Languages, Rewriting systems, and Verification of Infinite-State Systems *

Ahmed Bouajjani

Liafa laboratory, University of Paris 7,
Case 7014, 2 place Jussieu, F-75251 Paris Cedex 05, France.
Email: abou@liafa.jussieu.fr

1 Introduction

Verification of complex systems cannot be achieved without combining several analysis methods and techniques. A widely adopted approach consists in combining abstraction methods with algorithmic verification techniques. Typically, finite abstractions are built using automatic or semi-automatic methods and model-checking algorithms are applied on these abstractions in order to verify the desired properties. However, finding faithful finite abstractions is often hard since many aspects in the behavior of the system must be hidden or encoded in a nontrivial and ad-hoc manner. This is particularly true for software systems since their behavior depends in a very crucial manner on the manipulation of data structures and variables which are assumed to range over infinite domains (e.g., unbounded stacks, queues, arrays, counters), or over finite domains whose sizes are left as parameters. Moreover, many systems are defined as networks of parametric size, i.e., they are assumed to work for an arbitrary number of processes running in parallel.

Hence, there is a real need (1) to define models allowing to capture essential aspects which are beyond the expressive power of finite models (e.g., manipulation of unbounded variables, parametrization), and (2) to develop algorithmic verification techniques which can be applied to these infinite-state models.

In this paper, we consider models based on rewriting systems and we develop an algorithmic approach for analyzing automatically such models. In the framework we adopt, configurations of systems are seen as words or vectors of words, and actions are represented by means of rewriting rules. Different rewriting policies can be considered, e.g., prefix, cyclic, or factor rewriting. They allow to model different classes of systems, e.g., pushdown systems, communicating systems through FIFO channels, or parametrized networks of identical finite-state processes connected according to a linear topology.

Then, the main problem we address is the problem of computing a representation of the infinite set of all reachable configurations in a model. Solving this problem is indeed the kernel of most of the verification methods. In our setting, this problem relies on computing the closure of a language by a rewriting system,

* This work was supported in part by the European Commission (FET project ADVANCE, contract No IST-1999-29082).

i.e., given a rewriting system R and a language ϕ, compute $R^*(\phi)$, where R^* is the reflexive-transitive closure of the relation induced by R. We present several closure results concerning different classes of languages and rewriting systems, and we show the applications of these results in symbolic reachability analysis of different infinite-state systems. The results we present in this paper are not new. Our aim here is to present a general approach for algorithmic verification of infinite-state systems, and to show in a simple and uniform manner several results we have established in the last few years.

The paper is organized as follows: In Section 2 we present the principle of a general algorithmic verification approach based on symbolic reachability analysis. In Section 3 we define classes of rewriting systems and show their use as models for various kinds of infinite-state systems. In Section 4 we presents results on the computability of the closure of languages by rewriting systems, and show their relevance in verification. Finally, in Section 5, we give a presentation of related work.

2 Symbolic Reachability Analysis

A system can be modeled as a pair $(\mathcal{C}, \rho)$ where $\mathcal{C}$ is the set of all possible *configurations* of the system, and $\rho \subseteq \mathcal{C} \times \mathcal{C}$ is a binary *transition relation* between configurations.

Given a relation ρ, let us denote by ρ^i the relation obtained by i compositions of ρ, i.e., ρ^0 is the identity relation, and for $i \geq 0$, $\rho^{i+1} = \rho^i \circ \rho$. Then, let ρ^* be the reflexive-transitive closure of the relation ρ, i.e., $\rho^* = \bigcup_{i \geq 0} \rho^i$.

Given a relation ρ and a configuration γ, let $\rho(\gamma) = \{\gamma' \in \mathcal{C} \ : \ (\gamma, \gamma') \in \rho\}$. Intuitively, $\rho(\gamma)$ is the set of all immediate successors of the configuration γ, and $\rho^*(\gamma)$ is the set of all reachable configurations from γ. These definitions can be generalized straightforwardly to sets of configurations.

Verification problems, especially for safety properties, can be often reduced to the reachability analysis problem, i.e., to computing the set of all reachable configurations starting from a given (possibly infinite) set of *initial* configurations $\phi \subseteq \mathcal{C}$. In our setting, this consists in computing the set $\rho^*(\phi)$. More precisely, the problem is to construct a finite representation of the set $\rho^*(\phi)$, given a finite representation of the set ϕ. Then, the central problem we address can be stated as follows:

Π : *identify classes of recursive binary relations $\mathcal{R}$ between configurations as well as classes of finite representation structures $\mathcal{S}_1$ and $\mathcal{S}_2$ corresponding to two classes of sets of configurations, such that for every effectively $\mathcal{S}_1$-representable set ϕ and every relation $\rho \in \mathcal{R}$, the set $\rho^*(\phi)$ is effectively $\mathcal{S}_2$-representable.*

In order to be relevant to system verification, this problem must be addressed for classes of representation structures enjoying some minimal closure and decidability properties (e.g., the decidability of the inclusion test). Often, it is interesting to consider a stronger version of the problem above, where we require that $\mathcal{S}_1$ and $\mathcal{S}_2$ are the same class. Of course, few classes of models for practical infinite-state systems have a decidable reachability problem. Hence, it is clear that the verification problem of infinite-state systems cannot be reduced in general to finding a solution to the problem (Π). However, solutions to the problem (Π) can be embedded in a more general (or more pragmatic) approach in order to tackle classes of infinite-state systems with an undecidable reachability problem. The idea is the following one: if we cannot provide an algorithm for computing directly the set $\rho^*(\phi)$, we adopt a semi-algorithmic approach (i.e., termination is not guaranteed) based on an iterative exploration of the reachable configurations. In order to speed up this procedure and help its termination, we use within this exploration solutions for the problem (Π) concerning subrelations of ρ. That is, at each iteration we compute the ρ-image of the reachable configurations found so far, as well as, when possible, their images by *transitive closures* of some (compositions of) subrelations of ρ. Hence, solutions for the problem (Π), even if they concern restricted classes of relations, can be relevant for enhancing the iterative reachability analysis procedure, provided that the computed intermediate sets belong to the same class of representation structures.

