}$ such that $(q, q') \in E_f^i \cup E_b^i$.*

Definition 4. *If A^{i+1} is incrementally larger than A^i, then the set Q^i can be partitioned into $\{Q_b^i, Q_f^i\}$, such that*

- *The set Q_f^i contains the states q covered by E_f^i, i.e., for which there exists q' such that $(q, q') \in E_f^i$;*
- *The set Q_b^i contains the remaining states.*

The set Q^{i+1} can now be partitioned into $\{Q_H^{i+1}, Q_{I_0}^{i+1}, Q_T^{i+1}\}$, where

- *The* head part *Q_H^{i+1} is the image by E_b^i of the set Q_b^i;*
- *The* tail part *Q_T^{i+1} is the image by E_f^i of the set Q_f^i, dismissing the states that belong to Q_H^{i+1} (the intention is to have an unmodified head part);*
- *The* increment *$Q_{I_0}^{i+1}$ contains the states that do not belong to either Q_H^{i+1} or Q_T^{i+1}.*

It is worth mentioning that, according to these definitions, the strongly connected components of A^{i+1} are each fully contained in either its head part, tail part, or increment.

Our expectation is that, when moving from one automaton to the next in the sequence, the detected increment will always be the same. We formalize this property by the following definition.

Definition 5. *The sequence of automata* A^i, A^{i+1}, ..., A^{i+k} grows incrementally *if*

- *for each* $j \in [0, k-1]$, A^{i+j+1} *is incrementally larger than* A^{i+j}*;*
- *for each* $j \in [1, k-1]$, *the increment* $Q_{I_0}^{i+j+1}$ *is the image by* E_b^{i+j} *of the increment* $Q_{I_0}^{i+j}$.

Consider a sequence A^i, A^{i+1}, ..., A^{i+k} that grows incrementally. The tail part Q_T^{i+j} of A^{i+j}, $j \in [2, \ldots, k]$, will then consist of $j-1$ copies of the increment plus a part that we will name the *tail-end part*. Precisely, Q_T^{i+j} can be partitioned into $\{Q_{I_1}^{i+j}, Q_{I_2}^{i+k}, \ldots, Q_{I_{j-1}}^{i+j}, Q_{T_f}^{i+j}\}$, where

- for each $\ell \in [1, \ldots, j-1]$, the *tail increment* $Q_{I_\ell}^{i+j}$ is the image by the relation $E_f^{i+j-1} \circ E_f^{i+j-2} \circ \cdots \circ E_f^{i+j-\ell}$ of the "head" increment $Q_{I_0}^{i+j-\ell}$, where "$\circ$" denotes the composition of relations;
- the *tail-end set* $Q_{T_f}^{i+j}$ contains the remaining elements of Q_T^{i+j}.

Our intention is to extrapolate the automaton A^{i+k} by adding more increments, following the same regular pattern as the one detected in the incrementally growing sequence. In order to do this, we need to compare the transitions leaving different increments. We use the following definition [BLW03,Leg03].

Definition 6. *Let* A^{i+k} *be the last automaton of an incrementally growing sequence, let* $Q_{I_0}^{i+k}, \ldots, Q_{I_{k-1}}^{i+k}$ *be the isomorphic increments detected within* T_0^{i+k}*, and let* $Q_{T_f}^{i+k}$ *be its "tail end" set. Then, an increment* $Q_{I_\alpha}^{i+k}$ *is said to be* communication equivalent *to an increment* $Q_{I_\beta}^{i+k}$ *iff, for each pair of corresponding states* (q, q'), $q \in Q_{I_\alpha}^{i+k}$ *and* $q' \in Q_{I_\beta}^{i+k}$, *and* $a \in \Sigma$, *we have that, either*

- $\delta(q, a) \in Q_{I_\alpha}^{i+k}$ *and* $\delta(q', a) \in Q_{I_\beta}^{i+k}$, *hence leading to corresponding states by the existing isomorphism,*
- $\delta(q, a)$ *and* $\delta(q', a)$ *are both undefined,*
- $\delta(q, a)$ *and* $\delta(q', a)$ *both lead to the same state of the tail end* $Q_{T_f}^{i+k}$, *or*
- *there exists some* γ *such that* $\delta(q, a)$ *and* $\delta(q', a)$ *lead to corresponding states of respectively* $Q_{I_{\alpha+\gamma}}^{i+k}$ *and* $Q_{I_{\beta+\gamma}}^{i+k}$.

