EXPLOITING PARTIAL SYMMETRIES IN WELL-FORMED NETS FOR THE REACHABILITY AND THE LINEAR TIME MODEL CHECKING PROBLEMS

Soheib Baarir [**,2] **Serge Haddad** [*,1]
Jean-Michel Ilié [**,2]

[*] *LAMSADE, UMR CNRS 7024, Université Paris Dauphine, Place du Maréchal de Lattre de Tassigny, 75775 PARIS, FRANCE*
[**] *LIP6, Université PARIS VI, 8 rue du Capitaine Scott,75015 PARIS, FRANCE*

Abstract : Taking advantage of the symmetries of a system is an efficient way to cope with the combinatory explosion involved by the verification process. Whereas numerous algorithms and tools efficiently deal with the verification of a symmetrical formula on a symmetrical model, the management of partial symmetries is still an open research topic. In this work, we present the design and the evaluation of two methods applicable on coloured Petri nets. These two methods are extensions of the symbolic reachability graph construction for the well-formed Petri nets. The first algorithm, called the extended symbolic reachability graph construction, tackles the reachability problem. The second one, called the symbolic synchronized product, checks a partially symmetric linear time formula on a net. The evaluations show that these two methods outperform the previous approaches dealing with partial symmetries. Furthermore they are complementary ones since the former while being less general gives better results than the latter when applied to the reachability problem.

Keywords: Coloured Petri Net, Well-Formed Petri Net, Partial Symmetry, Symbolic Reachability Graph

1. INTRODUCTION

The coloured Petri nets formalism is an expressive model extending the representation of concurrency by Petri nets with a data management via the coloured domains and functions. However this expressiveness leads in practice to huge state graphs considerably restricting their use for the reachability and the model checking problems. Therefore, a recurrent research topic is the building of a reduced graph equivalent to the original one w.r.t. some set of properties. Among the proposed approaches, the symmetry based method builds a symbolic reachability graph (SRG) where a node corresponds to a set of states leading to an equivalent behaviour up to some "admissible" colour permutation. In order to be applicable, such a method must detect the admissible permutations by a syntactical examination of the net. This requirement has motivated the introduction of the well-formed nets model which is

[1] haddad@lamsade.dauphine.fr
[2] {Souheib.Baarir,Jean-Michel.Ilie}@lip6.fr

equivalent to the coloured Petri net model but with a restricted syntax allowing the automatic computation of the SRG (Chiola *et al.* 1993). On this reduced graph, one solves the reachability problem and more generally the linear time temporal logic formula truth whenever the atomic propositions of the formula are symmetrical (e.g. $\mathbf{AND}_{c \in C(p)} m(p)(c) = 1$). (Clarke *et al.* 1996) establish the correctness of this checking in a general framework.

However, the previous approach suffers from two limitations. On the one hand, it is well-known that without process identities, many distributed problems do not have solutions. Indeed in distributed algorithms, identities comparisons break deadlock situations. Modelling such algorithms produces nets whose behaviour is symmetrical with the exception of a small set of transitions. Symmetrical methods are not able to efficiently handle partial symmetries since they require a symmetry upon the whole model.

On the other hand, many usual formulas have asymmetric propositions. For instance, the fairness formula "*If some process is waiting for a resource, then it will get it*" implies to distinguish a process (i.e. a colour in the net). By detecting the symmetries of the automaton related to the formula, (Emerson and Prasad Sistla 1996) efficiently handle this kind of formula. Unfortunately this technique fails when analyzing the automaton associated to the asymmetric formula: "*If some process is waiting for a resource then it will get it, provided none of the processes with higher identity will require the resource in the future*". Again viewed as a model, the automaton is partially symmetric. A more refined analysis of the automaton slightly extends the range of application but the main limitations remain (Ajami *et al.* 1998, Emerson and Trefler 1999).

Here, we present the efficient design of two methods for partially symmetric models and/or formulas in well-formed nets. The first method which solves the reachability problem and refines the one presented in (Haddad *et al.* 1995), may be summarized as follows: (1) partition the transitions in two subsets, the symmetrical and the asymmetrical ones; (2)build an Extended Symbolic Reachability Graph (ESRG), where each node corresponds to a set of symbolic states of the SRG which are developed as soon as an asymmetrical transition is fireable; (3) fire symbolically the symmetrical transitions from the representant of the node and asymmetrical ones from the developed symbolic markings ; (4) as soon as the symbolic markings associated to a node are no more relevant (i.e. the asymmetrical transitions have been fired and **all** the symbolic markings are reachable), they are deleted.

