Electronic Notes in Theoretical Computer Science 50 No. 4 (2001)
URL: http://www.elsevier.nl/locate/entcs/volume50.html 16 pages

Regular Model Checking using Widening Techniques [2]

Tayssir Touili [1]

LIAFA, Université Paris VII
2, place Jussieu, case 7014, F-75251 Paris Cedex 05

Abstract

In this paper, we consider symbolic model checking of safety properties of linear parametrized systems. Sets of configurations are represented by regular languages and actions by regular relations. Since the verification problem amounts to the computation of the reachability set, we focus on the computation of $R^*(\phi)$ for a regular relation R and a regular language ϕ. We present a technique called *regular widening* that allows, when it terminates, the computation of either the reachability set $R^*(\phi)$ of a system or the transitive closure R^* of a regular relation. We show that our method can be uniformly applied to several parametrized systems. Furthermore, we show that it is powerful enough to simulate some existing methods that compute either R^* or $R^*(\phi)$ for each R (resp. ϕ) belonging to a subclass of regular relations (resp. belonging to a subclass of regular languages).

1 Introduction

Many protocols dealing with distributed systems are defined in a parametrized way. It is then important to be able to model check these protocols in their parametrized version, i.e., independently of the number of processes in the system. The problem of verifying these parametrized systems being undecidable [2], we must either identify decidable subclasses [11,12,10], or provide semi-algorithms for which termination is not guaranteed. In this work, we restrict ourselves to the model checking of safety properties of linear parametrized systems and propose a semi-decision procedure that solves this problem.

A linear parametrized system can be seen as an infinite union of similar systems S_n composed of n indistinguishable processes organized in a linear

[1] Email: Tayssir.Touili@liafa.jussieu.fr
[2] This work was supported in part by the European Commission (FET project ADVANCE, contract No IST-1999-29082).

topology. Since safety properties express that some "dangerous" configurations cannot be reached by the system, the standard way to model check such properties is to compute the set of all reachable configurations and verify that it does not contain the bad ones.

This method can be applied in the case of parametrized systems if we are able to express sets of system states of arbitrary size. To that end, we need to find a finite symbolic representation of infinite sets of states of a system.

Following [16,1,15,6], we use regular languages to represent infinite sets of configurations: we encode the global state of a system S_n into a word of length n by concatenating the local states of the different processes. Consequently, a set of configurations can be represented by a language. In this context, each action of the system can be seen as a regular relation R over words such that $(w, w') \in R$ means that the considered action transforms the global state of the system from w to w'.

In this framework, to compute the reachability set of a given parametrized system whose actions are represented by the relations $\{R_1, \ldots, R_n\}$, the standard algorithm starts from ϕ_0, the set of initial configurations of the system, and produces at each step the set of successors corresponding to a single application of one relation R_i. However, this naive algorithm will not terminate in the case of parametrized systems since the reachability set is infinite. To solve this problem we use acceleration techniques [21,1,15,6,18] that consist in applying at each step the operation R_i^* instead of R_i, in order to speed up the termination of the computation.

The verification problem is then reduced to the computation of $R^*(\phi)$ for a regular language ϕ and a regular relation R, where R^* is the transitive-reflexive closure of R. This problem being in general undecidable we need to find subclasses $\mathcal{R}$ of regular relations such that we can effectively characterize R^* (e.g., by a transducer) for every relation R in $\mathcal{R}$.

But sometimes, computing $R^*(\phi)$ for some regular language ϕ is easier than characterizing R^* (especially when R^* is not regular). Take as an example the relation R corresponding to the rewriting rule $ab \leftrightarrow ba$, and the language $\phi = a^*b^*$, then it is easy to see that $R^*(\phi)$ equals $(a+b)^*$ whereas R^* cannot be characterized by a finite transducer since it is not regular ($R^*((ab)^*) = \{w \in \Sigma^* \mid |w|_a = |w|_b\}$). Consequently, we need to find subclasses $\mathcal{R}$ of regular relations and subclasses $\mathcal{L}$ of regular languages such that we can effectively compute $R^*(\phi)$ for every relation R in $\mathcal{R}$ and every language ϕ in $\mathcal{L}$.

Several works have been done in this direction [8,4,3,5]. In particular, Abdulla et al.[1] have introduced a class of unary regular relations[3] for which they characterize R^* by a finite transducer. In [7], we have identified a subclass of regular languages called *Alphabetic Pattern Constraints* (APC) which can naturally be used to model parametrized systems such as mutual exclusion protocols. We have provided an algorithm that computes $R^*(\phi)$ for every

[3] unary relations are relations that change only one position in a given word

APC language ϕ and every semi-commutation relation R (i.e., a relation corresponding to a set of rewriting rules of the form $ab \rightarrow ba$). This kind of relations can be used to model protocols in which processes communicate by exchanging information between neighbors, such as the Token Ring Protocol.

However, in the general case, if we are not able to compute the exact reachability set of a system, we can compute an upper approximation of it, and conclude that the system is correct if this upper approximation does not include any bad configuration.

In [6], we have proposed a mechanism, called *regular widening*, that allows when it terminates the computation of an upper approximation of either R^* or $R^*(\phi)$ for a regular language ϕ and a regular relation R. The principle of our method consists in guessing automatically, by comparing ϕ and $R(\phi)$, the limit of the iteration of R: if the situation $\phi = \phi_1.\phi_2$ and $R(\phi) = \phi_1.\Delta.\phi_2$ holds, then we guess that R will always introduce a "Δ" in the middle and add $\phi_1.\Delta^*.\phi_2$ to the reachability set.