In order to extrapolate A^{i+k}, i.e., to guess the value of A^{i+k+1}, we simply insert an extra increment $Q_{I_{e_1}}^{i+k}$ between the head part of A^{i+k} and its head increment $Q_{I_0}^{i+k}$ and define the transitions leaving it in order to make it communication equivalent to $Q_{I_0}^{i+k}$. Of course, before doing so, it is heuristically sound to check that a sufficiently long prefix of the increments of A^{i+k} are communication equivalent with each other.

5 An Acceleration Step for Sequences of Weak Automata

Given a sequence of automata, consider an automaton A^{e_0} of this sequence to which the extrapolation step described at the end of the last section can be applied. Its state set Q can be decomposed into a head part Q_H, a series of k increments $Q_{I_0}, \ldots, Q_{I_{k-1}}$, and a tail end part Q_{T_f}. Repeatedly applying the extrapolation step yields a series of extrapolated automata $A^{e_1}, A^{e_2}, \ldots$. Our goal is to build a single automaton that accepts all the words accepted by these automata, i.e., an automaton $A^{e_*} = \bigcup_{i \geq 0} A^{e_i}$. The automaton A^{e_*} can be built from A^{e_0} by the following algorithm.

1. Build an isomorphic copy $A_{I_0\,copy}$ of the automaton formed by the states in Q_{I_0}, the transitions between them, and the outgoing transitions from these states to states in Q_{I_1}, Q_{I_2}, $\ldots$, $Q_{I_{k-1}}$, and Q_{T_f}.
2. Make all the states of $A_{I_0\,copy}$ non-accepting;
3. For each state $q \in Q_{I_0} \cup Q_H$ and $a \in \Sigma$, if $\delta(q, a)$ leads to a state q' in an increment Q_{I_j}, $1 \leq j \leq k-1$, then
   (a) Add transitions labeled by a from q to the state corresponding to q' (by the increment isomorphism) in every increment Q_{I_ℓ} with $1 \leq \ell < j$, as well as to the state corresponding to q' in $A_{I_0\,copy}$;
   (b) If $q \in Q_{I_0}$, then let q_{copy} be the state corresponding to q in $A_{I_0\,copy}$. Add transitions labeled by a from q_{copy} to the state corresponding to q' in every increment Q_{I_ℓ} with $1 \leq \ell < j$, as well as to the state corresponding to q' in $A_{I_0\,copy}$.

These construction rules allow A^{e_*} to simulate the computations of any of the A^{e_i}, $i \geq 0$. Notice that the transitions added at the last step may introduce new cycles from states of $A_{I_0\,copy}$ to themselves (following these cycles amounts to visiting some additional number of increments). Since the accepting runs of the A^{e_i} can only go through a finite number of increments, it is essential to make these cycles non-accepting, which is the reason behind the duplication of the first increment into $A_{I_0\,copy}$.

6 Safety and Preciseness

After having constructed the acceleration A^{e_*} of a sequence A^1, $A^2, \ldots$ of automata, it remains to check whether it accurately corresponds to what we really intend to compute, i.e., $\bigcup_{i>0} A^i$. This is done by first checking that the acceleration is *safe*, in the sense that it captures all behaviors of $\bigcup_{i>0} A^i$, and then checking that it is *precise*, i.e. that it has no more behaviors than $\bigcup_{i>0} A^i$. We check both properties using sufficient conditions. We develop separately these conditions for the two problems outlined in Section 3.

6.1 Transitive Closure of a Relation

Let T_0 be a reflexive infinite-word transducer over the alphabet $\Sigma \times \Sigma$, and let T_0^{e*} be the result of applying the acceleration algorithm described in Section 5 to a sequence $(T_0)^{s_1}, (T_0)^{s_2}, (T_0)^{s_3}, \ldots$ of suitably sampled powers of T_0.