The second method solves the linear time model checking problem and applies on a symmetric well-formed net and an asymmetric Büchi automaton. It is an *instanciation* of a generic method presented in (Haddad *et al.* 2000). To any state of this automaton is associated a local partition of each colour class such that two colours inside the same partition are equivalent w.r.t. the atomic propositions labelling the state. Then a Symbolic Synchronized Product (SSP) is built where a state of this product is composed by a symbolic marking and a state of the automaton. Contrary to the SRG building, the decomposition of coloured classes into static subclasses is local to each state. Furthermore when building a state, we analyze its structure in order to minimize the decomposition into static subclasses whereas preserving the equivalence with the ordinary synchronized product.

The balance of the remaining paper is the following one. In the second section, we informally present the well-formed Petri nets and the standard model checking methods. Then we depict the principles of our methods. In the forth section, we evaluate them on two examples. At last, we conclude and we give perspectives to this work.

2. PRELIMINARIES

Below, we describe the main features of WFN. The reader can refer to (Chiola *et al.* 1993) for a formal definition :

- In a WFN, a colour domain is a cartesian product of colour classes which may be viewed as primitive domains. This product is possibly empty (e.g. a place which contains neutral tokens) and may include repetitions (e.g. a transition which synchronises two colours inside a class). A class is divided into static subclasses. The colours of a class have the same nature (processes, resources, etc.) whereas the colours inside a static subclass have the same potential behaviour (batch processes, interactive processes, etc.).
- In a WFN, a colour function is built by standard operations (linear combination, composition, etc.) on basic functions. There are three basic functions: a projection which selects an item of a tuple and is denoted by a typed variable (e.g. X, Y); a diffusion, a constant function which returns the bag composed by all the colours of a class or a subclass and is denoted S_C where C is the corresponding (sub)class; and a successor function which applies on an *orderered* class and returns the colour following a given colour.
- In a WFN, transitions are guarded by expressions (in fact even the coloured functions may

be guarded but we do not describe here this feature). An expression is a boolean combination of atomic predicates. An atomic predicate either identifies two variables $[X = Y]$ or restricts the domain of a variable to a static subclass $[X \in D]$.

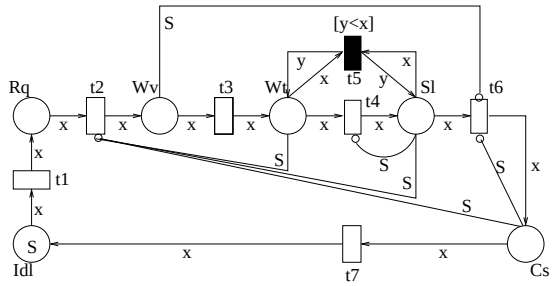

Figure 1. Modelling a distributed algorithm with priorities for a critical section

The WFN of Figure 1 models a distributed algorithm for critical section accesses. There is a single class C which represents the processes. The variables X and Y are typed by C. Since no confusion is possible, the constant S_C has been abbreviated by S. Let us note, for instance, that the colour domain of Rq is C and the colour domain of t_5 is $C \times C$. Roughly speaking, the distributed algorithm builds successive waves of requests (by firings of t_2 until some firing of t_3). Then one decides between the requests of a wave (inside the places Wv, Wt, Sl) according to the order of the processes identities. This tie break is done by the asymmetric transition t_5 which exchanges requests between Wt and Sl. This transition has priority over the other ones, thus t_6 is fireable when no more exchange is possible.

The class C is split into $D_1, \ldots, D_n$ static subclasses (one per colour). This decomposition is required to express the predicate $[Y < X]$ which is an abbreviation of $\mathrm{OR}_{i<j}([Y \in D_i]$ AND $[X \in D_j])$.

