A MODEL CHECKING METHOD FOR PARTIALLY SYMMETRIC SYSTEMS

Serge Haddad[1], Jean-Michel Ilié[2] and Khalil Ajami[2]

(1) lab. Lamsade, Université Paris Dauphine, France, Serge.Haddad@lamsade.dauphine.fr

(2) lab. LIP6, Université Paris VI, France, Jean-Michel.Ilie@lip6.fr, Khalil.Ajami@lip6.fr

Abstract A new method of model checking is proposed based on the existence of symmetries in system. We show how to fully handle the partial symmetries of both properties and systems. Our method does not depend on a particular formalism and *a priori* can be applied to any one. Well-formed Petri Nets are used as an illustration.

Keywords: Verification and validation, Temporal logic, model-checking, symmetries, partial symmetries, Büchi automata, well-formed Petri nets.

1. INTRODUCTION

Model checking of temporal logic formulas over finite state systems is now a widely used method of verification of protocols and distributed algorithms. Such a technique has led to numerous tools (Holzmann, 1997,...). However the main drawback of this kind of verification is the complexity factor depending on the size of the space of reachable states. Thus different improvements have been proposed (and implemented in tools). The partial order analysis exploits the independence of events of the system in order to avoid explorations of equivalent paths in the state graph (Godefroid and Wolper, 1994). Another fruitful research direction is based on the symmetries of the system to be analysed. The main idea is to build a quotient graph where nodes denote set of equivalent reachable states. Then the reduced graph is shown to be equivalent to the original one w.r.t. to some generic properties (Jensen, 1986,Chiola et al., 1991).

However the modeller often needs to check particular properties related to the behaviour of its system. It is then necessary to adapt the quotient graph building with the aim of verifying a temporal logic formula. The key point is the characterisation of symmetries of a formula and we now discuss the previous approaches to this problem. We limit our presentation to the linear time logic (e.g. LTL). So let us briefly recall its usual verification algorithm (Gerth et al., 1993):

- the negation of the formula is translated into a Büchi automaton,
- the synchronised product of the automaton and the state graph of the system is built (product of compatible states and transitions),
- a path is searched within the synchronised product, such that it ends by a loop containing a state associated with one of the acceptation states of the Büchi automaton. We call such a path, an invalidating path.

The formula is true if and only if such an invalidating path does not exist. Although the size of the automaton is exponentially larger than the size of formula, it remains low compared to the number of states in the system. This last parameter is the main complexity factor.

In a first approach (Clarke et al., 1996), one restricts the atomic propositions of the language to symmetric propositions. For instance, "*In a future state, all processes will be idle*" or "*If some process is waiting for a resource, then some process will get it*" are symmetric formulas. In other words, a formula is symmetric if its atomic propositions are invariant under any process identities permutation. For checking such formulas, one substitutes a quotient state graph to the state graph and one applies the above algorithm without any change. Unfortunately, many usual formulas are not considered as symmetric. For instance, the fairness formula "*If some process is waiting for a resource, then it will get it*" is not considered as a symmetric formula.

The second approach (Emerson and Prasad Sistla, 1996), defines symmetric Büchi automata. An automaton is symmetric if, given a (accepting) state and a process identities permutation, there is another (accepting) state whose atomic propositions are obtained by the permutation applied on the atomic propositions of the first state and for any successor of one state, there is a successor of the other one for which the property is again fulfilled (and this recursively). Starting from this automaton and the model of the system, one directly builds a quotient synchronised product. It is then shown that the existence of an invalidating path in the original synchronised product and the quotient structure are equivalent. The symmetric formulas of the previous method, and also new temporal properties like the discussed fairness property are considered as symmetric formulas.

Anyway, this technique does not cover the case of partially symmetric formulas which are considered as asymmetrical. Here is a partially symmetric 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*". Therefore in (Ajami et al., 1998), the authors have proposed a more refined definition of the quotient synchronised product than the previous one:

- One computes an equivalence relation between states of the Büchi automaton per item of the "*symmetry group*" of the model in such a way that two states are equivalent whenever they induce the same present and future w.r.t. the action of this item.
- A quotient synchronised graph is built by applying these equivalence relations on the pairs (state of the model, state of the automaton).

In case of a symmetric automaton, the quotient graph is identical to the previous one, but the technique also reduces the complexity of the verification in case of a partially symmetric automaton. In other words, it generalises the previous techniques.