Let us see this in more details. From the definition of the set $\rho^*(\phi)$, a procedure for computing it would be to construct iteratively the non-decreasing sequence of sets $\phi_i = \bigcup_{0 \leq j \leq i} \rho^j(\phi)$, for $i \geq 0$, until $\phi_{k+1} \subseteq \phi_k$ for some $k \geq 0$. In such a case, we have necessarily $\phi_k = \rho^*(\phi)$. Actually, the sequence $(\phi_i)_{i \geq 0}$ can be computed by taking

$$\begin{aligned} \phi_0 &= \phi \\ \phi_{i+1} &= \phi_i \cup \rho(\phi_i) \end{aligned}$$

Of course, in order to be able to apply this procedure, we need a class of representation structures $\mathcal{S}$ such that: (1) ϕ is $\mathcal{S}$-representable, (2) $\mathcal{S}$ is effectively closed under union and computing the ρ-image, and (3) the inclusion test is decidable in $\mathcal{S}$.

However, it is obvious that this naive procedure does not terminate in general (in all nontrivial cases where $\mathcal{C}$ is infinite). Therefore, we enhance this procedure using a *fixpoint acceleration* technique, according to the terminology used in the abstract interpretation community [CC77].

Let us first introduce some notation. Given a finite set of relations $\mathcal{R}$, we denote by $Comp(\mathcal{R})$ the smallest set of relations which contains $\mathcal{R}$ and which is closed under the operations of union ($\cup$) and composition ($\circ$) of relations.

Now, let ρ be a relation and ϕ be a set of initial configurations representable in a class of representation structure $\mathcal{S}$. A new reachability analysis procedure for computing $\rho^*(\phi)$ can be defined by considering a decomposition

$$\rho = \rho' \cup \rho_1 \cup \ldots \cup \rho_m$$

and by defining a finite set of relations $\theta_1, \ldots, \theta_n \in Comp(\{\rho_1, \ldots, \rho_m\})$ such that it is possible to compute and to represent in $\mathcal{S}$ the set $\theta_i^*(\psi)$, for each

$i \in \{1, \ldots, n\}$ and for every ψ in the class $\mathcal{S}$. Typically, the decomposition of ρ we consider is extracted from its definition as a finite union of relations, each of them corresponding to a possible action (or set of actions) in the modeled system, and very often, the θ_i's can be chosen to be the ρ_i's themselves.

Then, the new iterative procedure we apply consists in computing the sequence of non-decreasing sets $(\psi_i)_{i \geq 0}$ defined as follows:

$$\psi_0 = \phi$$
$$\psi_{i+1} = \psi_i \cup \rho'(\psi_i) \cup \theta_1^*(\psi_i) \cup \ldots \cup \theta_n^*(\psi_i)$$

until $\psi_{k+1} \subseteq \psi_k$ for some $k \geq 0$. Clearly, we have for every $i \geq 0$, $\phi_i \subseteq \psi_i$ and $\psi_i \subseteq \bigcup_{i \geq 0} \rho^i(\phi) = \rho^*(\phi)$. This means that, if this procedure stops, it computes precisely an $\mathcal{S}$-representation of the set $\rho^*(\phi)$.

The procedure described above generates the set of reachable configurations according to a breadth first search strategy, using additional transitions called *meta-transitions* (as in [BW94]), each of them corresponding to iterating an arbitrary number of times the application of a transition relation θ_i, for $i \in \{1, \ldots, n\}$. Actually, different search strategies may be adopted for generating the set of reachable configurations, e.g., a depth first search strategy with priority to meta-transitions.

Of course, the method proposed above does not guarantee termination. The reachability analysis procedure terminates only if we can define a suitable finite set of meta-transitions. This is obviously related to our ability to find solutions to the problem (Π) stated at the beginning of this section.

3 Models Based on Rewriting systems

We consider here models which correspond to systems operating on sequential data structures (such as stacks or queues). In these models, configurations are vectors of words, and transition relations between configurations are defined by means of sets of rewriting rules.

3.1 Rewriting systems

Let Σ be a finite alphabet (set of symbols). For $n \geq 1$, an n-dim *rewriting rule* r over Σ is a pair $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ where $\boldsymbol{x}, \boldsymbol{y} \in (\Sigma^*)^n$. We denote such a rule by $r: \boldsymbol{x} \mapsto \boldsymbol{y}$. The *left hand side* (resp. *right hand side*) of r, denoted by $lhs(r)$ (resp. $rhs(r)$), is the vector $\boldsymbol{x}$ (resp. $\boldsymbol{y}$).

A n-dim *rewriting system* is a finite set of n-dim rewriting rules. We consider hereafter three notions of *rewriting relations* between vectors of words. Given an n-dim rewriting system R, the prefix (resp. cyclic, factor) rewriting relation associated with R is the relation R_p (resp. R_c, R_f) $\subseteq (\Sigma^*)^n \times (\Sigma^*)^n$ such that for every $\boldsymbol{u} = (u_1, \ldots, u_n), \boldsymbol{v} = (v_1, \ldots, v_n) \in (\Sigma^*)^n$, $(\boldsymbol{u}, \boldsymbol{v}) \in R_p$ (resp. R_c, R_f) if and only if there exists a rule $r : (x_1, \ldots, x_n) \mapsto (y_1, \ldots, y_n) \in R$ such that for every $i \in \{1, \ldots, n\}$, we have respectively,

(*Prefix rewriting*) $\exists w_i \in \Sigma^*.\ u_i = x_i w_i$ and $v_i = y_i w_i$,
(*Cyclic rewriting*) $\exists w_i \in \Sigma^*.\ u_i = x_i w_i$ and $v_i = w_i y_i$,
(*Factor rewriting*) $\exists w_i, w_i' \in \Sigma^*.\ u_i = w_i x_i w_i'$ and $v_i = w_i y_i w_i'$.

3.2 Models of infinite-state systems

The models we consider are defined as pairs $(\mathcal{C}, \rho)$ where the set of configurations is $\mathcal{C} = (\Sigma^*)^n$, and the transition relation ρ is one of the rewriting relations $R_\dagger$, with $\dagger \in \{p, c, f\}$, for some given rewriting system R.

Automata with unbounded sequential data structures: The very common models of pushdown systems and FIFO-channel systems can be straightforwardly represented in our framework.

Indeed, prefix rewriting models the actions of a system manipulating pushdown stacks, and cyclic rewriting corresponds to operations on FIFO queues (or communication channels). One additional dimension in the rewriting system can be used to encode the control states.