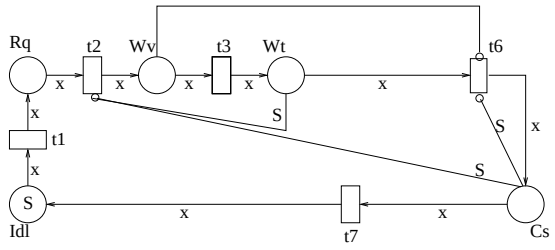

Figure 2. Modelling a distributed algorithm for a critical section

The WFN of Figure 2 represents a similar algorithm except that in a wave, the accesses to the critical section are not controlled. Since the potential behaviours of all processes are equivalent, the class is not splitted (i.e. it is composed by a single static subclass).

The symbolic reachability graph (SRG) building lies on a compact representation for a set of equivalent ordinary markings called a symbolic marking. In a symbolic marking, each static subclass is divided into dynamical subclasses. A dynamical subclass is only specified by its size, thus each consistent choice of colours for the dynamical subclasses leads to an ordinary marking. The implicit colours of a dynamical subclass are supposed to be *in the same state*. Consequently the marking of places is defined w.r.t. to dynamical subclasses instead of colours.

In order to build the SRG, a symbolic firing rule is applied. This rule is similar to the ordinary one except that the dynamical subclasses instanciating the firing must be reduced to singletons by an appropriate splitting. At last, different representations of a symbolic marking are possible. So a canonical one is defined and each time a new symbolic marking is built, it is canonized.

As an illustration, building the SRG of Figure 2 leads to exponential savings of space w.r.t. the ordinary reachability graph (RG). However, the SRG of Figure 1 has exactly the same size of the ordinary graph. Indeed the static decomposition of C forbids any saving by the standard SRG.

We briefly recall the principles of the standard model checking of a linear time temporal logic formula. At first, one builds the Büchi automaton of the negation of the formula. This automaton is then synchronized with the state graph i.e. a state of the synchronized product is composed by a state of the automaton and a state of the reachability graph which fulfills the atomic properties labelling the automaton state. There is an edge in the synchronized product iff the corresponding edges are present in the state graph and in the automaton. If a successful path is found on the synchronized product, the formula is invalidated. Usually, all the constructions are done on-the-fly.

As discussed in the introduction, using the SRG for a partially symmetric formula often requires to define "artificial" static subclasses. For instance, expressing the wave construction of the net of Figure 2 leads to the following LTL formula:

$$\mathrm{G}(Rq_1 \text{ AND } Wt_2 \Rightarrow (\text{NOT } Cs_1 \text{ U } Cs_2)) \quad (1)$$

Here the two colours 1 and 2 must be singleton static subclasses in order to apply a synchronized product between the SRG and the automaton.

3. DESCRIPTION OF THE TWO METHODS

The common idea of both methods is to treat dynamically (on-the-fly) asymmetries.

Regarding the reachability problem, the asymmetry is due to the existence of some transitions

whose treatment depends on colours. The ESRG method exploits the fact that these dependencies are only local so can be abstracted for other transitions.

Concerning model-checking, asymmetries can arise due the need to observe particular colours during the system runs. However, this particular observation is not relevant for all parts of the verification process. The SSP method takes profit from such observations to construct the more adapted group of symmetries.

The ESRG method: The first step of the method consists to partition the transitions in two subsets: the *asymmetric* transitions which involve a static subclass occurring either in some coloured function around the transition or in the guard and the other ones called the *symmetric* transitions.

Each node of the ESRG contains a symbolic marking which does not take into account the static subclasses (i.e. to each class corresponds a single subclass). Such symbolic marking is called *symmetrical representation (SR)*. When an asymmetrical transition is enabled in a node, it is completed with all the possible symbolic markings which refine the SR by taking into account the static subclasses. Such markings are called the *eventualities* of the SR. The firing of an asymmetrical transition is always processed from the eventualities of a node. The firing of a symmetrical transition is processed from the eventualities of a node if some of them (but not all) are present. A key point of the method is the removal of the eventualities of a node. It occurs when all the possible eventualities are present (i.e. the whole states represented by the SR are reachable) and all the asymmetric transitions fireable from the eventualities have been fired. We call this kind of node a *saturated* node. In order to accelerate this removal, we give priority to the firing of asymmetric transitions from nodes with all the eventualities present and then to the firing of symmetric transitions from saturated nodes since the reached node is immediately saturated.