Our technique can be seen, in the framework of Abstract Interpretation, as the application of *widening operators* to accelerate the computation of fixpoints [9]. However, contrary to the notion of widening operator defined in [9], our *widening* techniques do not guarantee termination.

The applicability of the widening principle introduced in [6] is restricted to the cases where R can be applied in a unique position in ϕ. In this paper, we propose extensions of this mechanism so that: (1) we could deal with the cases where R can be applied in several positions in ϕ. We point out that if R is well-founded, then these widening techniques are *exact*, i.e., compute the exact reachability set (or the exact transitive closure of R). (2) We show how to compose these widening techniques, during the exploration of the reachability set, to compute the effect of the iteration of a sequence of relations. (3) We show that our new principle is powerful enough to simulate several existing constructions of either R^* or $R^*(\phi)$ for each R (resp. ϕ) belonging to a subclass of regular relations (resp. belonging to a subclass of regular languages). Indeed, we prove that it simulates the construction of R^* presented in [1] and allows the *exact* computation of $R^*(\phi)$ for every APC language ϕ and every semi-commutation relation R [7]. These two constructions constitute two novel proofs of the theorems given in [1] and [7].

Our method has been implemented in a tool prototype based on MONA [14]. This tool has been used to verify, in a fully automatic way, a number of mutual exclusion protocols including the Bakery algorithm by Lamport, Burns protocol, Dijkstra's and Szymanski's algorithms as well as the token ring protocol.

Outline. In the next section we present the main contribution of this paper: the *regular widening* techniques. An application of our method to the verification of the Burns protocol and a brief description of our tool prototype are given in Section 3. In Section 4 we show that the widening can simulate the computation of R^* given in [1], and prove that it computes $R^*(\phi)$ for every APC language ϕ and every semi-commutation relation R.

2 Regular Widening Techniques

Given a regular language ϕ and a regular relation R, our aim is to compute $R^*(\phi) = \bigcup_{i \geq 0} R^i(\phi)$. If we proceed naively, i.e., if we start by computing $R(\phi)$,$R^2(\phi)$, $R^3(\phi)$, ... etc, until we find an index k such that $R^{k+1}(\phi) \subseteq \bigcup_{i \leq k} R^i(\phi)$ we will certainly not terminate in all non trivial cases. Take as an example the Token Ring Protocol for mutual exclusion, where the right to access the critical section is represented by a token that the processes pass from the left to the right. The state of a process is then a if it has the token, and b if not. In the beginning, the leftmost process has the token, the initial configurations are then represented by the expression $\phi = ab^*$. The action that consists in passing the token from the left to the right can be represented by the relation $R = \mathit{copy}(\Sigma^*)(ab, ba)\mathit{copy}(\Sigma^*)$ where for each $L \in \Sigma^*$, $\mathit{copy}(L)$ denotes the set $\{(x, x) \in (\Sigma \times \Sigma)^* \mid x \in L\}$. Let us try to compute the reachability set $R^*(\phi)$ of this system. It can be seen that for each $k \geq 0$, $R^k(\phi) = b^k ab^*$, and thus there exists no index k such that $R^{k+1}(\phi) \subseteq \bigcup_{i \leq k} R^i(\phi)$ and our computation will never converge. What we need here is an automatic method that allows the computation of the limit $R^*(\phi) = b^*ab^*$. In this paper we present a mechanism called *regular widening* that can be used to compute this limit.

2.1 Regular Widening: Principle

In this section, we describe a new technique, called *regular widening*, that allows the *automatic* computation of an upper approximation of the effect of the iteration of a regular relation an arbitrary number of times. This mechanism is based on the comparison of ϕ and $R(\phi)$ and on the detection of some patterns that will be added at each application of R and which will finally produce loops in the limit language. To show the intuition, we will present gradually different widening principles: the first one is the most elementary and can be used only in the cases where R and ϕ are such that R can be applied in a unique position in ϕ. The second widening principle is useful when ϕ contains several positions where R can be applied, but R can change only one position at a time. Finally, the last principle, which is an extension of the two previous ones, is the most general and can be applied to different classes of systems.

Elementary Widening principle

In [6] we have introduced an elementary widening technique. Given a regular language ϕ and a regular relation R, the principle of our method was to compare both ϕ and $R(\phi)$. If we detect the existence of three languages ϕ_1, ϕ_2 and Δ such that $\phi = \phi_1.\phi_2$ and $R(\phi) = \phi_1.\Delta.\phi_2$, then we guess that, at each step, the effect of the application of R is the addition of a "Δ" in the middle and we add $\phi_1.\Delta^*.\phi_2$ to the set of reachable states. We continue the application of R to the new set until convergence. At the end, if this procedure terminates, we are sure that the computed set is an upper approximation of $R^*(\phi)$ since all we have done is adding $\phi_1.\Delta^*.\phi_2$. We will show later how this technique can be used to compute exactly $R^*(\phi)$ for each well-founded relation R.

To illustrate this widening principle, let us come back to the example of the Token Ring Protocol given above. We have $\phi = ab^*$ and $R(\phi) = \underline{b}ab^*$, we detect the addition of the letter b to the left of ϕ and compute b^*ab^*. Notice that we have computed the *exact* set $R^*(ab^*) = b^*ab^*$. We can see that the intersection of the reachability set with the bad configurations $(a+b)^*a(a+b)^*a(a+b)^*$ is empty. We have thus shown, using widening techniques, that the Token Ring Protocol is correct. We will see in Section 3 that the widening principles presented in this section allow the model checking of numerous mutual exclusion protocols.