For instance, consider a system which manipulates one pushdown stack (resp. one FIFO queue). A rule $r : (a, x) \mapsto (b, y)$ where $a, b \in \Sigma$ and $x, y \in \Sigma^*$, represents the action of (1) testing whether the sequence of symbols x can be removed from the stack (resp. the queue), and if yes, (2) moving from the control state a to the control state b, and putting the sequence y into the stack (resp. the queue) after having removed x from it.

In the sequel, we call an n-dim *controlled rewriting system* any set of $(n+1)$-dim rules of the form $(a, \boldsymbol{x}) \mapsto (b, \boldsymbol{y})$ where $a, b \in \Sigma$ and $\boldsymbol{x}, \boldsymbol{y} \in (\Sigma^*)^n$.

Parametrized networks: We use factor rewriting for modelling parametrized systems with an arbitrary number of identical finite-state components (processes), connected according to a linear topology.

Let Σ be the finite set of states of each of these components. Then, in order to reason uniformly about the family of systems with arbitrary number of components, we consider that a configuration is a finite word over Σ, the i^{th} element of the word corresponding to the state of the i^{th} component, and various classes of languages (e.g., regular languages) can be used to represent sets of configurations of arbitrary lengths. Therefore, actions of such parametrized systems can be represented naturally as rewriting rules, each of them corresponding to simultaneous moves in some components of the system. The kind of rules we consider here allow to represent moves involving a finite number of processes located within a bounded distance from each other. Typically, communication (rendez-vous) between immediate neighbors can be modeled by rewriting rules of the form $ab \mapsto cd$, meaning that if two processes i and $i+1$ are in states a and b respectively, then they can move simultaneously to their respective states c and d.

Take as an example a simple version of the token-passing mutual exclusion protocol: We assume that processes are arranged linearly. A process who owns the right to enter the critical section (the token) can transmit it to its right neighbor. Each process has two possible states, either 1 if it owns the token, or 0 otherwise. We suppose that initial configurations are all those in which the leftmost process has the token. Since the number of processes is not fixed, the set of initial configurations is precisely the language 10^*. Then, the transition relation between configurations, which models the action of passing the token from left to right, corresponds to the relation R_f, where $R = \{10 \mapsto 01\}$. It is easy to see that the set of reachable configurations is $R_f^*(10^*) = 0^*10^*$.

4 Results

We present in this section solutions of the problem (Π) when the class of representation structures correspond to (subclasses of) recognizable sets. Let us recall that an n-dim recognizable set is a finite union of sets of the form $L_1 \times \ldots \times L_n$ where each L_i is a regular set (i.e., FSM definable), for $i \in \{1, \ldots, n\}$.

Clearly, the class of recognizable sets enjoys the closure and decision properties required from symbolic representation structures. Indeed, this class is closed under all boolean operations, it is also closed under the application of regular relations (notice that the relation $R_\dagger$, with $\dagger \in \{p, c, f\}$, is obviously regular for any rewriting systems R), and its inclusion problem is decidable.

4.1 Prefix rewriting

The following theorem has been proved several times by authors from different communities with different motivations (see e.g., [Cau92,BEM97,FWW97]).

Theorem 1. *Let R be a 1-dim controlled rewriting system. Then, for every effectively recognizable set ϕ, the set $R_p^*(\phi)$ is effectively recognizable.*

In [BEM97,FWW97], versions of this theorem are used to define verification algorithms for pushdown systems against different specification logics (temporal logics and μ-calculi). Reachability analysis and model checking techniques for pushdown systems have applications in the domain of program analysis [EK99,ES01].

Unfortunately, there is no algorithm which constructs the set $R_p^*(\phi)$ for any 2-dim rewriting system R and recognizable set ϕ. This can be shown by a straightforward reduction of the Post correspondence problem.

4.2 Cyclic rewriting

It is well known that a finite automaton equipped with a FIFO queue is as powerful as a Turing machine. So, the R_c^* image is obviously not computable for any 1-dim controlled rewriting system. Moreover, such a model can be simulated very easily by a 1-dim cyclic rewriting system: a rule of the form $(q, x) \mapsto (q', y)$ can be simulated by the application of a rule $qx \mapsto yq'$ followed by rotation rules of the form $a \mapsto a$, for all symbols a which are not control states.

Hence, in order to solve the problem (Π) for cyclic rewriting, it is necessary to restrict the considered class of rewriting systems. A typical restriction is to consider controlled rewriting systems corresponding to control loops. A *control loop* is a set of rules

$$r_1 : (q_1, \boldsymbol{x_1}) \mapsto (q'_1, \boldsymbol{y_1})$$
$$\cdots$$
$$r_m : (q_m, \boldsymbol{x_m}) \mapsto (q'_m, \boldsymbol{y_m})$$

such that, (1) $\forall i, j \in \{1, \ldots, m\}$ with $i \neq j$, $q_i \neq q_j$ and $q'_i \neq q'_j$, (2) $\forall i \in \{1, \ldots, m-1\}$, $q'_i = q_{i+1}$, and (3) $q'_m = q_1$.

Boigelot et al. have shown the following result:

Theorem 2 ([BGWW97]). *Let R be a 1-dim control loop. Then, for every effectively recognizable set ϕ, the set $R_c^*(\phi)$ is effectively recognizable.*

For systems of higher dimensions (even for 2-dim systems), the R_c^* image is not recognizable, in general. Indeed, consider for instance the self-loop $R = \{(q, \epsilon, \epsilon) \mapsto (q, a, a)\}$. Then, $R_c^*(q, \epsilon, \epsilon) = \{(q, a^n, a^n) \ : \ n \geq 0\}$ which is a non-recognizable set.

[BGWW97] provides a characterization of the control loops R such that R_c^* preserves recognizability, as well as an algorithm which constructs for such loops a finite automaton representing the R_c^* image of any given recognizable set.