However none of the existing methods cover two important situations. At first, they require the system to be symmetric. Many models do not fulfill this requirement. In most cases, they are partially symmetric, meaning that large parts of the specification is symmetric but some critical modules are asymmetric. In (Haddad et al., 1995), a specific algorithm is proposed for partially symmetric high-level Petri nets, but it mainly deals with the reachability problem and it does not seem that it can be extended to the model checking. In (Emerson and Trefler, 1999), asymmetric behaviour is allowed in "*symmetric states*" (left invariant by any process identities permutation). Unfortunately, this kind of asymmetry is very restrictive.

The second situation which cannot be exploited by the previous methods concerns the formula. Actually, the two automaton-based techniques assume that some symmetric relations hold on the structure of the automaton. However in automata of partially symmetric formula, most of the states of the automaton are partially symmetric, but the automaton is globally asymmetric (i.e. the equivalence relations between states are almost reduced to the identity).

In the present work, we will show how to overcome these two problems. In the second section, a new method is presented which applies on a symmetric system and any Büchi automaton. Each state of this automaton is associated with the group of symmetries which left its set of atomic propositions globally invariant. No supplementary condition is required on the structure of the automaton. The method does not depend on a specific formalism for the system and can be applied over any model where symmetries can be syntactically checked. So in the third section, we illustrate a possible implementation on well-formed Petri nets for which symmetries have been intensively studied Chiola et al., 1993. More importantly, we show how to apply our method on asymmetric systems. The key point is that such systems are defined as synchronised products of a symmetric model and an asymmetric Büchi automaton. In our opinion, many protocols and algorithms can be represented with the help of this formalism. In the conclusion, we discuss about combinations of techniques and experimentations.

2. PRINCIPLES OF OUR METHOD

2.1. VERIFICATION OF TRANSITION SYSTEMS

In order to represent the general aspects of our method, we consider it at a semantic level, in other terms we consider the transition system that is generated from the syntactic model, e.g. a Petri net.

Definition 1 *A (finite) transition system* $S = (Q, Q_0, R, Prop, \Pi)$ *is defined by :* Q*, a (finite) set of states;* Q_0*, the set of initial states;* R*, a transition between states* $R(q, q')$*, also denoted* $q \to q'$*;* $Prop$*, the set of atomic propositions;* Π *an* ***injective*** *mapping from* Q *to* 2^{Prop}*.*

The requirements that Π is injective ensures that each state is totally characterised by the values of its atomic propositions. Since we focus on the verification of state formulas, there is no label attached to arcs. Our approach could be adapted easily to formulas expressing conditions on state transitions. The final result (see proposition 8) holds for any finite branching transition systems, i.e. the number of initial states is finite and each reachable state has a finite number of successors.

One may express a linear time temporal logic formula with languages like LTL (Pnueli, 1977) or μ-calcul (Kozen, 1983). During the model checking process, the considered formula can be translated in a Büchi automaton (Vardi, 1996).

Definition 2 *A Büchi automaton* $A = (B, B_0, R, Prop, \Pi, F)$ *is defined by :* B*, a finite set of states;* B_0*, the subset of* B *of the initial states;* R*, a transition relation between states* $R(b, b')$*, also denoted* $b \to b'$*;* $Prop$*, the set of atomic propositions;* Π *a mapping from* B *to* 2^{Prop}*;* F*, the subset of* B *of the accepting states.*

The verification problem of a formula expressed by the Büchi automaton is usually reduced to the search of an infinite run within the system : such a run $\{q_i\}_{i=0..\infty}$ with $q_0 \in Q_0$ must correspond to an infinite path $\{b_i\}_{i=0..\infty}$ with $b_0 \in B_0$ within the automaton. More precisely, the atomic propositions labelling every state b_i must also be present in state q_i : $\Pi(b_i) \subset \Pi(q_i)$. In addition, an accepting state must occur infinitely often in the considered path. This leads to the definition of the following key concept, called the synchronised product.

Definition 3 *Let S be a transition system and A a Büchi automaton. The synchronised product between S and A is a graph $Gr(S, A) = (V, V_0, R)$, defined by : $V = \{(q, b)$ s.t. $\Pi(b) \subset \Pi(q)\}$, the set of nodes; $V_0 = \{(q, b) \in V$ s.t. $q \in Q_0 \wedge b \in B_0\}$, the set of initial states; R, the transition relation between nodes, denoted $\rightarrow$, and such that $(q, b) \rightarrow (q', b')$ iff $q \rightarrow q'$ and $b \rightarrow b'$.*

One standard model checking principle consists of a translation of the negation of the formula in a Büchi automaton, the building of the synchronised product, and the search of an "invalidating" path within the synchronised product.