Unary Widening principle

In this paper, we extend the previous widening principle so that we could compute $R^*(\phi)$ (or an upper approximation of it) if we detect several growths between ϕ and $R(\phi)$. Indeed, the applicability of the last widening principle depends on both ϕ and R, since the situation $\phi = \phi_1.\phi_2$ and $R(\phi) = \phi_1.\Delta.\phi_2$ may occur only when R can be applied in a unique position in ϕ as it is the case in the previous example. Let us consider for instance $\phi = a^*ba^*$ and $R = copy(\Sigma^*)(a,c)copy(\Sigma^*)$, then $R(\phi) = a^*\underline{ca^*}ba^* + a^*b\underline{a^*c}a^*$. Our widening principle cannot be used here since there are two different growths (the underlined parts) due to the fact that R can be applied either to the left or to the right of b (i.e., there are two possible positions in ϕ where R can be applied). Thus, we need a more general widening principle.

To force the convergence of the computation, we need to remark that at each step R can add either ca^* to the left of b or a^*c to its right, and guess that the limit will be $a^*\underline{(ca^*)^*}b\underline{(a^*c)^*}a^* = (a+c)^*b(a+c)^*$. A stronger widening principle can then be the following: add $\phi_1\Delta_1^*\phi_2 \ldots \Delta_{n-1}^*\phi_n$ to the reachability set whenever the situation (1) is detected:

$$\phi = \phi_1 \ldots \phi_n \text{ and } R(\phi) = \bigcup_i \phi_1\phi_2 \ldots \phi_i\Delta_i\phi_{i+1} \ldots \phi_n \tag{1}$$

Regular Widening: a general principle

The previous principle is still not powerful enough. Indeed, the situation described above can occur only if R can be applied in several positions in ϕ but can only change a unique position at each step. For instance, if R is binary (can change two positions at a time), then this principle is not valid anymore. Take as an example $R = copy(\Sigma^*)\big((a,c)copy(\Sigma^*)(b,d) + (a,e)copy(\Sigma^*)(b,f)\big)copy(\Sigma^*)$ and $\phi = a^*b^*$, then $R(\phi) = \underline{a^*c}a^*b^*\underline{db^*} + \underline{a^*e}a^*b^*\underline{fb^*}$. We can see that our above principle is not applicable. We need then to strengthen it.

Obviously, the effect of R is always to add one "a^*c" to the left *and* one "db^*" to the right or, one "a^*e" to the left *and* one "fb^*" to the right. It follows that $R^*(\phi)$ is the set $\{w \in \underline{(a^*c + a^*e)}^* a^*b^*\underline{(db^* + fb^*)}^* \mid |w|_c = |w|_d \wedge |w|_e = |w|_f\}$, where $|w|_a$ denotes the number of occurences of the letter a in the word w. Since we are restricted to regular languages (so that we could easily test inclusions and compute unions and intersections), we can consider as an upper approximation of $R^*(\phi)$ the regular language $\underline{(a^*c + a^*e)}^* a^*b^*\underline{(db^* + fb^*)}^*$.

A more general widening principle can then be the following. If this situation holds:

$$\phi = \phi_1\phi_2\cdots\phi_n \text{ and } R(\phi) = \bigcup_i \phi_1.\Delta_{1,i}.\phi_2.\Delta_{2,i}.\phi_3\cdots\phi_{n-1}.\Delta_{n-1,i}.\phi_n. \tag{2}$$

then add $\phi_1(\sum_i \Delta_{1,i})^*\phi_2(\sum_i \Delta_{2,i})^*\phi_3\cdots\phi_{n-1}(\sum_i \Delta_{n-1,i})^*\phi_n$ to the reachability set.

We can extend this principle to the case where ϕ is a finite union of languages of the form $\phi_1\phi_2\cdots\phi_n$. But, before stating our new widening principle, we need to introduce some preliminary definitions:

Definition 2.1 Let ϕ and ϕ' be two regular languages. Then we note $\phi \lhd \phi'$ iff there exist two sequences (ϕ_i^j) and $(\Delta_{k,i,j})$ such that the following holds:

- $\phi = \bigcup_j \phi_1^j\phi_2^j\cdots\phi_n^j$, and
- $\phi' = \bigcup_j\bigcup_i \phi_1^j.\Delta_{1,i,j}.\phi_2^j.\Delta_{2,i,j}.\phi_3^j\cdots\phi_{n-1}^j.\Delta_{n-1,i,j}.\phi_n^j$.

Moreover, we define a binary operator ∇ by:
If $\phi \lhd \phi'$, $\nabla(\phi,\phi') = \bigcup_j \phi_1^j(\sum_i \Delta_{1,i,j})^*\phi_2^j(\sum_i \Delta_{2,i,j})^*\phi_3^j\cdots\phi_{n-1}^j(\sum_i \Delta_{n-1,i,j})^*\phi_n^j$.

Thus, to compute $R^*(\phi)$ we compare ϕ and $R(\phi)$, if $\phi \lhd R(\phi)$ holds then we add $\nabla\big(\phi, R(\phi)\big)$ to the reachability set. The condition $\phi \lhd R(\phi)$ says that $\phi = \bigcup_j \phi_1^j\phi_2^j\cdots\phi_n^j$ and $R(\phi) = \bigcup_j\bigcup_i \phi_1^j\Delta_{1,i,j}\phi_2^j\Delta_{2,i,j}\phi_3^j\cdots\phi_{n-1}^j\Delta_{n-1,i,j}\phi_n^j$. Intuitively, this means that for each indexes k and j, the application of R adds between ϕ_k^j and ϕ_{k+1}^j one language among the sequence $(\Delta_{k,i,j})_i$. We can then conclude that the application of R will always add these languages between the ϕ_k^j's and guess that

$$\bigcup_j \phi_1^j(\sum_i \Delta_{1,i,j})^* \phi_2^j(\sum_i \Delta_{2,i,j})^* \phi_3^j \cdots \phi_{n-1}^j(\sum_i \Delta_{n-1,i,j})^* \phi_n^j$$

is an upper approximation of $R^*(\phi)$.