In [BH97] we show that the effect of iterating any control loop can be characterized using representation structures defining a class of non-recognizable sets enjoying all needed closure and decision properties for symbolic reachability analysis. These structures, called CQDD's, correspond to a class of constrained (products of) deterministic automata. The constraints we consider are expressed in Presburger arithmetics and concern the number of times transitions of the automata are taken in the accepting runs. For instance, the set $R_c^*(q, \epsilon, \epsilon)$ above can be defined as a product of two automata A_1 and A_2 each of them recognizing the language a^*, under the constraint imposing that the number of a-transitions taken in each of the two automata are the same (see [BH97] for more details on CQDD's). We have the following result:

Theorem 3 ([BH97]). *Let R be a n-dim control loop. Then, for every effectively CQDD representable set ϕ, the set $R_c^*(\phi)$ is effectively CQDD representable.*

A similar theorem can be shown for prefix rewriting, i.e., the class of CQDD's is closed under R_p^* for any n-dim control loop R.

As mentioned in Section 3, cyclic (controlled) rewriting systems are suitable for modeling communicating systems through FIFO channels, e.g., communication protocols. In many cases, the purpose of these protocols is to ensure a perfect data transfer through unreliable channels. Hence, it is natural in this context to consider models where channels are *lossy* in the sense that they can lose a message at any time. In our setting, the lossiness assumption can be taken into account by considering a *weak* cyclic rewriting relation, where configurations can get smaller according to the subword relation (meaning that some symbols are lost), before and after any cyclic rewriting step.

Let $\preceq \subseteq \Sigma^* \times \Sigma^*$ be the subword relation, i.e., $a_1 \ldots a_n \preceq b_1 \ldots b_m$ if there exists $i_1, \ldots, i_n \in \{1, \ldots, m\}$ such that $i_1 < \ldots < i_n$ and $\forall j \in \{1, \ldots, n\}.\ a_j = b_{i_j}$. We consider the product generalization of this relation to vectors of words.

Let R be a n-dim rewriting system over Σ. We define the weak cyclic rewriting relation $R_{wc} \in (\Sigma^*)^n \times (\Sigma^*)^n)$ as follows: for every $\boldsymbol{u}, \boldsymbol{v} \in (\Sigma^*)^n$, $(\boldsymbol{u}, \boldsymbol{v}) \in R_{wc}$ if and only if there exist $\boldsymbol{u}', \boldsymbol{v}' \in (\Sigma^*)^n$ such that $\boldsymbol{u}' \preceq \boldsymbol{u}$, $\boldsymbol{v}' \in R_c(\boldsymbol{u}')$, and $\boldsymbol{v} \preceq \boldsymbol{v}'$.

An n-dim language L is *downward closed* w.r.t. the subword relation if $\forall \boldsymbol{u}, \boldsymbol{v} \in (\Sigma^*)^n$, if $\boldsymbol{v} \in L$ and $\boldsymbol{u} \preceq \boldsymbol{v}$, then $\boldsymbol{u} \in L$. Let $L_{\preceq}$ denote the *downward closure* of L, i.e., the smallest downward closed set which includes L. Clearly, for every rewriting system R and every set ϕ, the set $R_{wc}^*(\phi)$ is downward closed. Hence, by showing that every downward closed set w.r.t $\preceq$ is a recognizable set, the following fact can be deduced.

Theorem 4 ([AČJT96,CFI96]). *For every rewriting system R, and every set ϕ, the set $R_{wc}^*(\phi)$ is a recognizable set.*

Theorem 4 does not say that the set $R_{wc}^*(\phi)$ is constructible, even though it is recognizable. Actually, we know from the results in [May00] that:

Theorem 5. *There is no algorithm which constructs the set $R_{wc}^*(\phi)$ for any given 1-dim controlled rewriting system R and recognizable set ϕ.*

We can refine Theorem 4 by defining representation structures which capture precisely the class of downward closed sets. These representation structures correspond to a particular subclass of regular expressions called *simple regular expressions* (SRE for short). Their definition is as follows: Let us call *atomic expression* any expression of the form a where $a \in \Sigma$, of the form A^* where $A \subseteq \Sigma$. A *product* is either the empty word ϵ, of a finite sequence $e_1 \cdots e_m$ of atomic expressions. Then, an SRE is either $\emptyset$, or a finite union $p_1 + \cdots + p_n$ of products. A n-dim SRE set is a finite union of Cartesian products of n SRE sets. It is very easy to see that every SRE set is downward closed w.r.t. the subword relation. Conversely, by showing that for every recognizable set L, the set $L_{\preceq}$ is effectively SRE representable, we obtain the following fact:

Theorem 6 ([ABJ98]). *SRE sets are precisely the downward closed sets w.r.t. the subword relation.*

The class SRE has interesting properties which makes it suitable for efficient reachability analysis of lossy FIFO-channel systems.

Theorem 7 ([ABJ98]). *The class of SRE sets is closed under union, intersection, and application of regular relations (e.g., rewriting relations). Moreover, the inclusion problem for SREs can be solved in polynomial time.*

Notice that the class of SREs is not closed under complementation. Indeed, complements of SRE languages are upward closed languages w.r.t. the subword relation. They correspond to finite unions of languages of the form $\Sigma^* a_1 \Sigma^* a_2 \cdots a_n \Sigma^*$.

Theorem 8 ([ABJ98]). *Let R be a n-dim control loop. Then, for every SRE set ϕ, it is possible to construct an SRE representation of the set $R^*_{wc}(\phi)$ which has a polynomial size w.r.t. the size of ϕ.*

Based on the two theorems above, we derive a symbolic reachability analysis procedure as described in Section 2. This procedure has been implemented and used to analyze in a fully automatic manner several infinite-state models of communication protocols such as the alternating bit protocol, the sliding window protocol, and the bounded retransmission protocol [AAB99,ABS01].