From an implementation point of view, since the eventualities are also symbolic markings, no specific development is required for the firing rule. However a symbolic firing is more time consuming since both the symbolic marking and optionally the eventuality are computed and canonized.

The expected reduction of the space complexity of the ESRG building w.r.t. the SRG building is mainly related to the proportion of asymmetric transitions in the WFN. When all transitions are symmetric, the ESRG is the SRG. We expect that the reduction is maximal when the proportion of asymmetric transitions is small. Furthermore when this ratio increases, the reduction may remain important whenever the saturation effect is predominant.

The SSP method: The *Symbolic Synchronized Product* method (SSP) aims at synchronizing a partially symmetric Büchi automaton with the runs of a symmetrical system. The reader can refer to (Haddad *et al.* 2000) for the generalization to asymmetrical system specifications. The first step of the method consists to analyze the atomic propositions labelling a node of the Büchi automaton. *Local static subclasses* are associated to each node so that two colours inside a static subclass are equivalent w.r.t. the propositions (i.e. permuting the two colours let invariant the set of atomic propositions). Given a state b, we denote its local static decomposition by $ls(b)$.

A state of the synchronized product is defined by: a local decomposition into static subclasses LS, a symbolic marking defined upon these subclasses SM and a state of the automaton b. The atomic propositions of b are satisfied by all the ordinary markings associated to the symbolic marking.

The key point is the building of the successors of a node. We choose a successor for b, say b'. Then we apply a symbolic firing rule on SM and we obtain a new symbolic marking, say SM'. The next step consists to obtain the subset of ordinary markings of SM' which fulfill the atomic propositions of b'. We do it in a symbolic way by refining the static decomposition LS into $LS \cap ls(b')$. This refinement leads to replace SM' by a set of symbolic markings where we straightforwardly decide for each one if all of its ordinary markings satisfy the atomic propositions of b' (or none). We then check on the subset of remaining symbolic markings whether we can group some of them. The involved techniques are intricate and will not be described here.

Compared to the static method of (Ilié and Ajami 1997) for the model checking using the SRG, the expected reduction of space complexity depends on the degree of symmetry of the states of the Büchi automaton induced by the atomic propositions.

4. EVALUATION

We have implemented our symbolic methods by reusing the core implementation of the GreatSPN software. GreatSPN is a well-known software which computes the SRG of well formed nets, including the management of the symbolic marking representation (Chiola and Gaeta 1995).

Since both SSP and ESRG methods are based on the notion of *dynamic symmetries* we first implemented the module DySy, a *dynamic manager of*

symmetries. Then we reused the GreatSPN core without significant changes in its internal structures, and we developed a modular implementation of the SSP and the ESRG methods. The computation of the SSP requires an external module which processes the synchronized product with the Büchi automata of an LTL formula, in order to perform the model checking. The LTL model checker called SPOT (see http://spot.lip6.fr), has been used (fig. 3).

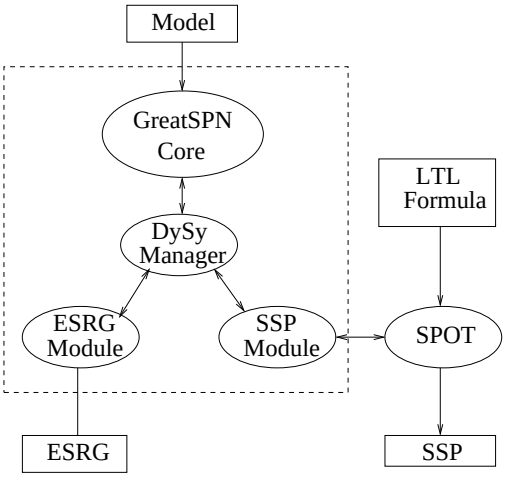

Figure 3. Architecture of the system

In the next subsections, the time spent as well as the memory consumed are measured for our two symbolic approaches, in comparison with previous ones. The memory consumption is measured in number of nodes.
It is worth noting that we used a 2 Ghz Intel Pentium IV machine, with 775 Mbytes memory size and working on Linux 9.1 Operating System.