Lemma 2. *The transducer T_0^{e*} is a safe acceleration if $L_\omega(T_0^{e*} \circ T_0^{e*}) \subseteq L_\omega(T_0^{e*})$.*

Indeed, we have $L_\omega(T_0) \subseteq L_\omega(T_0^{e*})$ and thus, by induction, $L_\omega((T_0)^i) \subseteq L_\omega(T_0^{e*})$ (since T_0 is reflexive).

In practice, checking the condition expressed by Lemma 2 requires to complement T_0^{e*}. By construction (see Section 5), this transducer is weak but generally not deterministic. Our approach consists in determinizing T_0^{e*}, and then checking whether the resulting transducer is inherently weak. In the positive case, this transducer can be turned into a weak deterministic one and easily complemented. Otherwise, the test cannot be completed.

We now turn to determining whether the acceleration is precise. This amounts to checking that any word accepted by T_0^{e*} or, equivalently, by some $T_0^{e_i}$, is also accepted by some sampled power $(T_0)^{s_j}$ of the transducer T_0. The idea, borrowed from the procedure developed for finite-word transducers [Leg03,BLW03], is to check that this property can be proved inductively. Our sufficient condition is formalized by the following definition.

Definition 7. *A sequence $T_0^{e_1}, T_0^{e_2}, \ldots$ of extrapolated transducers is* inductively precise *if, for all $i > 1$ and words $w \in L_\omega(T_0^{e_i})$, there exist $0 < j, j' < i$ such that $w \in L_\omega(T_0^{e_j} \circ T_0^{e_{j'}})$.*

To check inductive preciseness, we use weak automata with counters. It is possible to add a counter c to T_0^{e*} in such a way that, when a word $w \in \Sigma \times \Sigma$ is accepted, the value of c stabilizes before reaching the final accepting strongly connected component, and gives the index i of an extrapolated transducer $T_0^{e_i}$ that accepts w (the construction is similar to the one proposed in [Leg03]). The resulting counter automaton is denoted T_c^{e*}. Furthermore, we write $T_{c=i}^{e*}$ (resp. $T_{c<i}^{e*}$) to denote the automaton T_c^{e*} in which the final value of the counter is required to be equal (resp. less than) i in order to accept a word.

Using three copies $T_{c_1}^{e*}$, $T_{c_2}^{e*}$ and $T_{c_3}^{e*}$ of T_0^{e*}, the inductive preciseness criterion can be expressed as

$$\forall w \in (\Sigma \times \Sigma)^\omega, \forall i > 1\, [w \in L_\omega(T_{c_1=i}^{e*}) \supset w \in L_\omega(T_{c_2<i}^{e*} \circ T_{c_3<i}^{e*})]. \qquad (1)$$

This condition can be checked in the following way. It can be shown that, for any word $w \in (\Sigma \times \Sigma)^\omega$ and counter value $i > 0$, w can at most be accepted by one run of $T_{c_1=i}^{e*}$. Moreover, the transitions of $T_{c_1}^{e*}$ are labeled by symbols in $\Sigma \times \Sigma$ and by a counter incrementing operation $+i$, where i is in a finite range $0 \leq i \leq d$. The automaton $T_{c_1}^{e*}$ is, by construction, weak and deterministic with respect to the alphabet $\Sigma \times \Sigma \times [0, d]$ (we denote by $Ta_{c_1}^{e*}$ the automaton $T_{c_1}^{e*}$ considered over this augmented alphabet). Thanks to these properties, deciding Condition (1) reduces to checking

$$L_\omega(Ta_{c_1}^{e_*} \cap (T_{c_2<c_1}^{e_*} \circ T_{c_3<c_1}^{e_*})) = L_\omega(Ta_{c_1}^{e_*}). \quad (2)$$

Since it is sufficient to consider the differences $c_1 - c_2$ and $c_1 - c_3$, we are left with the problem of checking equality of accepted languages between a two-counter weak automaton and a weak deterministic automaton. Following the strategy described in [Leg03,BLW03], we impose synchronization conditions on the pairs of counters (c_1, c_2) and (c_1, c_3) so as to reduce the problem to a finite-state one. It is worth mentioning that, since $Ta_{c_1}^{e_*}$ is weak and deterministic, the automaton accepting the left-hand side of (2) must be inherently weak in order to satisfy the condition. This makes it possible to check easily (2), by first computing minimized deterministic automata accepting both sides [Löd01], and then checking whether their transition graphs are isomorphic.