Now, it is very easy to construct a model for which computing the effect of control loops does not help the reachability analysis procedure to terminate. Consider for instance the system R with two rules $r_1 : a \mapsto bb$ and $r_2 : b \mapsto aa$ corresponding to two self-loops (loops on a single control state q which is omitted here). It can be seen that $R^*_{wc}(a) = a^* b^* + b^* a^*$ (notice that, due to lossiness, there cannot be any constraints on the numbers of a and b in the reachable configurations). However, it is impossible to compute this set by reasoning about a finite number of iterated compositions of the two rules of R (control loops corresponding to compositions of the two considered self-loops). To see this, let us consider the relation corresponding to any of such a loop. This relation can be written as

$$\theta = \{r_2\}^{m_k}_{wc} \circ \{r_1\}^{n_k}_{wc} \circ \cdots \circ \{r_2\}^{m_1}_{wc} \circ \{r_1\}^{n_1}_{wc}$$

where the m_i's and n_i's are positive integers. It is can be checked that, for every word $w \in \Sigma^*$, $\theta^*(w)$ is always a finite language

For instance, let $w = babab$. Then, we have

$$\{r_1\}^*_{wc}(w) = \{babb^2, bb^2, bb^4\}_{\preceq} = \{babb^2, bb^4\}_{\preceq}$$
$$\{r_2\}^*_{wc}(w) = \{ababa^2, aba^2, a^2, aba^4, a^4, a^6\}_{\preceq} = \{ababa^2, aba^4, a^6\}_{\preceq}$$

Notice that the number of possible iterations of the relations $\{r_1\}_{wc}$ and $\{r_2\}_{wc}$ is always bounded. It depends on the number of occurrences of a's (resp. b's) in the initial word w.

As another example, take $\theta = \{r_2\}_{wc} \circ \{r_1\}_{wc}$ and $w = a$. Then, we have

$$\theta(w) = \{r_2\}_{wc}(\{b^2\}_{\preceq}) = \{ba^2\}_{\preceq}$$
$$\theta^2(w) = \theta(\{ba^2, a^2, ba, b, a, \epsilon\}) = \{r_2\}_{wc}(\{ab^2\}_{\preceq}) = \{ba^2\}_{\preceq}$$

Thus, we have $\theta^*(a) = \{ba^2\}_{\preceq}$.

Since $R^*_{wc}(a)$ is an infinite set, and the iteration of each relation θ of the form specified above can only produce a finite set of words, it can be concluded that the reachability analysis procedure using only meta-transitions like θ does not terminate in this case.

An interesting question is under which conditions it is possible to compute the effect of iterating *nested control loops*. Unfortunately, we have the following negative result:

Theorem 9 ([ABB01]). *There is no algorithm which constructs the set $R^*_{wc}(\phi)$ for any given 1-dim rewriting system R and any set ϕ.*

This means, that it is even impossible to compute the effect of sets of self-loops of the form $(q,x) \mapsto (q,y)$ where x and y are two words over Σ. To prove this result, we need rules where the left hand side x is of size 2. However, the situation is different when this size is assumed to be at most one.

We consider that an n-dim rewriting rule r is *context-free* if $lhs(r) \in (\Sigma \cup \{\epsilon\})^n$. A n-dim context-free rewriting system is a set of n-dim context-free rules. For instance, the system $R = \{a \mapsto bb, b \mapsto aa\}$ considered above is a context-free system. We have the following result:

Theorem 10 ([ABB01]). *Let R be a 1-dim context-free rewriting system. Then, for every effectively SRE set ϕ, the set $R^*_{wc}(\phi)$ is effectively SRE.*

Using Theorem 5, it is very easy to show that the result above cannot be extended to 2-dim context-free rewriting systems. Therefore, the question is under which conditions it is possible to construct the effect of n-dim context-free systems. We propose hereafter one such condition.

A rewriting system R is a *ring* if, for every rule $r : (x_1, \ldots, x_n) \mapsto (y_1, \ldots, y_n)$ in R, $\exists i \in \{1, \ldots, n\}$ such that $\forall j \neq i.\ x_j = \epsilon$ and $\forall j \neq (i+1) \bmod n.\ y_j = \epsilon$.

Thus, each rule r in a ring is either of the form $(\epsilon, \ldots, \epsilon, x_i, \epsilon, \ldots, \epsilon) \mapsto (\epsilon, \ldots, \epsilon, y_{i+1}, \epsilon, \ldots, \epsilon)$ or of the form $(\epsilon, \ldots, \epsilon, x_n) \mapsto (y_1, \epsilon, \ldots, \epsilon)$. Intuitively, the each rule in these systems correspond to actions of FIFO-channel systems where a word x is received from a channel of index i, and a word y is sent to the channel of index $(i+1) \bmod n$.

Theorem 11 ([ABB01]). *Let R be a n-dim context-free ring. Then, for every effectively SRE set ϕ, the set $R^*_{wc}(\phi)$ is effectively SRE.*

4.3 Factor rewriting

As mentioned in Section 3, factor rewriting rules can be used to represent transitions in parametrized systems (networks) with an arbitrary number of identical finite-state components. An interesting class of rewriting rules which appear in this context are the so-called *semi-commutations*: A 1-dim rewriting rule is a semi-commutation if it is of the form $ab \mapsto ba$ where $a, b \in \Sigma$. A semi-commutation rewriting system is a set of semi-commutation rules.

Semi-commutations are naturally used to model transitions corresponding to information exchange between neighbors, e.g., token passing protocols for mutual exclusion (see Section 3), leader election algorithms, etc. We present later an example where semi-commutation appear in the model of a lift controller for an arbitrary number of floors. In that example, semi-commutation rules correspond to the actions of moving up or down from one floor to its immediate successor.

It is well known that the class of recognizable sets is in general not closed under R_f^* where R is any semi-commutation system. For instance, consider the system $R = \{ab \mapsto ba\}$. Then, it is easy to see that for $\phi = (ab)^*$, the set $R_f^*(\phi)$ is not recognizable.

Therefore, the question is to find a class of representation structures defining a subclass of recognizable sets which is closed under iterative semi-commutation rewriting. As an answer to this question, we propose a subclass of regular expressions called APC (alphabetic pattern constraints). We define APCs exactly as the SREs introduced above, except that we also allow in APCs atomic expressions of the form a, where $a \in \Sigma$ (APC are not downward closed w.r.t. $\preceq$ in general). In other words, APC languages are finite unions of languages of the form $\Sigma_1^* a_1 \Sigma_2^* \cdots a_n \Sigma_{n+1}^*$ where the a_i's are symbols in Σ and the Σ_i's are subsets of Σ. (The class APC coincides with the class of languages on level 3/2 of Straubing's concatenation hierarchy [PW97].)