Definition 4 *Let S be a transition system and A a Büchi automaton expressing the negation of formula f. Formula f is valid iff there is no invalid path in $Gr(S, A)$. An invalidating path of formula f is an infinite path within the synchronised product $Gr(S, A)$, starting from one initial node and including infinitely often the subset of nodes $\{(q, b)$ s.t. $b \in F\}$*

If the system is finite, the verification procedure is reduced to the search of an elementary path ending by a state yet visited, s.t. between the two occurrences, a state referring an accepting condition is met. Many improvements to this method can be found in the literature, among them there are the on-the-fly techniques which search for an invalidating path simultaneously to the building of $Gr(S, A)$ (Gerth et al., 1993). All these techniques can be applied in the context of the quotient synchronised product we develop in section 2.3.

2.2. SYMMETRIC TRANSITION SYSTEMS

In order to highlight all the developed concepts, we consider in this section a model of a protocol where four processes, denoted by $C = \{u, v, w, x\}$, execute the same program. For sake of simplicity, we assume that communications are instantaneous, so that the global system is entirely defined by the local states of the processes. In the following, the local state of any process, e.g. x, is symbolically represented by a variable, e.g. $l(x)$. The definition domain for $l(x)$ is {idle, transmit, wait, access}.

Because "symmetries" are usually handled by group theory, we now recall some elementary notions of group (Lang, 1977).

Definition 5 *Let G be a group, containing a neutral element id, which operation is denoted by $(\bullet)$.*

- *Let E be a set, an action of G over E is a mapping from $G \times E$ to E s.t. the image of (g, e), denoted by $g.e$, fulfills : $\forall e \in E\; id.e = e \quad \forall g, g' \in G\; (g \bullet g').e = g.(g'.e)$*
- *The isotropy (sub)group G_e of an element e is defined by : $G_e = \{g \text{ s.t. } g.e = e\}$*
- *Let H be a subgroup of G, the orbit $H.e$ of e under H is defined by : $H.e = \{g.e \text{ s.t. } g \in H\}$*
- *This action can be straightforwardly extended to the powerset of E by : $g.E' = \{g.e \text{ s.t. } e \in E'\}$*

Let us assume that E is the set of the atomic propositions of our model. For instance, proposition $[l(u) = idle]$ means that the local state of process u is idle. If G is the group of permutations of C and if g is the permutation which exchanges processes u and v, then $g.[l(u) = idle] = [l(v) = idle]$. Let $\{[l(u) = idle], [l(v) = idle]\}$ be a set of atomic propositions, then the isotropy group of this subset is the subgroup of permutations which let the subset $\{u, v\}$ globally invariant.

We are now able to characterise what is a symmetric transition system.

Definition 6 *Let S be a transition system and G be a group acting on Prop.*
S is said to be symmetric (w.r.t. G) iff :

- *Every state has a " symmetric " state w.r.t. any element of G : $\forall q \in Q, \forall g \in G, \exists q' \in Q,\; \Pi(q') = g.\Pi(q)$ The action of the group on the states is extended, by denoting $g.q$, the unique q' of the former formula.*
- *The set of initial states is invariant under the action of G : $G.Q_0 = Q_0$.*
- *The action of G is congruent w.r.t. the transition relation : $\forall q, q' \in Q,\; \forall g \in G,\;\; q \to q' \Leftrightarrow g.q \to g.q'$*

Observe that all these conditions hold regarding our model of protocol, by defining G as the group of process permutations.

2.3. MODEL CHECKING OF SYMMETRIC TRANSITION SYSTEM

As a state is associated with a subset of $Prop$ in both the Büchi automaton and the model, we will use the action of G on the powerset of $Prop$. Let b be a state of Büchi automaton A, $G_{\Pi(b)}$ will simply be denoted G_b (although one cannot define an action of G on B). Observe that G_b is strongly related to the symmetry degree of b independently of the structure of the Büchi automaton.

Let us consider different cases of G_b in the context of our example. If this subgroup equals the group G, then the state is totally symmetric, e.g. this is the case for proposition set $\{[l(u) = idle], [l(v) = idle], [l(w) = idle], [l(x) = idle]\}$. In contrast, whenever the isotropy group is reduced to the identity (denoted $\{id\}$), the state is totally asymmetric, e.g. $\{[l(u) = idle], [l(v) = transmit], [l(w) = wait], [l(x) = access]\}$. In most cases, the isotropy group is not trivial (i.e. it differs from id and G). Let us note that in the context of our model, the isotropy group of processes implicitly corresponds to a partition of processes, namely the group of permutations which left each set of the partition invariant. For $\{[l(u) = idle], [l(v) = idle]\}$, the partition of C is $\{\{u, v\}, \{w, x\}\}$. According to a state of the Büchi automaton, we call such a partition *the local partition of the state.*

Our aim is to build a quotient of the synchronised product over which it is possible to search an invalidating path directly. In this " quotient" structure, each node is characterised by a triple (H, O, b), where H is a subgroup of G, $O \subset Q$, $b \in B$. Moreover, the following two points must hold:

(C1) $\forall q \in O, \Pi(b) \subset \Pi(q)$

(C2) $H.O = O$

The intuitive idea is that the node (H, O, b) is an aggregation of a set of nodes of the synchronised product, more precisely $\{(q, b)\}_{q \in O}$. As we will see now, H is used in the definition in order to give a sound definition of the successor relation.