6.2 Limit of a Sequence of Reachable Sets

Let T_0 be a reflexive infinite-word transducer over the alphabet $\Sigma \times \Sigma$ and let A be an automaton over Σ. Let $A^{e_0}, A^{e_1}, \ldots$ be automata extrapolated from a finite sampled subsequence of A, $T_0(A)$, $(T_0)^2(A)$, $(T_0)^3(A)$, ... Let A^{e_*} be the result produced by the acceleration algorithm described in Section 5 (we thus have $L_\omega(A^{e_*}) = \bigcup_{i \geq 0} L_\omega(A^{e_i})$).

Lemma 3. *The automaton A^{e_*} is a safe acceleration if $L_\omega(T_0(A^{e_*})) \subseteq L_\omega(A^{e_*})$.*

Like in the case of Lemma 2, this condition can be checked provided that determinizing $T_0(A^{e_*})$ produces an inherently weak automaton.

Determining whether the acceleration is precise is a more difficult problem, for which we provide only a partial solution. Our sufficient precision criterion is formalized by the following definition.

Definition 8. *Let $k > 0$. The sequence A^{e_0}, A^{e_1}, ... is* k-inductively precise *if for all $i > 0$ and words $w \in L_\omega(A^{e_i})$, there exists $j < i$ such that $w \in L_\omega(T_0^k(A^{e_j}))$.*

To check k-inductive preciseness, one can use the same strategy as the one outlined in Section 6.1, checking here only the value of the difference between two counters.

In practical applications, the condition expressed by Definition 8 is usually only satisfied when the sequence A, $T_0(A)$, $(T_0)^2(A)$, $(T_0)^3(A)$, ... can be sampled at periodic points (the value of k then corresponds to the chosen period). In other situations, when we are able to assess safety but not preciseness, the accelerated automaton A^{e_*} provides an overapproximation of $\bigcup_{i \geq 0}(T_0)^i(A)$. This can be seen as the result of *widening* the transition relation modeled by the transducer T_0, and is often sufficient for validating the property of interest.

7 Implementation Issues

Implementing the techniques presented in this paper requires potentially costly composition and determinization procedures. In this section, we describe three algorithmic improvements that, in many practical cases, decrease substantially the cost of these procedures.

Dominance. The determinization algorithm for weak automata is derived from the classical subset construction for finite-word automata. As it has been observed in [BLW03], the sets of states that are manipulated during determinization may contain a large number of states that do not influence the behavior of the resulting automaton, because they are *dominated* by other states. Adapting the dominance detection method proposed in [BLW03] to weak deterministic automata is direct and requires only minor algorithmic modifications.

Bisimulation reduction. Consider a transducer T_0 and an exponential sampling sequence $s_i = 2^i$ for all $i > 0$. Any transducer $(T_0)^{s_j}$, with $j > 1$, can easily be computed as the sequential composition of $(T_0)^{s_{j-1}}$ with itself. This composition produces, in general, an automaton with $O(N^2)$ states, with $N = |(T_0)^{s_j}|$, which is costly to determinize. In many experiments, it has been observed that applying a bisimulation reduction to the result of the composition before determinizing it reduces dramatically the cost of computing $(T_0)^{s_j}$.

Nonreflexive composition. Let T be a transducer and $T_0 = T \cup Id$. Consider again an exponential sampling $s_i = 2^i$ for $i > 0$. In experiments, one observes that computing the deterministic form of $(T_0)^{s_j}(A)$, where $j > 1$ and A is some given automaton, is usually much more costly than computing $(T)^{s_j}(A)$. The computation of $(T_0)^{s_j}$ can then be made more efficient by using the formula $(T_0)^{s_j} = ((T_0)^{s_{j-1}} \circ (T)^{s_{j-1}}) \cup (T_0)^{s_{j-1}}$ instead of $(T_0)^{s_j} = (T_0)^{s_{j-1}} \circ (T_0)^{s_{j-1}}$.

8 Experiments

The techniques presented in this paper have been tested on several case studies, using a prototype implementation that relies on the LASH package [LASH] for handling automata. The transition relations that have been considered are all defined over real variables, and represented by *Real Vector Automata* [BBR97], i.e., Büchi automata operating over the binary encoding of numbers.