The motivation behind the consideration of this particular kind of languages is that they appear naturally in many specification and verification contexts. First, APC languages can be used to express properties based on specifying some patterns appearing within configurations. Typically, negations of (some) safety properties are expressed by an APC defining the set of all bad patterns. For example, in the case of the token passing protocol mentioned in Section 3, the set of bad configurations, i.e., all those which do not satisfy the mutual exclusion property, is defined by $(0+1)^*1(0+1)^*1(0+1)^*$. Thus, since this set has empty intersection with the set of reachable configurations 0^*10^*, it can be concluded that the mutual exclusion property is satisfied. Furthermore, it turns out that the reachability sets of many infinite-state systems and parametrized systems, including communication protocols like the alternation-bit and the sliding window, and parametrized mutual exclusion protocols such as the token ring, Szymanski's, Burns', or Dijkstra's protocols, are all expressible as APCs (see [ABJ98,AAB99,ABJN99,JN00,BJNT00,Tou00]).

It can be shown that the class APC has the following properties.

Theorem 12 ([BMT01]). *The class of APCs is closed under union, intersection, and rewriting (application of single rewriting rules), but it is not closed under complementation. The inclusion problem for APCs is PSPACE-complete.*

The main closure result about APCs is the following:

Theorem 13 ([Tou00,BMT01]). *Let R be a semi-commutation rewriting system. Then, for every APC set ϕ, the set $R_f^*(\phi)$ is effectively APC.*

Actually, this result can be slightly extended to system including symbol substitutions. We call a symbol substitution rewriting system any set of rules of the form $a \mapsto b$. First, it is easy to see that APCs are effectively closed under R_f^* for any symbol substitution rewriting system. The proof of Theorem 13 can be easily adapted to rewriting systems which are sets of semi-commutations and symbol substitutions [Tou00].

Let us illustrate the use of these results on a simple example. We consider a lift controller which has the following behavior: People can arrive at any time to any floor and declare their will to move up or down. The lift is initially at the lower floor, and then it keeps moving from the lower floor to the upper one, and back. In its ascending (resp. descending) phase, it takes all the people who are waiting for moving up (resp. down) and ignores the others. They are taken into account in the next phase.

For every n (number of floors), a configuration of this system can be represented by a word of the form

$$\# x_1 \cdots x_j y x_{j+1} \cdots x_n \#$$

where $y \in \{a\uparrow, a\downarrow\}$, and $x_i \in \{\perp, b\uparrow, b\downarrow, b\uparrow\!\!\downarrow\}$, for $i \in \{1, \ldots, n\}$. The symbol corresponding to x_i represents the state of the i^{th} floor: $x_i = b\uparrow\!\!\downarrow$ if there are people waiting for moving up and other people (at the same floor) waiting for moving down, $x_i = b\uparrow$ (resp. $x_i = b\downarrow$) means that there are people waiting at this floor and all of them want to move up (resp. down), and $x_i = \perp$ means that nobody is waiting at this floor. The symbol corresponding to y gives the position of the lift: in the configuration given above, if $y = a\uparrow$ (resp. $y = a\downarrow$) then, the lift is at floor $j+1$ (resp. j), and it is moving up (resp. down).

The set of all initial configurations, for an arbitrary number of floors, is the set of words $\phi_0 = \# a\uparrow \perp^* \#$, which means that initially, the lift is at the lower floor and there is no requests at any floor. The dynamic of the system can be modeled by the following rewriting rules:

$$\perp \mapsto b\uparrow \tag{1}$$
$$\perp \mapsto b\downarrow \tag{2}$$
$$b\uparrow \mapsto b\uparrow\!\!\downarrow \tag{3}$$
$$b\downarrow \mapsto b\uparrow\!\!\downarrow \tag{4}$$
$$a\uparrow \perp \mapsto \perp a\uparrow \tag{5}$$
$$a\uparrow b\downarrow \mapsto b\downarrow a\uparrow \tag{6}$$
$$a\uparrow b\uparrow \mapsto \perp a\uparrow \tag{7}$$
$$a\uparrow b\uparrow\!\!\downarrow \mapsto b\downarrow a\uparrow \tag{8}$$
$$a\uparrow \# \mapsto a\downarrow \# \tag{9}$$
$$\perp a\downarrow \mapsto a\downarrow \perp \tag{10}$$
$$b\uparrow a\downarrow \mapsto a\downarrow b\uparrow \tag{11}$$
$$b\downarrow a\downarrow \mapsto a\downarrow \perp \tag{12}$$
$$b\uparrow\!\!\downarrow a\downarrow \mapsto a\downarrow b\uparrow \tag{13}$$
$$\# a\downarrow \mapsto \# a\uparrow \tag{14}$$

Rules 1, 2, 3, and 4 are symbol substitutions modeling the arrival of users. Let us call request their corresponding action. Rules 5 and 6 (resp. 10 and 11) are semi-commutations modeling the moves of the lift upward (resp. downward). They correspond to the action move-up (resp. move-down). Rules 7 and 8 (resp. 12 and 13) represent the action of taking at some floor the people who want to move up (resp. down). We call the corresponding actions take-up (resp. take-down). Finally, rules 9 and 14 represent the actions of switching from the ascending to the descending phase (action up2down), and vice-versa (action down2up).

Table 1 shows the computations of the reachable configurations of the lift controller according to a depth first search strategy with priority to meta-transitions (we omit some unimportant steps). The used meta-transitions are request* corresponding to the relation $\{1 \cup 2 \cup 3 \cup 4\}^*_f$, move-up* corresponding to $\{5 \cup 6\}^*_f$, and move-down* corresponding to $\{10 \cup 11\}^*_f$. The image by request* is easy to compute (APCs are effectively closed under iterated symbol substitution rewriting), and the images by move-up* and move-down* are computable by the algorithm underlying Theorem 13.

