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

*Regular Model Checking using Widening Techniques*

*Tayssir Touili 1*

*LIAFA, Universit e 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 con gurations are represented by regular languages and actions by regular relations. Since the veri cation problem amounts to the computation of the reachability set, we focus on the computation of R( ) for a regular relation R and a regular language . We present a technique called regular widening that allows, when it terminates, the computation of either the reachability set R( ) 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( ) for each R (resp. ) belonging to a subclass of regular relations (resp. belonging to a subclass of regular languages).*

# *1 Introduction*

*Many protocols dealing with distributed systems are de ned 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 sys- tem. 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 in nite union of similar* systems Sn composed of n indistinguishable processes organized in a linear topology. Since safety properties express that some \dangerous" con gura- tions cannot be reached by the system, the standard way to model check such

*1 Email:* [*Tayssir.Touili@liafa.jussieu.fr*](mailto:Tayssir.Touili@liafa.jussieu.fr)

*c 2001 Published by Elsevier Science B. V. Open access under* [*CC BY-NC-ND license.*](http://creativecommons.org/licenses/by-nc-nd/3.0/)

*properties is to compute the set of all reachable con gurations 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 nd a nite symbolic representation of in nite sets of states of a system.

*Following [16,1,15,6], we use regular languages to represent in nite sets of* con gurations: we encode the global state of a system Sn into a word of length n by concatenating the local states of the di erent processes. Consequently, a set of con gurations 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; w0) 2 R means that the considered action transforms the global state of the system from w to w0.

*In this framework, to compute the reachability set of a given parametrized* system whose actions are represented by the relations fR1;::: ; Rng, the stan- dard algorithm starts from 0, the set of initial con gurations of the system, and produces at each step the set of successors corresponding to a single ap- plication of one relation Ri. However, this naive algorithm will not terminate in the case of parametrized systems since the reachability set is in nite. To solve this problem we use acceleration techniques [21,1,15,6,18] that consist in applying at each step the operation R instead of Ri, in order to speed up the

*i*

*termination of the computation.*

*The veri cation problem is then reduced to the computation of R( ) for* a regular language and a regular relation R, where R is the transitive- re exive closure of R. This problem being in general undecidable we need to

*nd subclasses R of regular relations such that we can e ectively characterize*

*R (e.g., by a transducer) for every relation R in R.*

*But sometimes, computing R( ) for some regular language is easier than* characterizing R (especially when R is not regular). Take as an example the relation R corresponding to the rewriting rule ab $ ba, and the language

*= ab, then it is easy to see that R( ) equals (a+b) whereas R cannot be* characterized by a nite transducer since it is not regular (R((ab)) = fw 2

*j jwja = jwjbg). Consequently, we need to nd subclasses R of regular* relations and subclasses L of regular languages such that we can e ectively compute R( ) for every relation R in R and every language in L.

*Several works have been done in this direction [8,4,3,5]. In particular, Ab-* dulla et al.[1] have introduced a class of unary regular relations 2 for which they characterize R by a nite transducer. In [7], we have identi ed a sub- class of regular languages called Alphabetic Pattern Constraints (APC) which can naturally be used to model parametrized systems such as mutual exclu- sion protocols. We have provided an algorithm that computes R( ) for every APC language and every semi-commutation relation R (i.e., a relation cor- responding to a set of rewriting rules of the form ab ! ba). This kind of

*2 unary relations are relations that change only one position in a given word*

*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 con guration.

*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( ) for a regular language and a regular relation R. The principle of our method consists in guessing automatically, by comparing and R( ), the limit of the iteration of R: if the situation = 1: 2 and R( ) = 1: : 2 holds, then we guess that R will always introduce a \ " in the middle and add 1:: 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 x- points [9]. However, contrary to the notion of widening operator de ned 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 . 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 . 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 reach- ability set, to compute the e ect 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( ) for each R (resp. ) belonging to* a subclass of regular relations (resp. belonging to a subclass of regular lan- guages). Indeed, we prove that it simulates the construction of R presented in

*[1] and allows the exact computation of R( ) for every APC language 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 veri cation 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( ) for every APC language and every semi-commutation relation R.