The first series of test cases consisted of computing the reflexive and transitive closure of relations of the form $(x, x + (1/k))$ for several values of k. In each case, our detection algorithm was able to identify a suitable increment, and applying the acceleration step produced an exact representation of the expected result. The next batch of experiments concerned relations with both discrete and continuous features, such as those of Timed Petri Nets [Bow96]. It is worth mentioning that the disjunctive nature of such relations prevents the use of specific acceleration techniques like those proposed in [BBR97,BHJ03]. For all these case studies, the sampling sequence consisted of the powers of 2. In Table 1, we give the number of states of the minimal and deterministic form of the transducers that were iterated, of the computed closure, and of the largest power of the transducer that had to be constructed in order to detect an increment.

Relation	$\lvert T_0\rvert$	$\lvert T_0^*\rvert$	Max $\lvert T_0^i\rvert$
$(x, x + (1/2))$	9	15	38
$(x, x + (1/4))$	11	18	42
$(x, x + (1/3))$	8	22	53
$(x, x + (1/7)$	10	40	87
$(x, x + (1/17)$	30	84	197
$(x, x + (1/8192))$	33	51	86
$(x, x + (1/65536))$	39	60	98
$\{((x, y), (x + \alpha, y + \alpha)) \mid \alpha \in [0, 0.5],\ x \geq 0,\ y \geq 0\}$			
$\cup \{((x, y), (x, y + 1)) \mid x \geq 0,\ 0 \leq y \leq x - 1\}$	184	309	3500
$[\{((x, y), (x + 1, y - 1))\} \cup \{((x, y), (x - 1, y + 1))\} \cup$			
$\{((x, 0), (x + 1 + \alpha, 0)) \mid \alpha \in [0, 0.5]\}$			
$\cup \{((0, y), (0, y + 1 + \alpha)) \mid \alpha \in [0, 0.5]\}]$			
$\cap\, \mathbb{R}^2_{\geq 0} \times \mathbb{R}^2_{\geq 0}$	82	84	507
$\{((x, y), (x + 2 + \alpha, y - 1 - \alpha)) \mid \alpha \in [0, 0.5]\} \cap \mathbb{R}^2_{\geq 0} \times \mathbb{R}^2_{\geq 0}$	108	596	3944
$[\{((x, y), (x + (1/3), y - (1/2)))\} \cup$			
$\{((x, y), (x - (1/2), y + (1/3)))\}] \cap \mathbb{R}^2_{\geq 0} \times \mathbb{R}^2_{\geq 0}$	336	229	958
$\{((x, y, z), (x + 1 + \alpha, y + 2, z + \alpha)) \mid \alpha \in [0, 0.5]\}$	220	421	44621

Table 1. Examples of transducers modeling relations and their iteration.

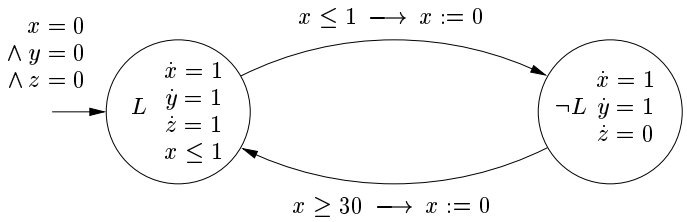

Fig. 1. Hybrid automaton modeling the leaking gas burner.

We also used our prototype for computing reachable sets. For the disjunctive relation $[\{((x,y),(x{+}2{+}\alpha,y{-}1{-}\alpha)) \mid \alpha{\in}[0,0.3]\} \cup \{((x,y),(x{-}1{-}\alpha,y{+}2{+}\alpha)) \mid \alpha{\in}[0,0.3]\}] \cap \mathbb{R}^2_{\geq 0} \times \mathbb{R}^2_{\geq 0}$, the computation of its transitive closure had to be stopped after generating transducers with more than one million states. On the other hand, computing the set of configurations reachable from the state $(1,1)$ resulted in a transducer with only 234 states, which was swiftly produced, the largest automaton build during the procedure having 635 states.