ϕ_0	request*	$\#a\!\uparrow (\perp + b\!\uparrow + b\!\downarrow + b\!\updownarrow)^*\#$	ϕ_1
ϕ_1	move-up*	$\#(\perp + b\!\downarrow)^* a\!\uparrow (\perp + b\!\uparrow + b\!\downarrow + b\!\updownarrow)^*\#$	ϕ_2
ϕ_2	request*	$\#(\perp + b\!\uparrow + b\!\downarrow + b\!\updownarrow)^* a\!\uparrow (\perp + b\!\uparrow + b\!\downarrow + b\!\updownarrow)^*\#$	ϕ_3
ϕ_3	take-up	$\#(\perp + b\!\uparrow + b\!\downarrow + b\!\updownarrow)^*(\perp + b\!\downarrow) a\!\uparrow (\perp + b\!\uparrow + b\!\downarrow + b\!\updownarrow)^*\#$	$\subseteq \phi_3$
ϕ_3	up2down	$\#(\perp + b\!\uparrow + b\!\downarrow + b\!\updownarrow)^* a\!\downarrow \#$	ϕ_4
ϕ_4	move-down*	$\#(\perp + b\!\uparrow + b\!\downarrow + b\!\updownarrow)^* a\!\downarrow (\perp + b\!\uparrow)^*\#$	ϕ_5
ϕ_5	request*	$\#(\perp + b\!\uparrow + b\!\downarrow + b\!\updownarrow)^* a\!\downarrow (\perp + b\!\uparrow + b\!\downarrow + b\!\updownarrow)^*\#$	ϕ_6
ϕ_6	take-down	$\#(\perp + b\!\uparrow + b\!\downarrow + b\!\updownarrow)^* a\!\downarrow (\perp + b\!\uparrow)(\perp + b\!\uparrow + b\!\downarrow + b\!\updownarrow)^*\#$	$\subseteq \phi_6$
ϕ_6	down2up	$\#a\!\uparrow (\perp + b\!\uparrow + b\!\downarrow + b\!\updownarrow)^*\#$	$= \phi_1$

Table 1. Reachability Analysis of the Lift Controller

As shown in Table 1, the reachability analysis terminates in this case thanks to the use of meta-transitions. It is worth noting that the reachability analysis procedure also gives (for free) a finite abstraction of the analyzed infinite-state model. Indeed, Table 1 defines an abstract reachability graph of the lift controller which is shown in Figure 1.

5 Related Work

Several papers propose symbolic reachability analysis techniques for infinite-state systems based on using representations of languages to define sets of configurations. In these works, sets of configurations are represented by means of various kinds of automata, regular expressions, and formulas of monadic first or second order logics (see e.g., [BG96,BEM97,BH97,BGWW97,KMM+97,WB98] [BJNT00,PS00,FIS00]).

Papers such as [KMM+97,WB98,BJNT00,PS00] introduce a uniform verification paradigm for infinite-state systems, called *regular model-checking*, based

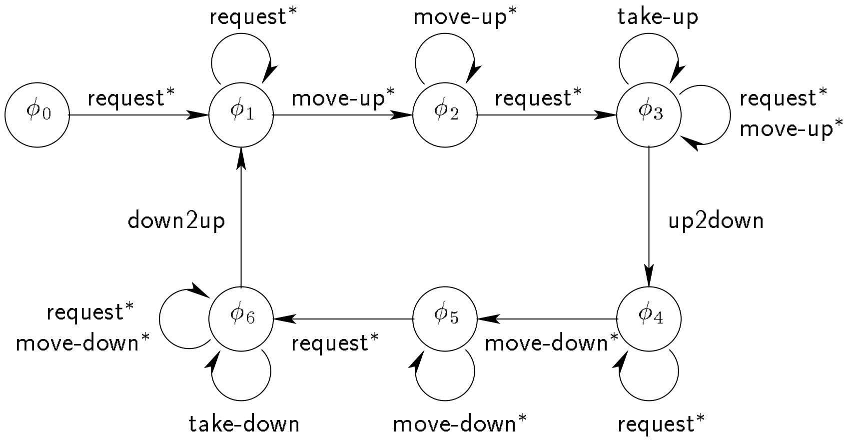

Fig. 1. Abstract Reachability Graph of the Lift Controller

on the use of regular languages (finite automata or WS1S formulas) as symbolic representations, and of regular relations (finite transducers or formulas) as models of transition relations of systems. The concepts we present is this paper are very close to those developed for regular model-checking. However, we can make the following comparison between the two frameworks.

First, we do not require here that the manipulated languages are regular. For instance, the results of [BH97] show that representation structures defining non-regular languages can be used and they are needed for some applications.

Moreover, the verification approach adopted in, e.g., [BJNT00,PS00] consists in constructing (when possible) transitive closures of regular relations, (i.e., given a regular relation ρ, construct a representation of ρ^*, as a finite transducer for instance). This problem is more general and of course harder than the problem (Π) we have considered in this paper (see Section 2), which is to construct the image of a given set ϕ by ρ^*. Indeed, there are many cases where $\rho^*(\phi)$ is computable for every ϕ in some class of languages, whereas ρ^* is not constructible, or at least, not regular (e.g., for relation induced by semi-commutation rewriting systems [BMT01]). Nevertheless, in the context of regular model checking, interesting classes of relations for which the transitive closure is computable have been identified in e.g., [ABJN99,JN00]. Other works propose incomplete procedures for computing transitive closures of relations [BJNT00,PS00,DLS01].

Also, for the sake of simplicity, we have considered in this paper only special kinds of rewriting systems (for instance, these rewriting systems cannot define all the relations considered in [ABJN99,JN00,BJNT00]). Of course, more general forms of rewriting systems can be used within the framework we present.

The symbolic reachability analysis approach we describe in this paper uses the concept of meta-transition introduced in [BW94] in order to help termination. This technique can be seen as a fixpoint acceleration in the context of abstract interpretation [CC77]. However, these works use widening operators which lead in general to the computation of an upper-approximation of the reachability set, whereas the results we present in this paper allow to perform exact computations. It is worth noting that widening operations are defined depending only on the intermediary sets which are generated during the computation of the reachability set, regardless of the applied actions. In contrast, the approach we adopt here for acceleration takes into account the applied actions (rewriting rules) in order to compute the exact effect of their iteration. In [BJNT00,Tou00], widening techniques on automata and transducers are defined for regular model-checking.

The use of rewriting systems as models for infinite-state systems has been considered for instance in [Cau92,Mol96,May98]. These works address different questions from the one considered here. They are concerned with the decidability and the complexity of behavioral equivalences such as bisimulation [Cau92,Mol96] or model-checking against various propositional temporal logics [May98]. Rewriting systems are also used to model parametrized networks of identical processes in [FO97] where rewriting techniques are applied for invariant checking, but no algorithms for automatic computation of the closure of languages by rewriting systems are provided.