Remark 2.2 *In fact, by this extrapolation principle, we may loose several informations about the reachability set. Indeed, as seen in the previous example, having* $\phi = \bigcup_j \phi_1^j \phi_2^j \cdots \phi_n^j$ *and* $R(\phi) = \bigcup_j \bigcup_i \phi_1^j \Delta_{1,i,j} \phi_2^j \Delta_{2,i,j} \phi_3^j \cdots \phi_{n-1}^j \Delta_{n-1,i,j} \phi_n^j$ *does not mean that for a given* j *and a given* k*, we can have an arbitrary number of* $\Delta_{k,i,j}$*'s between* ϕ_k^j *and* ϕ_{k+1}^j *since* R *adds at each step for every* j, k *only one* $\Delta_{k,i,j}$ *between* ϕ_k^j *and* ϕ_{k+1}^j*. Consequently, to be closer to the reachability set we must add to* $\bigcup_j \phi_1^j(\sum_i \Delta_{1,i,j})^* \phi_2^j(\sum_i \Delta_{2,i,j})^* \cdots (\sum_i \Delta_{n-1,i,j})^* \phi_n^j$ *the constraints saying that for every* i, j, k, k'*, the number of occurences of* $\Delta_{k,i,j}$ *is equal to the number of occurences of* $\Delta_{k',i,j}$*. By doing this, the set that we compute is not regular anymore. But since we need a regular upper approximation, we drop out the constraints. We can see that this extrapolation is accurate in two cases:*

- *if there is at most one* k *such that* $\Delta_{k,i,j} \neq \epsilon$*, which corresponds to the situation (1) where at each step* R *can be applied in only one position in* ϕ*.*
- *if the languages* $\Delta_{k,i,j}$*'s are downward closed w.r.t. the subword relation. For example,* $\{\big((a+b)^*(c+d)^*\big)^k e^*(g^*f^*)^k \mid k \geq 0\}$ *is equal to* $\big((a+b)^*(c+d)^*\big)^* e^*(g^*f^*)^*$ *since* $(a+b)^*(c+d)^*$ *and* g^*f^* *are downward closed w.r.t. the subword relation.*

In general, if no extrapolation can be done between ϕ and $R(\phi)$, we can start by computing the first elements of the sequence $\big(R^i(\phi)\big)_{i \geq 0}$ until we find two indexes i_1 and i_2 such that $i_1 < i_2$ and $R^{i_1}(\phi) \lhd R^{i_2}(\phi)$, then we add $\phi' = \nabla\big(R^{i_1}(\phi), R^{i_2}(\phi)\big)$ to the reachability set and continue our exploration by computing $R^{i_2+1}(\phi')$. If it is already included in the reachability set then we are done and the computation terminates, otherwise, we check whether the widening can be applied, ...etc. For instance, consider $R = \mathit{copy}(c^*)\big((b,c) + (a,c)\big)\mathit{copy}(a^*)$ and $\phi = ba^*$, then $R(\phi) = ca^*$, and $R^2(\phi) = cca^*$, we see that between ϕ and $R(\phi)$ no extrapolation can be done whereas between $R(\phi)$ and $R^2(\phi)$ we can extrapolate and obtain $R^+(\phi) = c^+a^*$.

2.2 Exact Widening

In this section we show how to use our widening techniques to compute the *exact* reachability set $R^*(\phi)$ for well-founded regular relations.

Definition 2.3 A relation R is well-founded if there is no infinite sequence of words $w_0, w_1, \ldots$ such that for every $i \geq 0$, $(w_{i+1}, w_i) \in R$.

Proposition 2.4 *[13] If R is well-founded then $\phi' = R^*(\phi)$ iff $\phi' = R(\phi') \cup \phi$.*

Thus, if R is well-founded, we can use our widening techniques to guess the reachability set and apply then the test given above to check whether the computed set is exactly equal to the reachability one. Hence, after the detection of the situation where $\phi \lhd R(\phi)$ holds, we shall check if R is well-founded whether

$$R\Big(\nabla\big(\phi, R(\phi)\big)\Big) \cup \phi = \nabla\big(\phi, R(\phi)\big) \tag{3}$$

holds and add $\nabla\big(\phi, R(\phi)\big)$ to the reachability set only if this equality is satisfied. By doing this we are sure that we have computed the *exact* reachability set.

2.3 Computation of transitive closures

As mentioned in [6], widening combined with test (3) can be used to construct the transitive closure R^+ of a regular relation R. To do that, it suffices to consider R as a language over $\Sigma^* \times \Sigma^*$ and define the relation $\omega_R = \{\big((w, w_1), (w, w_2)\big) \in (\Sigma^* \times \Sigma^*)^2 \mid (w_1, w_2) \in R\}$. Since $R^+ = \omega_R^*(R)$, our previous method can be applied to compute R^*. It is not hard to see that if R is well-founded than so is ω_R. Consequently, we can state the following proposition which results from the Proposition 2.4:

Proposition 2.5 *Computing the closure R^* by widening with test (3) is exact if R is well-founded.*

Example 2.6 Consider $\Sigma = \{a, b, c\}$ and the regular relation

$$R = copy(a^*)(a, b)copy(\Sigma^*)$$

We are going to illustrate the use of the widening techniques to construct the relation R^+. First, we compute $\omega_R(R) = copy(a^*)(a, b)copy(a^*)(a, b)copy(\Sigma^*)$, we detect that $\omega_R(R) = \Delta.R$ where $\Delta = copy(a^*)(a, b)$, and we conclude that $R^+ = \Delta^*.R = \big(copy(a^*) + (a, b)\big)^* copy(a^*)(a, b)copy(\Sigma^*)$. Indeed, it can be checked that the exactness test (3) succeeds and since R is well-founded, the relation produced equals exactly R^+.