We also applied our method to the more challenging problem of analyzing a linear hybrid automaton. The case study consisted of computing the closure of the transition relation of the *leaking gas burner* described in [ACH+95]. This system consists in a gas burner that leaks during periods of less than one time unit (t.u.), these periods being separated by at least 30 t.u. A linear hybrid automaton modeling the leaking gas burner is given in Figure 1. The states L and $\neg L$ correspond respectively to the leaking and non-leaking situations, x measures the leaking periods and the interval between them, y is absolute time, and z keeps track of the cumulated leaking duration. With our implementation, we were able to compute a superset of its reachable set of configurations. We then compared this set with the one produced by the technique described in [BHJ03] (which is specific to linear hybrid automata), from which it turned out that our computed set was actually exact. For this case study, the minimal and deterministic form of the transition relation is a transducer with 2406 states, and the representation of the reachability set has 524 states. The largest automaton considered by the increment detection procedure had 1892 states.

Our prototype was also applied to several other case studies. Those include an alternating bit protocol with unreliable channels and a timer [BSW69], as well as other disjunctive relations.

9 Conclusions

Attempting to verify infinite-state systems while working exclusively with automata-theoretic representations and algorithms can appear as a somewhat quixotic endeavor. Indeed, working with domain-specific representations will in most cases be more efficient. But, this efficiency is gained at the price of limited applicability, sometimes even within the chosen domain, for instance restricting oneself to convex sets while dealing with constraints. This is to be contrasted with the fact that the automaton-based representation used in this paper has, for instance, the unique capability of algorithmically handling arbitrary combinations of linear constraints involving both reals and integers. Also, a single representation makes the handling of multiple domains direct and, for some domains such as parametric systems, words and automata are the only available representation.

Furthermore, the results of this paper show that accelerating sequences of weak Büchi automata can be done without any specific attention to what the automata actually represent or any, a priori, restriction on their structure. To some extent, this possibility is already present in earlier work, in particular [Tou01],

but making it available for infinite-word automata is unique and very useful from the point of view of the applications that can be handled. Finally, exploiting rather basic algorithmic and implementation optimizations makes it possible to handle automata of significant size. There is certainly room for further improvement and lack of efficiency might, after all, not be a real issue for automata-based approaches.

The leaking gas burner example that has been handled is small, but has long been a challenge for algorithmic approaches and was handled without any specific tailoring of our fairly simple approach. What is really exciting about this direction of work is that so much could be achieved with so little.

References

[ACH+95] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138(1):3–34, 1995.

[AJMd02] P. A. Abdulla, B. Jonsson, P. Mahata, and J. d'Orso. Regular tree model checking. In *Proc. 14th Int. Conf. on Computer Aided Verification*, Lecture Notes in Computer Science. Springer-Verlag, July 2002.

[BBR97] B. Boigelot, L. Bronne, and S. Rassart. An improved reachability analysis method for strongly linear hybrid systems. In *Proceedings of the 9th International Conference on Computer-Aided Verification*, number 1254 in Lecture Notes in Computer Science, pages 167–177, Haifa, Israel, June 1997. Springer-Verlag.

[BGWW97] Bernard Boigelot, Patrice Godefroid, Bernard Willems, and Pierre Wolper. The power of QDDs. In *Proc. of Int. Static Analysis Symposium*, volume 1302 of *Lecture Notes in Computer Science*, pages 172–186, Paris, September 1997. Springer-Verlag.

[BHJ03] B. Boigelot, F. Herbreteau, and S. Jodogne. Hybrid acceleration using real vector automata. In *Proceedings 15th International Conference on Computer Aided Verification (CAV)*, volume 2725 of *Lecture Notes in Computer Science*, pages 193–205, Boulder, USA, 2003. Springer-Verlag.

[BJNT00] A. Bouajjani, B. Jonsson, M. Nilsson, and Tayssir Touili. Regular model checking. In E. A. Emerson and A. P. Sistla, editors, *Proceedings of the 12th International Conference on Computer-Aided Verification (CAV'00)*, volume 1855 of *Lecture Notes in Computer Science*, pages 403–418. Springer-Verlag, 2000.

[BJW01] Bernard Boigelot, Sébastien Jodogne, and Pierre Wolper. On the use of weak automata for deciding linear arithmetic with integer and real variables. In *Proc. International Joint Conference on Automated Reasoning (IJCAR)*, volume 2083 of *Lecture Notes in Computer Science*, pages 611–625, Siena, Italy, June 2001. Springer-Verlag.