Finally, we have considered in this paper only rewriting systems on words. The approach we present can also be extended to rewriting systems on other structures such as trees, rings, grids, and graphs in general, in order to deal with wider classes of systems. Let us mention some of the few existing results on this topic. In [KMM+97], an extension of the regular model-checking framework to the case of tree languages is proposed in order to verify parametrized networks with a tree-like topology. However, this paper does not provide acceleration techniques for reachability analysis. In [LS01], tree automata are used to characterize reachability sets (set of terms) for a class of processes with parallel and sequential composition which subsumes the class of context-free processes. Finally, we show in [BMT01] that Theorem 13 about closure under iterated semi-commutation rewriting can be generalized to the case of rings (circular words).

References

[AAB99] P. Abdulla, A. Annichini, and A. Bouajjani. Symbolic Verification of Lossy Channel Systems: Application to the Bounded Retransmission Protocol. In *TACAS'99*. LNCS 1579, 1999.

[ABB01] P. Abdulla, L. Boasson, and A. Bouajjani. Effective Lossy Queue Languages. In *ICALP'01*. LNCS, Springer-Verlag, 2001.

[ABJ98] P. Abdulla, A. Bouajjani, and B. Jonsson. On-the-fly Analysis of Systems with Unbounded, Lossy Fifo Channels. In *CAV'98*. LNCS 1427, 1998.

[ABJN99] P. Abdulla, A. Bouajjani, B. Jonsson, and M. Nilsson. Handling Global Conditions in Parametrized System Verification. In *CAV'99*. LNCS 1633, 1999.

[ABS01] A. Annichini, A. Bouajjani, and M. Sighireanu. TReX: A Tool for Reachability Analysis of Complex Systems. In *CAV'01*. LNCS, Springer-Verlag, 2001.

[AČJT96] P. Abdulla, K. Čerāns, B. Jonsson, and Y.-K. Tsay. General Decidability Theorems for Infinite-State Systems. In *LICS'96*. IEEE, 1996.

[BEM97] A. Bouajjani, J. Esparza, and O. Maler. Reachability Analysis of Push-down Automata: Application to Model Checking. In *CONCUR'97*. LNCS 1243, 1997.

[BG96] B. Boigelot and P. Godefroid. Symbolic verification of communication protocols with infinite state spaces using QDDs. In *CAV'96*. LNCS 1102, 1996.

[BGWW97] B. Boigelot, P. Godefroid, B. Willems, and P. Wolper. The power of QDDs. In *SAS'97*. LNCS 1302, 1997.

[BH97] A. Bouajjani and P. Habermehl. Symbolic Reachability Analysis of FIFO-Channel Systems with Nonregular Sets of Configurations. In *ICALP'97*. LNCS 1256, 1997. Full version in TCS **221** (1/2), pp 221-250, 1999.

[BJNT00] A. Bouajjani, B. Jonsson, M. Nilsson, and T. Touili. Regular Model Checking. In *CAV'00*. LNCS 1855, 2000.

[BMT01] A. Bouajjani, A. Muscholl, and T. Touili. Permutation Rewriting and Algorithmic Verification. In *LICS'01*. IEEE, 2001.

[BW94] B. Boigelot and P. Wolper. Symbolic Verification with Periodic Sets. In *CAV'94*. LNCS 818, 1994.

[Cau92] D. Caucal. On the Regular Structure of Prefix Rewriting. *TCS*, 106(1):61–86, 1992.

[CC77] P. Cousot and R. Cousot. Static Determination of Dynamic Properties of Recursive Procedures. In *IFIP Conf. on Formal Description of Programming Concepts*. North-Holland Pub., 1977.

[CFI96] Gérard Cécé, Alain Finkel, and S. Purushothaman Iyer. Unreliable Channels Are Easier to Verify Than Perfect Channels. *Inform. and Comput.*, 124(1):20–31, 1996.

[DLS01] D. Dams, Y. Lakhnech, and M. Steffen. Iterating transducers. In *CAV'01*. LNCS, Springer-Verlag, 2001.

[EK99] J. Esparza and J. Knoop. An automata-theoretic approach to interprocedural dataflow analysis. In *FOSSACS'99*. LNCS 1578, 1999.

[ES01] J. Esparza and S. Schwoon. A BDD-based Model Checker for Recursive Programs. In *CAV'01*. LNCS, Springer-Verlag, 2001.

[FO97] L. Fribourg and H. Olsén. Reachability sets of parametrized rings as regular languages. *Electronic Notes in Theoretical Computer Science*, 1997.

[FIS00] A. Finkel, S. Purushothaman Iyer, and G. Sutre. Well-abstracted transition systems. In *CONCUR'00*. LNCS 1877, 2000.

[FWW97] A. Finkel, B. Willems, and P. Wolper. A Direct Symbolic Approach to Model Checking Pushdown Systems. In *Infinity'97*, 1997.

[JN00] B. Jonsson and M. Nilsson. Transitive Closures of Regular Relations for Verifying Infinite-State Systems. In *TACAS'00*. LNCS 1785, 2000.

[KMM+97] Y. Kesten, O. Maler, M. Marcus, A. Pnueli, and E. Shahar. Symbolic model checking with rich assertional languages. In *CAV'97*. LNCS 1254, 1997.

[LS01] D. Lugiez, and P. Schnoebelen. The regular viewpoint on PA-processes. In *Theoretical Computer Science*. to appear, 2001.

[May98] R. Mayr. Decidability and Complexity of Model Checking Problems for Infinite State Systems. PhD Thesis, Technische Universitaet Muenchen, April 1998.

[May00] R. Mayr. Undecidable Problems in Unreliable Computations. In *LATIN'00*. LNCS 1776, 2000.

[Mol96] F. Moller. Infinite results. In *CONCUR'96*. LNCS 1119, 1996.

[PS00] A. Pnueli and E. Shahar. Liveness and acceleration in parametrized verification. In *CAV'00*. LNCS 1855, 2000.

[PW97] J.-E. Pin and P. Weil. Polynomial closure and unambiguous product. *Theory of Computing Systems*, 30:383–422, 1997.

[Tou00] T. Touili. Vérification de Réseaux Paramétrés Basée sur des Techniques de Réécriture. MSc. Thesis (French DEA) report, Liafa Lab., University of Paris 7, July 2000. http://verif.liafa.jussieu.fr/~touili.

[WB98] P. Wolper and B. Boigelot. Verifying systems with infinite but regular state spaces. In *CAV'98*. LNCS 1427, 1998.