2.4 Nested Widening

Let $\phi_0 = ba^*$ and $R = \{R_1, R_2, R_3\}$ [4] where:

- $R_1 = copy(a^*(b + c)(b + c)^*)(a, b)copy(a^*)$,
- $R_2 = copy(a^*)(b, c)copy(a + b + c)^*$,
- $R_3 = copy(a + b + c)^*(c, a)copy(a + b + c)^*$.

[4] This example is taken from the Bakery Algorithm's model [19].

Then, using our widening techniques, we can apply the different meta-transitions R_i^* and obtain:

$$\underbrace{ba^*}_{\phi_0} \xrightarrow{R_1^*} \underbrace{bb^*a^*}_{\phi_1} \xrightarrow{R_2} \underbrace{cb^*a^*}_{\phi_2} \xrightarrow{R_3} \underbrace{ab^*a^*}_{\phi_3} \xrightarrow{R_1^*} \underbrace{abb^*a^*}_{\phi_4} \xrightarrow{R_2} \underbrace{acb^*a^*}_{\phi_5} \xrightarrow{R_3} \underbrace{\underline{a}ab^*a^*}_{\phi_6} \xrightarrow{R_1^*} \cdots$$

Obviously, this computation does not terminate since the composed relation $R_3R_2R_1^*$ will always add the letter "a" to the left. What we need here is the computation of the effect of the iteration of the sequence $R_3R_2R_1^*$. We must extrapolate between $\phi_3 = ab^*a^*$ and $\phi_6 = \underline{a}ab^*a^*$ and compute, in one step, $(R_3R_2R_1^*)^*(\phi_3) = a^*ab^*a^*$. The computation will then terminate.

We show in this section how to use our widening principle, during the exploration of the set of reachable states, to compute the effect of the iteration of a sequence of relations. Given n regular relations $R_1, R_2, \ldots, R_n$ and a regular language ϕ, we want to compute $(R_1^*R_2^* \cdots R_n^*)^*(\phi)$ (or an upper approximation of it). For that, it suffices to consider the relation $R = R_1^*R_2^* \cdots R_n^*$ and to apply our previous results.

If we want to compute the *exact* reachability set (in the case where all the R_i's are well-founded), we need to check if the following equality $(R_1^* \cdots R_n^*)(\phi') \cup \phi = \phi'$ holds, where $\phi' = \nabla\big(\phi, (R_1^* \cdots R_n^*)(\phi)\big)$. To do that, we encounter two main difficulties:

(i) The first one is that to perform this test, we need to have the relations R_i^*, but even if the widening techniques have allowed the computation of $(R_1^*R_2^* \cdots R_n^*)^*(\phi)$, they may fail to characterize the relation R_i^* for some i in $\{1, \ldots, n\}$.

(ii) The second one is that the relation $R_1^* \cdots R_n^*$ is not well-founded since $copy(\Sigma^*) \subseteq R_1^* \cdots R_n^*$ and $copy(\Sigma^*)$ is not well-founded (for all $w \in \Sigma^*, (w, w) \in copy(\Sigma^*)$). Thus, even if the test succeeds we are not sure of the exactness of the result.

However, we can overcome these two problems by noticing that $R^* = (R_1^* \cdots R_n^*)^* = (R_1 + \cdots + R_n)^*$ and that the relation $R_1 + \cdots + R_n$ is computable. Thus, if all the R_i's are well-founded, then after the detection of the situation $\phi \lhd (R_1^* \cdots R_n^*)(\phi)$, we can check whether the relation $R_1 + \cdots + R_n$ is also well-founded. If it is the case, we perform the test using the relation $R_1 + \cdots + R_n$ instead of $R_1^*R_2^* \cdots R_n^*$. If the equation $(R_1 + \cdots + R_n)(\phi') \cup \phi = \phi'$ is satisfied, we add ϕ' to the reachability set.

Example 2.7

Let $R_1 = copy(\Sigma^*)(ac, ca)copy(\Sigma^*)$, $R_2 = copy(\Sigma^*)(ad, da)copy(\Sigma^*)$ and $\phi = (a+b)^*(c+d+e)^*(a+b)^*f^*(c+d+e)^*$. It is easy to see that:

$$\begin{aligned} R_1^*(\phi) = {} & (a+b)^*(a+c)^*(c+d+e)^*(a+b)^*f^*(c+d+e)^* \\ & + (a+b)^*(a+c)^*(c+d+e)^*(a+b)^*(a+c)^*(c+d+e)^* \end{aligned}$$

and that

$$R_2^*\big(R_1^*(\phi)\big) = (a+b)^*\underline{(a+c)^*(a+d)}^*(c+d+e)^*(a+b)^*f^*(c+d+e)^*$$
$$+(a+b)^*\underline{(a+c)^*(a+d)}^*(c+d+e)^*(a+b)^*\underline{(a+c)^*(a+d)}^*(c+d+e)^*$$