# *2 Regular Widening Techniques*

*Given a regular language and a regular relation R, our aim is to com-* pute R( ) = Ri( ). If we proceed naively, i.e., if we start by computing

*S*

*i 0*

*R( ),R2( ), R3( ), ::: etc, until we nd an index k such that Rk+1( ) Ri( ) we will certainly not terminate in all non trivial cases. Take as an*

*S*

*i k*

*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 con gurations are then represented by the expression = ab. The action that consists in passing the token from the left to the right can be repre- sented by the relation R = copy()(ab; ba)copy() where for each L 2, copy(L) denotes the set f(x; x) 2 ( ) j x 2 Lg. Let us try to compute the reachability set R( ) of this system. It can be seen that for each k 0, Rk( ) = bkab, and thus there exists no index k such that Rk+1( ) Ri( )

*S*

*i k*

*and our computation will never converge. What we need here is an automatic*

*method that allows the computation of the limit R( ) = bab. 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 e ect of the iteration of a regular relation an arbitrary number of times. This mechanism is based on the comparison of and R( ) and on the detection of some patterns that will be added at each application of R and which will nally produce loops in the limit language. To show the intuition, we will present gradually di erent widening principles: the rst one is the most elementary and can be used only in the cases where R and are such that R can be applied in a unique position in . The second widening principle is useful when 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 di erent classes of systems.

*Elementary Widening principle*

*In [6] we have introduced an elementary widening technique. Given a regular* language and a regular relation R, the principle of our method was to compare both and R( ). If we detect the existence of three languages 1; 2 and such that = 1: 2 and R( ) = 1: : 2, then we guess that, at

*each step, the e ect of the application of R is the addition of a \ " in the* middle and we add 1:: 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( ) since all we have done is adding 1: 2. We will show later how this

*technique can be used to compute exactly R( ) 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 = ab and R( ) =

*bab , we detect the addition of the letter b to the left of and compute* bab. Notice that we have computed the exact set R(ab) = bab. We can see that the intersection of the reachability set with the bad con gurations (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( ) (or an upper approximation of it) if we detect several growths between and R( ). Indeed, the applicability of the last widening principle depends on both and R, since the situation = 1: 2 and R( ) = 1: : 2 may occur only when R can be applied in a unique position in as it is the case in the previous example. Let us consider for instance = aba and R = copy()(a; c)copy(), then R( ) = acaba + abaca. Our widening principle cannot be used here since there are two di erent growths (the under- lined 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 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 ac to its right, and guess that the limit will be a(ca)b(ac)a = (a + c)b(a + c). A stronger widening

*principle can then be the following: add 1 2 ::: n to the reachability*

*1*

*set whenever the situation (1) is detected:*

*n 1*

*= 1 ::: n and R( ) = [ 1 2 ::: i i i+1 ::: n (1)*

*i*

*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

*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(c + e) (a; c)copy()(b; d) + (a; e)copy()(b; f ) copy(d + f ) and = ab, then R( ) = cabd + eabf . We can see that our above principle is not applicable. We need then to strengthen it.*

*Obviously, the e ect of R is always to add one \c" to the left and one \d"* to the right or, one \e" to the left and one \f " to the right. It follows that R( ) is the set fw 2 (c + e)ab(d + f ) j jwj = jwj ^ jwj = jwj g, where jwja 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( ) the regular language (c + e)ab(d + f ).

*c d e f*

*A more general widening principle can then be the following. If this situ-* ation holds:

*= 1 2 n and R( ) = [ 1: 1;i: 2: 2;i: 3 n 1: n 1;i: n: (2)*

*i*

*then add 1(P 1;i) 2(P 2;i) 3 n 1(P n 1;i) n to the reachability*

*i i i*

*set.*

*We can extend this principle to the case where is a nite union of lan-* guages of the form 1 2 n. But, before stating our new widening principle, we need to introduce some preliminary de nitions:

*De nition 2.1 Let and 0 be two regular languages. Then we note C 0*

*i there exist two sequences ( j) and (*

*) such that the following holds:*

*i*

*= S j j j , and*

*k;i;j*

*1 2 n*

*j*

*0 = SS j : 1*

*: j : 2*

*: j j*

*: 1*

*: j .*

*1 ;i;j 2*

*j i*

*;i;j 3*

*n 1*

*n ;i;j n*

*Moreover, we de ne a binary operator r by:*

*If C 0, r( ; 0) = S j (P 1*

*) j (P 2*

*) j j*

*(P 1*

*) j .*

*1 ;i;j 2*

*j i i*

*;i;j 3*

*n 1*

*i*

*n ;i;j n*

*Thus, to compute R( ) we compare and R( ), if C R( ) holds then* we add r ; R( ) to the reachability set. The condition C R( ) says

*that = S j j j and R( ) = SS j 1 j 2 j j 1 j .*

*1 2 n*

*j*

*1 ;i;j 2*

*j i*

*;i;j 3*

*n 1*

*n ;i;j n*

*Intuitively, this means that for each indexes k and j, the application of R adds*

*k+1*

*between j*

*k*

*and j*

*one language among the sequence (*

*k;i;j*

*)i. We can then*

*conclude that the application of R will always add these languages between* the j 's and guess that

*k*

*[ j (X 1*

*) j (X 2*

*) j j*

*(X 1*

*) j*

*1 ;i;j 2*

*j i i*

*;i;j 3*

*n 1*

*i*

*n ;i;j n*

*is an upper approximation of R( ).*

*Remark 2.2 In fact, by this extrapolation principle, we may loose several in-* formations about the reachability set. Indeed, as seen in the previous example,

*having = S j j j and R( ) = SS j 1 j 2 j j 1 j*

*1 2 n*

*j*

*1 ;i;j 2*

*j i*

*;i;j 3*

*n 1*

*n ;i;j n*

*does not mean that for a given j and a given k, we can have an arbitrary*

*k*

*k+1*

*number of*

*k;i;j*

*'s between j*

*and j*

*since R adds at each step for every j; k*

*only one*

*k;i;j*

*between j*

*and j*

*. Consequently, to be closer to the reach-*

*ability set we must add to S j (P 1*

*k*

*k+1*

*1*

*n*

*;i;j*

*) j (P 2*

*;i;j*

*) (P*

*n 1;i;j*

*) j*

*the*

*j i i i*

*2*

*constraints saying that for every i; j; k; k0, the number of occurences of k;i;j*

*is equal to the number of occurences of k0 ;i;j . By doing this, the set that we* compute is not regular anymore. But since we need a regular upper approx- imation, 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 k;i;j 6= , which corresponds to the* situation (1) where at each step R can be applied in only one position in .

*if the languages k;i;j's are downward closed w.r.t. the subword relation.* For example, f (a + b)(c + d) ke(gf)k j k 0g is equal to (a + b)(c + d) e(gf) since (a + b)(c + d) and gf are downward closed w.r.t. the subword relation.

*In general, if no extrapolation can be done between and R( ), we can*

*start by computing the rst elements of the sequence Ri( )*

*i 0*

*until we nd*

*two indexes i1 and i2 such that i1 < i2 and Ri1 ( ) C Ri2 ( ), then we add*

*0 = r Ri1 ( ); Ri2 ( ) to the reachability set and continue our exploration* by computing Ri2 +1( 0). 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 = copy(c) (b; c)+ (a; c) copy(a) and = ba, then R( ) = ca, and R2( ) = cca, we see that between and R( ) no extrapolation can be done whereas between R( ) and R2( ) we can extrapolate and obtain R+( ) = c+a.

*2.2 Exact Widening*

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

*De nition 2.3 A relation R is well-founded if there is no in nite sequence of words w0; w1;::: such that for every i 0, (wi+1; wi) 2 R.*

*Proposition 2.4 [13] If R is well-founded then 0 = R( ) i 0 = R( 0) [ .*

*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 C R( ) holds, we shall check if R is well-

*founded whether*

*R r ; R( ) [ = r ; R( ) (3)*

*holds and add r ; R( ) to the reachability set only if this equality is satis-*

*ed. 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 con-* struct the transitive closure R+ of a regular relation R. To do that, it suÆces to consider R as a language over and de ne the relation

*!R = f (w; w1); (w; w2) 2 ()2 j (w1; w2) 2 Rg. Since R+ = ! (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 !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 = fa; b; cg and the regular relation*