4.1 ESRG Evaluation

Table 1 shows how ESRG and SRG behave on the asymmetrical model of Figure 1. On this example, the SRG is the RG because of the partition of the colors of the color class C in elementary static subclasses. In Table 1, columns denoted *CPU Time* shows the *CPU construction time* of each structure. The Column *SM* represent the number of symbolic markings in the SRG. The number of saturated and unsaturated ESMs are shown in the columns *(#ESM, #SAT)* and *(#ESM, #NOT SAT)*, respectively. The total number of stored eventualities are given away in the column *E*. The last column (denoted *Ratio*) highlights the obtained space gain.

Time evaluation :
Regarding the time spent for building each structure, we find similar values. This can be easily explained by the fact that the number of computed eventualities is equal to the number of *SM* in the SRG. Actually, the enabling of the asymmetrical

Proc-esses	SRG(RG)		ESRG				Ratio
	Cpu Time	# Symb. Mark. : SM	Cpu Time	# ESM		# Event. : E	SM / (# ESM + E)
				# Sat	# Not Sat		
3	0	139	0	33	6	15	2
5	2	2709	1	96	20	325	6
7	41	50159	25	221	42	4655	10
9	1147	911017	939	394	72	56745	15
11	------	16378179	54074	661	110	638285	25

Table 1. Evaluation of the ESRG method

transitions in the ESRG approach is tested on the eventualities.

Space evaluation :
There is a *super-linear* (Figure 4) gain in memory occupation because numerous ESM are saturated, and then are saved without their eventualities. We also observe that the number of eventualities, E, progresses much more slowly than the SM one. This is due to the fact that the progression of the number of eventualities depends on the number of unsaturated ESM, which are in a very small proportion compared to the total number of ESM ($\#SAT + \#NOT\ SAT$). On a model with a less important proportion of asymmetric transitions, we would obtain an even greater reduction. We also notice that the last SM value (for 11 processes) in the Table 1 is estimated from the ESRG construction, because GreatSPN reaches its limit when computing the SRG of the asymmetric model with 10 processes.

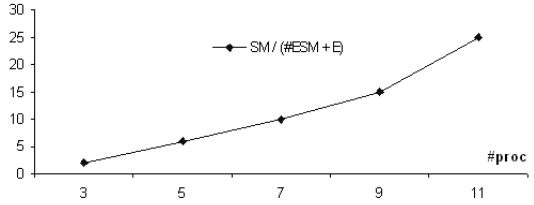

Figure 4. Ratio of SM w.r.t (#ESM + E)

4.2 Symbolic Synchronized Product Evaluation

In order to test our LTL verification method, we choose to check the formula 1 on the symmetric WFN of Figure 2 wherein the whole synchronized product structure is constructed, because of the truth of the formula 1 of section 2. Table 2 compares the SSP approach against both the standard synchronized product (SP-rg) and the symbolic one, obtained from the SP-srg presented in (Ilié and Ajami 1997). The columns designed by *CPU Time*, show the CPU construction time for each structure. The columns denoted *#States*, represent the number of states for each structure. The last two columns depicted *s-rg/s-ssp* and *s-srg/s-ssp* represent the obtained space gain.

Proc-esses	SP-rg		SP-srg		SSP		Ratio	
	Cpu Time	#States : s-rg	Cpu Time	#States : s-srg	Cpu Time	#States : s-ssp	s-rg / s-ssp	s-srg / s-ssp
3	1	129	1	129	1	42	3	3
5	2	2655	2	739	4	155	17	4
7	53	51711	4	2197	12	392	131	5
9	920	974847	8	4871	31	801	1217	6
11	-------	-------	14	9129	60	1430	-------	7

Table 2. Evaluation of the SSP method.

Time evaluation :
Regarding the CPU time, the SSP building goes slightly slower than the SP-srg. Actually, the SSP requires more operations than the SP-srg, even if it contains less nodes. In contrast, the building of the SSP goes faster than the SP-rg because it is two order magnitude smaller (column *s-rg/s-ssp*).