By comparing $R_2^*\big(R_1^*(\phi)\big)$ and ϕ, we detect the following situation:

$$\phi = \underbrace{(a+b)^*}_{\phi_1^1}\underbrace{(c+d+e)^*(a+b)^*}_{\phi_2^1}\underbrace{f^*(c+d+e)^*}_{\phi_3^1}$$
$$+\underbrace{(a+b)^*}_{\phi_1^2}\underbrace{(c+d+e)^*(a+b)^*}_{\phi_2^2}\underbrace{(c+d+e)^*}_{\phi_3^2}$$

and

$$R_2^*\big(R_1^*(\phi)\big) = \phi_1^1(a+c)^*(a+d)^*\phi_2^1\epsilon\phi_3^1 + \phi_1^2(a+c)^*(a+d)^*\phi_2^2(a+c)^*(a+d)^*\phi_3^2.$$

By applying our widening principle, we compute the set

$$(a+b)^*\underline{(a+c+d)}^*(c+d+e)^*(a+b)^*f^*(c+d+e)^* +$$
$$(a+b)^*\underline{(a+c+d)}^*(c+d+e)^*(a+b)^*\underline{(a+c+d)}^*(c+d+e)^*$$

which is exactly equal to $(R_2^*R_1^*)^*(\phi)$ since $(a+c)^*(a+d)^*$ is downward closed w.r.t. the subword relation (Remark 2.2).

2.5 Detecting Widening situations

The widening techniques described above can be completely automated if there exists an automatic mechanism that detects, given two languages ϕ and ϕ', the situation where

$$\phi = \bigcup_j \phi_1^j\phi_2^j\cdots\phi_n^j \text{ and } \phi' = \bigcup_j\bigcup_i \phi_1^j\Delta_{1,i,j}\phi_2^j\Delta_{2,i,j}\phi_3^j\cdots\phi_{n-1}^j\Delta_{n-1,i,j}\phi_n^j$$

and computes these different languages. In the following, we are going to describe such a mechanism. Let $\mathcal{A}$ (resp. $\mathcal{A}'$) be the automaton representing the language ϕ (resp. ϕ'), and let Q (resp. Q') be the set of states of $\mathcal{A}$ (resp. the set of states of $\mathcal{A}'$). q_0 (resp. q_0') is the initial state of $\mathcal{A}$ (resp. the initial state of $\mathcal{A}'$), and q_F (resp. q_F') is the final state of $\mathcal{A}$ (resp. the final state of $\mathcal{A}'$).

Intuitively, we need to cut the automaton $\mathcal{A}$ into parts that delimit the ϕ_i^j's. This can be done by finding the states of Q that delimit these different parts of the automaton $\mathcal{A}$. To that end, we build a kind of synchronous product $\mathcal{P}$ between $\mathcal{A}$ and $\mathcal{A}'$ as follows.

The states of $\mathcal{P}$ are in $Q \times Q'$. We start from the state (q_0, q_0') and progress simultaneously in both $\mathcal{A}$ and $\mathcal{A}'$. Then, for each reachable state (q, q'), we can choose non-deterministically to continue progressing in both $\mathcal{A}$ and $\mathcal{A}'$ (if we can) or to stop moving in $\mathcal{A}$ and continue progressing in $\mathcal{A}'$. To every path without circuits in $\mathcal{P}$, we associate the sequence of states $(q_i, q_i')_i$ such that:

- $p = (q_0, q'_0) \xrightarrow{*} (q_1, q'_1) \xrightarrow{*} (q_1 = q_2, q'_2) \xrightarrow{*} (q_3, q'_3) \xrightarrow{*} (q_4 = q_3, q'_4) \xrightarrow{*} \cdots \xrightarrow{*} (q_F, q'_F)$,
- the path between (q_0, q'_0) and (q_1, q'_1) is followed simultaneously by both $\mathcal{A}$ and $\mathcal{A}'$,
- from (q_1, q'_1) only $\mathcal{A}'$ progresses until it reaches the state q'_2,
- for every $k > 0$, from $(q_{2k-1} = q_{2k}, q'_{2k})$, the two automata start again moving together in a synchronised way until they reach the state (q_{2k+1}, q'_{2k+1}) from which only $\mathcal{A}'$ moves, ..., etc until we reach the state (q_F, q'_F).

These states related to one possible path in $\mathcal{P}$ correspond to an index j and an index i such that, for $k \geq 1$ we have:

- ϕ^j_k is the language represented by the automaton $\mathcal{P}$ with $(q_{2(k-1)}, q'_{2(k-1)})$ as initial state and (q_{2k-1}, q'_{2k-1}) as final state.
- $\Delta_{k,i,j}$ is the language represented by the automaton $\mathcal{P}$ with (q_{2k-1}, q'_{2k-1}) as initial state and (q_{2k-1}, q'_{2k}) as final state.

Note that this method allows the computation of all the possible decompositions of the required form of ϕ and ϕ'.

3 Applications: Mutual Exclusion Protocols

We have implemented our widening principle in a tool based on MONA [14]. For the time being, only the elementary principle has been considered. Regular languages (resp. regular relations) are represented by finite automata (resp. regular transducers) using the library of MONA. This tool has been used to model check in a fully automatic way several mutual exclusion protocols, namely the Burns, the Dijkstra, the Szymanski, the Bakery and the Token Ring Protocols.

In this section, we illustrate the application of the widening techniques to analyse the Burns Algorithm [17]. We show that this protocol cannot be analysed by the principle intoduced in [6] since it requires the use of the nested widening principle.