*R = copy(a)(a; b)copy()*

*We are going to illustrate the use of the widening techniques to construct the* relation R+. First, we compute !R(R) = copy(a)(a; b)copy(a)(a; b)copy(), we detect that !R(R) = :R where = copy(a)(a; b), and we conclude that R+ =:R = copy(a) + (a; b) copy(a)(a; b)copy(). 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 0 = ba and R = fR1; R2; R3g 3 where:*

*R1 = copy(a(b + c)(b + c))(a; b)copy(a),*

*R2 = copy(a)(b; c)copy(a + b + c),*

*R3 = copy(a + b + c)(c; a)copy(a + b + c).*

*Then, using our widening techniques, we can apply the di erent meta-transitions* R and obtain:

*i*

*R R R R R R R*

*ba !1 bba !2 cba !3 aba !1 abba !2 acba !3 aaba !1*

*|{ z0 }*

*|{ z1 }*

*|{ z2 }*

*| { z3 }*

*|{ z4 }*

*|{ z5 }*

*| { z6 }*

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

*Obviously, this computation does not terminate since the composed rela-*

*tion R3R2R*

*1*

*will always add the letter \a" to the left. What we need here*

*is the computation of the e ect of the iteration of the sequence R3R2R. We* must extrapolate between 3 = aba and 6 = aaba and compute, in one step, (R3R2R)( 3) = aaba. The computation will then terminate.

*1*

*1*

*We show in this section how to use our widening principle, during the ex-* ploration of the set of reachable states, to compute the e ect of the iteration of a sequence of relations. Given n regular relations R1; R2;::: ; Rn and a regular

*language , we want to compute (RR R )( ) (or an upper approxima-*

*1 2 n*

*tion of it). For that, it suÆces to consider the relation R = RR R and*

*to apply our previous results.*

*1 2 n*

*If we want to compute the exact reachability set (in the case where all the* Ri's are well-founded), we need to check if the following equality (R R )( 0)[

*1 n*

*= 0 holds, where 0 = r ; (R R )( ) . To do that, we encounter two*

*1 n*

*main diÆculties:*

*(i) The rst one is that to perform this test, we need to have the relations* R, but even if the widening techniques have allowed the computation of (RR R )( ), they may fail to characterize the relation R for some

*i*

*1 2 n i*

*i in f1;::: ; ng.*

*(ii) The second one is that the relation R R*

*is not well-founded since*

*1 n*

*copy() R R and copy() is not well-founded (for all w 2*

*1 n*

*; (w; w) 2 copy()). 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 R )*

*1 n*

*= (R1 + + Rn) and that the relation R1 + + Rn is computable.*

*Thus, if all the Ri's are well-founded, then after the detection of the situ-* ation C (R R )( ), we can check whether the relation R1 + + Rn

*1 n*

*is also well-founded. If it is the case, we perform the test using the relation*

*R1+ +Rn instead of RR R . If the equation (R1 + +Rn)( 0)[ = 0*

*1 2 n*

*is satis ed, we add 0 to the reachability set.*

*Example 2.7*

*Let R1 = copy()(ac; ca)copy(), R2 = copy()(ad; da)copy() and*

*= (a + b)(c + d + e)(a + b)f(c + d + e). It is easy to see that:*

*R( )= (a + b)(a + c)(c + d + e)(a + b)f(c + d + e)*

*1*

*and that*

*+ (a + b)(a + c)(c + d + e)(a + b)(a + c)(c + d + e)*

*R R( ) = (a + b)(a + c)(a + d)(c + d + e)(a + b)f(c + d + e)*

*2*

*1*

*+(a + b)(a + c)(a + d)(c + d + e)(a + b)(a + c)(a + d)(c + d + e)*

*By comparing R R( ) and , we detect the following situation:*

*2*

*1*

*= (a + b) (c + d + e)(a + b) f(c + d + e)*

*|{ z1 }| { z1 }| { z1 }*

*1*

*2*

*3*

*+ (a + b) (c + d + e)(a + b) (c + d + e)*

*and*

*|{ z2*

*1*

*}| { z2*

*2*

*}| { z2 }*

*3*

*R R( ) = 1(a + c)(a + d) 1 1 + 2(a + c)(a + d) 2(a + c)(a + d) 2:*

*2*

*1*

*1*

*2*

*3*

*1*

*2*

*3*

*By applying our widening principle, we compute the set*

*(a + b)(a + c + d)(c + d + e)(a + b)f(c + d + e) + (a + b)(a + c + d)(c + d + e)(a + b)(a + c + d)(c + d + e)*