[BLW03] Bernard Boigelot, Axel Legay, and Pierre Wolper. Iterating transducers in the large. In *Proc. 15th Int. Conf. on Computer Aided Verification*, volume 2725 of *Lecture Notes in Computer Science*, pages 223–235, Boulder, USA, July 2003. Springer-Verlag.

[Bow96] F. D. J. Bowden. Modelling time in Petri nets. In *Proceedings of the second Australia-Japan Workshop on Stochastic Models in Engineering, Technology and Management*, Gold Coast, Australia, July 1996.

[BRW98] Bernard Boigelot, Stéphane Rassart, and Pierre Wolper. On the expressiveness of real and integer arithmetic automata. In *Proc. 25th Colloq. on Automata, Programming, and Languages (ICALP)*, volume 1443 of *Lecture Notes in Computer Science*, pages 152–163. Springer-Verlag, July 1998.

[BSW69] Bartlett, K. A., Scantlebury, R. A., and Wilkinson, P. T. A note on reliable full–duplex transmission over half–duplex links. *Communications of the ACM*, 12(5):260–261, May 1969.

[BT02] A. Bouajjani and T. Touili. Extrapolating tree transformations. In *Proc. 14th Int. Conf. on Computer Aided Verification*, Lecture Notes in Computer Science. Springer-Verlag, July 2002.

[DLS01] D. Dams, Y. Lakhnech, and M. Steffen. Iterating transducers. In *Proceedings 13th International Conference on Computer Aided Verification (CAV)*, volume 2102 of *Lecture Notes in Computer Science*, pages 286–297, Paris, France, 2001. Springer.

[Hop71] J. E. Hopcroft. An $n \log n$ algorithm for minimizing states in a finite automaton. *Theory of Machines and Computation*, pages 189–196, 1971.

[JN00] B. Jonsson and M. Nilson. Transitive closures of regular relations for verifying infinite-state systems. In S. Graf and M. Schwartzbach, editors, *Proceeding of the 6th International conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'00)*, volume 1875 of *Lecture Notes in Computer Science*, pages 220–234. Springer, 2000.

[KMM+97] Y. Kesten, O. Maler, M. Marcus, A. Pnueli, and E. Shahar. Symbolic model checking with rich assertional languages. In *Proceedings of 9th International Conference on Computer-Aided Verification (CAV'97)*, volume 1254 of *Lecture Notes in Computer Science*, pages 424–435. Springer, 1997.

[LASH] The Liège Automata-based Symbolic Handler (LASH). Available at `http://www.montefiore.ulg.ac.be/~boigelot/research/lash/`.

[Leg03] A. Legay. Iterating transducers: from experiments to concepts and algorithms. Master's thesis, University of Liège, 2003.

[Löd01] C. Löding. Efficient minimization of deterministic weak $\omega-$automata. *Information Processing Letters*, 79(3):105–109, 2001.

[MSS86] D. E. Muller, A. Saoudi, and P. E. Schupp. Alternating automata, the weak monadic theory of the tree and its complexity. In *Proc. 13th Int. Colloquium on Automata, Languages and Programming*, pages 275–283, Rennes, 1986. Springer-Verlag.

[Saf92] S. Safra. Exponential determinization for ω-automata with strong-fairness acceptance condition. In *Proceedings of the 24th ACM Symposium on Theory of Computing*, Victoria, May 1992.

[Tou01] T. Touili. Regular model checking using widening techniques. In *Proceeding of Workshop on Verification of Parametrized Systems (VEPAS'01)*, volume 50 of *Electronic Notes in Theoretical Computer Science*, 2001.

[WB95] Pierre Wolper and Bernard Boigelot. An automata-theoretic approach to Presburger arithmetic constraints. In *Proc. Static Analysis Symposium*, volume 983 of *Lecture Notes in Computer Science*, pages 21–32, Glasgow, September 1995. Springer-Verlag.

[WB98] Pierre Wolper and Bernard Boigelot. Verifying systems with infinite but regular state spaces. In *Proc. 10th Int. Conf. on Computer Aided Verification*, volume 1427 of *Lecture Notes in Computer Science*, pages 88–97, Vancouver, July 1998. Springer-Verlag.