The Burns Algorithm can be modelled by the relations given in figure 1 (see the full version of the paper for more details [20]).

R_1: $\mathit{copy}(c_1^*)(c_1, c_2)\mathit{copy}(\Sigma^*)$,
R_2: $\mathit{copy}(\Sigma^*(c_2 + c_3 + c_4)\Sigma^*)(c_2, c_1)\mathit{copy}(\Sigma^*)$,
R_3: $\mathit{copy}(c_1^*)(c_2, c_3)\mathit{copy}(\Sigma^*)$,
R_4: $\mathit{copy}(\Sigma^*)(c_3, c_4)\mathit{copy}(c_1^*)$,
R_5: $\mathit{copy}(\Sigma^*)(c_4, c_1)\mathit{copy}(\Sigma^*)$.

Figure 1. Relations representing Burns algorithm

Using our widening techniques, we can compute the transitive closures of the previous relations since they are all well-founded:

- $R_1^+ = copy(c_1^*)(c_1, c_2)\big(copy(c_1) + (c_1, c_2)\big)^* copy(\Sigma^*)$,
- $R_2^+ = copy(\Sigma^*(c_2 + c_3 + c_4)\Sigma^*)\big(copy(\Sigma) + (c_2, c_1)\big)^+$,
- $R_3^+ = R_3 = copy(c_1^*)(c_2, c_3)copy(\Sigma^*)$,
- $R_4^+ = R_4 = copy(\Sigma^*)(c_3, c_4)copy(c_1^*)$,
- $R_5^+ = copy(\Sigma^*)\big(copy(\Sigma) + (c_4, c_1)\big)^+$.

Let us then try to compute the reachability set without applying the nested widening: $\underbrace{c_1^*}_{\phi_0} \stackrel{R_1^*}{\to} \underbrace{(c_1 + c_2)^*}_{\phi_1} \stackrel{R_3}{\to} \underbrace{c_1^* c_3(c_1 + c_2)^*}_{\phi_2} \stackrel{R_1^*}{\to} \underbrace{(c_1 + c_2)^* c_3(c_1 + c_2)^*}_{\phi_3} \stackrel{R_3}{\to} \cdots$

We notice that the computation will not terminate since the sequence $R_3R_1^*$ will always add $c_3(c_1 + c_2)^*$ to the right. We must then extrapolate between ϕ_0=c_1^* and ϕ_2=$R_3R_1^*(\phi_0)$= $c_1^*\underline{c_3(c_1 + c_2)}^*$ by applying our nested widening principle. The computation terminates: $c_1^* \stackrel{R_1^*}{\to} (c_1 + c_2)^* \stackrel{R_3}{\to} c_1^*c_3(c_1 + c_2)^* \stackrel{(R_3R_1^*)^*}{\to}$ $(c_1 + c_2 + c_3)^* \stackrel{R_4}{\to} (c_1 + c_2 + c_3)^*c_4c_1^* \stackrel{R_5}{\to} (c_1 + c_2 + c_3)^*c_1c_1^* \subseteq (c_1 + c_2 + c_3)^*$.

We can then check that the Burns algorithm satisfies mutual exclusion since the intersection of the reachability set and the bad configurations $\Sigma^*c_4\Sigma^*c_4\Sigma^*$ is empty.

4 Completeness Results

In this section, we illustrate the power of our method. We show that it simulates the computation of R^* given in [1], and prove that it computes $R^*(\phi)$ for every APC language ϕ and every semi-commutation relation R. We point out that these two computations require the use of the general widening principle and that the elementary one introduced in [6] is not sufficient in these two cases.

4.1 Simulation of the construction of [1]

In [1], Abdulla et al. have introduced a class of unary regular relations, that we will call *context-relations* hereafter, for which they characterize R^* by a finite transducer.

In this section, we show that our widening techniques allow the computation of the transducer R^* for every context-relation R.

Definition 4.1 A context-relation is a relation of the form

$$R = copy(\phi_L)R_0copy(\phi_R)$$

such that:

- ϕ_L is a regular language which can be accepted by a deterministic finite-state automaton with a unique accepting state, and where all outgoing transitions from the accepting state are self-loops.
- ϕ_R is a language such that its reverse language ϕ'_R satisfies the above conditions.
- $R_0 = \{(a_i, b_i) \in \Sigma \times \Sigma \mid \forall i, j, a_i \neq b_j\}$.

Theorem 4.2 *The widening techniques allow the computation of R^* for every context-relation R.*

Proof We only sketch the proof idea (see the full version of the paper [20] for the complete proof). We compute the transducers of R, R^2, R^3 and R^4 successively. We detect several growths between R^3 and R^4. Our new widening principle yields *exactly* the transducer R^* given in [1].

□

4.2 Closure of APCs under semi-commutations

Definition 4.3 Let Σ be a finite alphabet. An atomic expression over Σ is either a letter a of Σ or a star expression $(a_1 + a_2 + \cdots + a_n)^*$, where $a_1, a_2, \ldots, a_n \in \Sigma$. A product p over Σ^* is a (possibly empty) concatenation $e_1 e_2 \cdots e_n$ of atomic expressions $e_1, e_2, \ldots, e_n$ over Σ. An Alphabetic Pattern Constraints (APC) over Σ^* is an expression of the form $p_1 + \cdots + p_n$, where $p_1, \ldots, p_n$ are products over Σ^*. A single semi-commutation relation is a relation of the form $copy(\Sigma^*)(ab, ba)copy(\Sigma^*)$. A semi-commutation relation is a finite union of single semi-commutation relations.