Space evaluation :
A linear order reduction (Figure 5) is witnessed when comparing the space used by the SSP and the SP-srg (column *s-srg/s-ssp*). Generally, the reduction obtained by larger sets of ordinary markings than those allowed in the SRG is more important that the increase due to the fact that these sets may have a non empty intersection.

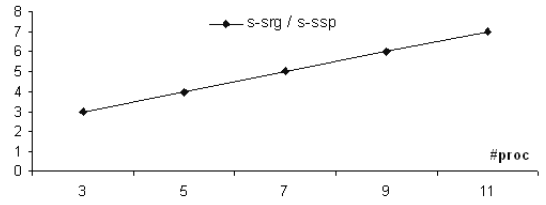

Figure 5. Ratio of s-srg w.r.t s-ssp

5. CONCLUSION AND PERSPECTIVES

In this work, we have presented the design and the evaluation of two verification methods exploiting partial symmetries and applicable on well-formed Petri nets. These two methods are extensions of the symbolic reachability graph construction for the well-formed Petri nets. The extended symbolic reachability graph construction tackles the reachability problem. The symbolic synchronised product checks a partially symmetric linear time formula on a net. On standard case studies, we have shown that these two methods outperform the previous approaches dealing with partial symmetries.

We plan to extend this work in two directions. On the one hand, we notice that the asymmetry is often introduced by the presence of guards on transitions like $X < Y$. Expressing such guards in the well-formed nets formalism implies to split the colour class of these variables into singleton static subclasses. So we want to handle in a dynamic way these guards (i.e. during the symbolic construction). On the other hand, as a method for the reachability problem, the symbolic synchronized product suffers from the following drawback: given two nodes of the product, it does not detect that the set of states associated to one of them is included in the set of states associated to other one. We are currently developping this additional test in the tool. However, taking advantage of this test for the model-checking problem requires a theoretical development on which we are working.

REFERENCES

Ajami, K., S. Haddad and J.-M. Ilié (1998). Exploiting Symmetry in Linear Temporal Model Checking: one Step Beyond. In: *Proc. of Tools and Algorithms for the Construction and Analysis of Systems TACAS'98, part of Theory and practice of Software ETAPS'98*. Vol. 1384 of *LNCS*. Springer Verlag. Lisbon - Portugal. pp. 52–67.

Chiola, G. and R. Gaeta (1995). Efficient Simulation of Parallel Architectures Exploiting Symmetric Well-formed Petri Net Models. In: *Sixth International Workshop on Petri nets and Performance Models*. IEEE Computer Society Press. Durham, NC, USA.

Chiola, G., C. Dutheillet, G. Franceschinis and S. Haddad (1993). Stochastic well-formed coloured nets for symmetric modelling applications. *IEEE Transactions on Computers* **42**(11), 1343–1360.

Clarke, E.M., R. Enders, T. Filkorn and S. Jha (1996). Exploiting Symmetry in Temporal Logic Model Chacking. *Formal Methods and System Design* **9**, 77–104.

Emerson, E. A. and R. J. Trefler (1999). From Asymmetry to Full Symmetry: New Techniques For Symmetry Reduction in Model Checking. In: *Proc of CHARMEÕ99*. Lecture Notes in Computer Science. Springer Verlag. Bad Herrenalb - Germany. pp. 142–156.

Emerson, E.A. and A. Prasad Sistla (1996). Symmetry and Model Checking. *Formal Methods and System Design* **9**, 307–309.

Haddad, S., J.M. Ilié and K. Ajami (2000). A model checking method for partially symmetric systems. In: *Proceedings of FORTE/PSTV'00*. Kluwer Academic Publishers. Pisa, Italy. pp. 121–136.

Haddad, S., J.M. Ilié, M. Taghelit and B. Zouari (1995). Symbolic Reachability Graph and Partial Symmetries. In: *Proc. of the* 16^{th} *Intern. Conference on Application and Theory of Petri Nets*. Vol. 935 of *LNCS*. Springer Verlag. Turin, Italy. pp. 238–257.

Ilié, J.-M. and K. Ajami (1997). Model Checking through Symbolic Reachability Graph. In: *Proc. of Theory and Practice of Software Development TAPSOFT'97 - part of 7th CAAP*. Vol. 1214 of *LNCS*. Springer Verlag. Lille - France. pp. 213–224.