*which is exactly equal to (RR)( ) since (a + c)(a + d) is downward closed*

*2 1*

*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 and

*0, the situation where*

*= [ j j j*

*and 0 = [[ j 1*

*j 2*

*j j*

*1 j*

*1 2 n*

*j*

*1 ;i;j 2*

*j i*

*;i;j 3*

*n 1*

*n ;i;j n*

*and computes these di erent languages. In the following, we are going to* describe such a mechanism. Let A (resp. A0) be the automaton representing the language (resp. 0), and let Q (resp. Q0) be the set of states of A (resp. the set of states of A0). q0 (resp. q0 ) is the initial state of A (resp. the initial state of A0), and qF (resp. q0 ) is the nal state of A (resp. the nal state of

*0*

*F*

*A0).*

*Intuitively, we need to cut the automaton A into parts that delimit the*

*j's. This can be done by nding the states of Q that delimit these di erent* parts of the automaton A. To that end, we build a kind of synchronous product P between A and A0 as follows.

*i*

*The states of P are in Q Q0. We start from the state (q0; q0 ) and progress* simultanously in both A and A0. Then, for each reachable state (q; q0), we can choose non-deterministically to continue progressing in both A and A0 (if we can) or to stop moving in A and continue progressing in A0. To every path without circuits in P, we associate the sequence of states (qi; q0)i such that:

*0*

*i*

*p = (q ; q0 ) !*

*(q ; q0 ) ! (q = q ; q0 ) ! (q ; q0 ) ! (q*

*= q ; q0 ) !*

*!*

*0 0*

*(qF ; q0 ),*

*F*

*1 1 1 2 2 3 3*

*4 3 4*

*the path between (q0; q0 ) and (q1; q0 ) is followed simultanously by both A*

*0 1*

*and A0,*

*from (q1; q0 ) only A0 progresses until it reaches the state q0 ,*

*1 2*

*for every k > 0, from (q2k 1 = q2k; q0 ), the two automata start again moving*

*2k*

*together in a synchronised way until they reach the state (q2k+1; q0*

*2k+1*

*) from*

*which only A0 moves, :: :, etc until we reach the state (qF ; q0 ).*

*F*

*These states related to one possible path in P correspond to an index j* and an index i such that, for k 1 we have:

*j is the language represented by the automaton P with (q2(*

*k*

*k 1)*

*0*

*2(k 1)*

*; q*

*) as*

*initial state and (q2k 1; q0*

*2k 1*

*) as nal state.*

*k;i;j is the language represented by the automaton P with (q2k 1; q0*

*2k 1*

*) as*

*initial state and (q2k 1; q0*

*2k*

*) as nal state.*

*Note that this method allows the computation of all the possible decom-* positions of the required form of and 0.

# *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 nite 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 gure 1*

*(see the full version of the paper for more details [20]).*

*R1: copy(c)(c1; c2)copy(),*

*1*

*R2: copy((c2 + c3 + c4))(c2; c1)copy(),*

*R3: copy(c)(c2; c3)copy(),*

*1*

*R4: copy()(c3; c4)copy(c),*

*1*

*R5: copy()(c4; c1)copy().*

*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+ = copy(c)(c ;c ) copy(c )+ (c ;c ) copy(),*

*1*

*1*

*1*

*2*

*1*

*1*

*2*

*R+ = copy((c + c + c )) copy( )+ (c ;c ) +,*

*2*

*2*

*3*

*4*

*2*

*1*

*R+ = R3 = copy(c)(c2; c3)copy(),*

*3 1*

*R+ = R4 = copy()(c3; c4)copy(c),*

*4 1*

*R+ = copy() copy( ) + (c ;c ) +.*

*5*

*4*

*1*

*Let us then try to compute the reachability set without applying the nested*

*widening: c*

*R*

*! (c*

*1*

*+ c ) !R3 cc (c*

*R*

*+ c ) !1 (c*

*+ c )c (c*

*+ c ) !R3*

*1 1 2*

*1 3 1 2*

*1 2 3 1 2*

*|{ z0 } | { z1 } | { z2 } | { z3 }*

*We notice that the computation will not terminate since the sequence R3R* will always add c3(c1 + c2) to the right. We must then extrapolate between