In the following, we are going to show that the widening techniques are able to compute the closure of APCs under semi-commutations. To obtain exact results, we use widening with test (3). The proof can be found in the full version of the paper [20].

Theorem 4.4 *$R^*(\phi)$ can be computed by widening for every* APC *language ϕ and every semi-commutation relation R.*

The example 2.7 above illustrates this theorem.

5 Conclusion

We have described a technique called *regular widening* that allows, when it terminates, the computation of an upper approximation of the reachability set of a linear parametrized system. More precisely, our technique allows the computation of an upper approximation of either R^* or $R^*(\phi)$ for a regular relation R and a regular language ϕ. We have shown that our method can be uniformly applied to several parametrized systems. Moreover, we have shown that our technique is powerful enough to simulate several existing constructions of either R^* or $R^*(\phi)$ such as those presented in [1] and [7].

Our approach offers a promising direction in the automatic verification of infinite-state systems. Indeed, the widening mechanism advocated in this paper can be used as an acceleration technique in the analysis of systems having symbolic representation structures based on languages, such as pushdown systems and fifo-channel systems.

In this paper we have only considered the model checking of safety properties since they can be naturally reduced to a reachability problem. But our method can be applied to model check ω-regular properties [6] since it can, in some cases, be used to compute R^*.

In Remark 2.2, we have seen that to characterize precisely the reachability set, we need to consider arithmetical constraints. It would then be interesting to extend this work beyond the regular framework by considering nonregular representations of sets of configurations such as CQDDs [5] that combine finite-state automata with arithmetical constraints. On the other hand, our technique can be applied only in the analysis of *linear* parametrized systems. An other possible direction for future research is to generalize our widening mechanism to more general networks where processes are arranged in a tree or a grid architecture, for example.

Acknowledgments.

I would like to thank Ahmed Bouajjani for reading and commenting several versions of this paper.

References

[1] P. A. Abdulla, A. Bouajjani, B. Jonsson, and M. Nilsson. Handling global conditions in parametrized system verification. *Lecture Notes in Computer Science*, 1633:134–150, 1999.

[2] K .R. Apt and D. C. Kozen. Limits for automatic verificarion of infinite-state concurrent systems. *Information Processing Letters*, pages 22:307–309, 1986.

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

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

[5] A. Bouajjani and P. Habermehl. Symbolic Reachability Analysis of FIFO-Channel Systems with Nonregular Sets of Configurations. In *ICALP'97*. LNCS 1256, 1997.

[6] A. Bouajjani, B. Jonsson, M. Nilsson, and T. Touili. Regular model checking. In *12th Intern. Conf. on Computer Aided Verification (CAV'00)*. LNCS, Springer-Verlag, 2000.

[7] A. Bouajjani, A. Muscholl, and T. Touili. Permutation Rewriting and Algorithmic Verification. In *Proc. 17th Symp. on Logic in Computer Science (LICS'01)*. IEEE, 2001.

[8] D. Caucal. On Word Rewriting Systems Having a Rational Derivation. In *FoSSaCS 2000*, number 1784, page 48. Lecture Notes in Computer Science, 2000.

[9] 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.

[10] J. Esparza, A. Finkel, and R. Mayr. On the Verification of Broadcast Protocols. In *14th IEEE Symposium on Logic in Computer Science (LICS'99). Trento, Italy*. IEEE, 1999.

[11] E. A. Emerson and K. S. Namjoshi. Reasoning about rings. In *Proc. 22th ACM Symposium on Principles of Programming Languages, POPL'95, San Francisco*, January 1995.

[12] E. A. Emerson and K. S. Namjoshi. Automatic verification of parametrized synchronous systems. In *Proc. 8th International Conference on Computer Aided Verification, CAV'96*, Rutgers (N.J.), 1996.

[13] L. Fribourg and H. Olsen. Reachability sets of parametrized rings as regular languages. In *Infinity'97.* volume 9 of Electronical Notes in Theoretical Computer Science. Elsevier Science, 1997.

[14] J. G. Henriksen, J. Jensen, M. Joergensen, and N. Klarlund. MONA: Monadic second-order logic in practice. *Lecture Notes in Computer Science*, 1019:89–100, 1995.

[15] B. Jonsson and M. Nilsson. Transitive closures of regular relations for verifying infinite-state systems. In *6th TACAS (TACAS 00).* in LNCS, Springer-Verlag, 2000.

[16] Y. Kesten, O. Maler, M. Marcus, A. Pnueli, and E. Shahar. Symbolic model checking with rich assertional languages. In O. Grumberg, editor, *Proc. CAV'97*, volume 1254 of *LNCS*, pages 424–435. Springer, June 1997.

[17] N. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers, Inc, 1996.

[18] A. Pnueli and E. Shahar. Liveness and acceleration in parametrized verification. In *12th Intern. Conf. on Computer Aided Verification (CAV'00).* LNCS, Springer-Verlag, 2000.

[19] T. Touili. Vérification de réseaux paramétrés basée sur des techniques de réécriture. MSc. Thesis (french DEA) Report, Liafa Lab., University of Paris7, July 2000. `http://verif.liafa.jussieu.fr/~touili/`

[20] T. Touili. Regular Model Checking using Widening Techniques. Research Report, Liafa Lab., May 2001. `http://verif.liafa.jussieu.fr/~touili/`

[21] Pierre Wolper and Bernard Boigelot. Verifying systems with infinite but regular stae spaces. In *Proc. 10th Int. Conf. on Computer Aided Verification, CAV'98*, volume 1254 of *Lecture Notes in Computer Science*, pages 88–97. Springer Verlag, 1998.