*1*

*0=c and 2=R3R( 0)= c c3(c1 + c2) by applying our nested widening prin-*

*1 1 1*

*3 1*

*R*

*(R R )*

*ciple. The computation terminates: c !1 (c + c ) !R3*

*cc (c + c )*

*1 1 2*

*1 3 1 2 !*

*(c + c*

*+ c ) !R4 (c*

*+ c + c )c c !R5 (c*

*+ c + c )c c (c + c*

*+ c ).*

*1 2 3*

*1 2 3 4 1*

*1 2 3 1 1*

*1 2 3*

*We can then check that the Burns algorithm satis es mutual exclusion*

*since the intersection of the reachability set and the bad con gurationsc4c4* is empty.

# *4 Completeness Results*

*In this section, we illustrate the power of our method. We show that it simu-* lates the computation of R given in [1], and prove that it computes R( ) for every APC language 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 suÆcient 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

*nite transducer.*

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

*De nition 4.1 A context-relation is a relation of the form*

*R = copy( L)R0copy( R)*

*such that:*

*L is a regular language which can be accepted by a deterministic nite-state* automaton with a unique accepting state, and where all outgoing transitions from the accepting state are self-loops.

*R is a language such that its reverse language 0 satis es the above con-*

*R*

*ditions.*

*R0 = f(ai; bi) 2 j 8i; j; ai 6= bjg.*

*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, R2, R3 and R4 successively. We detect several growths between R3 and R4. Our new widening principle yields exactly the transducer R given in [1].

*2*

*4.2 Closure of AP Cs under semi-commutations*

*De nition 4.3 Let be a nite alphabet. An atomic expression over is either a letter a of or a star expression (a1 + a2 + + an), where* a1; a2;::: ; an 2 . A product p over is a (possibly empty) concatenation e1e2 en of atomic expressions e1; e2;::: ; en over . An Alphabetic Pattern Constraints (APC) over is an expression of the form p1 + + pn, where p1;::: ; pn are products over. A single semi-commutation relation is a rela- tion of the form copy()(ab; ba)copy(). A semi-commutation relation is a

*nite 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( ) can be computed by widening for every APC language*

*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( ) for a regular relation R and a regular language . 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 construc- tions of either R or R( ) such as those presented in [1] and [7].

*Our approach o ers a promising direction in the automatic veri cation* of in nite-state systems. Indeed, the widening mechanism advocated in this

*paper can be used as an acceleration technique in the analysis of systems hav-* ing symbolic representation structures based on languages, such as pushdown systems and fo-channel systems.

*In this paper we have only considered the model checking of safety prop-* erties since they can be naturally reduced to a reachability problem. But our method can be applied to model check !-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 nonregu- lar representations of sets of con gurations such as CQDDs [5] that combine

*nite-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 veri cation. Lecture Notes in Computer Science, 1633:134{150, 1999.*

*[2] K .R. Apt and D. C. Kozen. Limits for automatic veri carion of in nite-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 Con gurations. 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 Veri cation (CAV'00). LNCS, Springer- Verlag, 2000.*

*[7] A. Bouajjani, A. Muscholl, and T. Touili. Permutation Rewriting and Algorithmic Veri cation. 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 Veri cation 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 veri cation of parametrized synchronous systems. In Proc. 8th International Conference on Computer Aided Veri cation, CAV'96, Rutgers (N.J.), 1996.*

*[13] L. Fribourg and H. Olsen. Reachability sets of parametrized rings as regular languages. In In nity'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 in nite-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 veri cation. In 12th Intern. Conf. on Computer Aided Veri cation (CAV'00). LNCS, Springer-Verlag, 2000.*

*[19] T. Touili. V eri cation de r eseaux param etr es bas ee sur des techniques de r e ecriture. MSc. Thesis (french DEA) Report, Liafa Lab., University of Paris7,*

*July 2000.* [*http://verif.liafa.jussieu.fr/*](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/*](http://verif.liafa.jussieu.fr/) *touili/*

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