The building of the quotient structure starts from the set of initial nodes, defined as follows : $(G_{b0}, G_{b0}.q_0, b_0)$ with $q_0 \in Q_0$, $b_0 \in B_0$ and $\Pi(b_0) \subset \Pi(q_0)$. Observe that condition $C2$ is fulfilled trivially because G_{b0} is a group, moreover condition $C1$ is deduced from the definition of G_{b0}.

Then, there remains to define the successor relation.
(H_2, O_2, b_2) is a successor of (H_1, O_1, b_1), denoted $(H_1, O_1, b_1) \rightarrow (H_2, O_2, b_2)$,

iff $b_1 \to b_2$ **and** $\exists q_1 \in O_1$, $\exists q_2 \in Q$ s.t. $q_1 \to q_2$ and $\Pi(b_2) \subset \Pi(q_2)$. **Then** $O_2 = (H_1 \cap G_{b_2}).q_2$ and $H_2 \subset G_{O_2}$.

Observe that both conditions $C1$ and $C2$ are fulfilled by the new node. Effectively, from $O_2 \subset G_{b_2}.q_2$, one deduces that: $\forall q'_2 \in O_2, \exists g \in G_{b_2}$ *s.t.* $q'_2 = g.q_2$. So, $\Pi(b_2) = g.\Pi(b_2) \subset g.\Pi(q_2) = \Pi(q'_2)$. Moreover, as $H_2 \subset G_{O_2}$ then $H_2.O_2 = O_2$.

In order to make the successor relation operational, one must specify how the subgroup H_2 is chosen. With regard to the successor relation, it is interesting to maximise H_2, i.e. to choose $H_2 = G_{O_2}$, but this choice might not be possible w.r.t. the syntactical aspects of the model. In any case, H_2 can be chosen as the subgroup $(H_1 \cap G_{b_2})$. The following lemma and proposition 1 show the validity of our construction.

Lemma 1 *Let S be a symmetric transition system and A be a Büchi automaton. Let $Gr(S, A)$ be the corresponding synchronised product and $GRQ(S, A)$ the corresponding quotient structure.*

- *Let (H_1, O_1, b_1) and (H_2, O_2, b_2) be two nodes of GRQ(S,A).* $(H_1, O_1, b_1) \to (H_2, O_2, b_2) \Rightarrow \forall q_2 \in O_2,\ \exists q_1 \in O_1,\ (q_1, b_1) \to (q_2, b_2)$
- *Let $(q_0, b_0) \to (q_1, b_1) \to ... \to (q_n, b_n) \to ...$ be a (infinite) path in $Gr(S, A)$, then $\exists (H_0, O_0, b_0) \to (H_1, O_1, b_1) \to ... \to (H_n, O_n, b_n) \to ...$ a (infinite) path in $GRQ(S, A)$ s.t. $q_i \in O_i$ $(i \in 1..n)$.*

It is worth noting that the first point becomes false in the case where one exchanges its consequence by the following one: " $\forall q_1 \in O_1, \exists q_2 \in O_2,\ (q_1, b_1) \to (q_2, b_2)$".

Proposition 1 *For a symmetric finite branching transition system, there is an invalidating path in the synchronised product if and only if there is an invalidating path in the quotient structure.*

3. APPLICATION TO THE WELL-FORMED PETRI NETS

The well-formed Petri net model offers a global solution for the design, verification and performance evaluation of distributed systems graphs. A well-formed Petri net is a high level Petri net model. Its specific syntax has made possible to define methods which take profit from the symmetry relations existing in systems. One of the associated methods is the building of the symbolic reachability graph which is a highly compact structure used to represent the state space. It can be used to

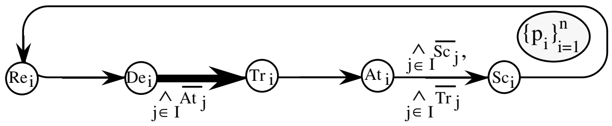

Figure 1 Automaton of one process

analyse some standard properties of the Petri nets as well as to derive an aggregated Markov chain in order to compute performance indices (Chiola et al., 1993). We are interested in adapting this construction to our method, but we do not fully describe neither the well-formed Petri net definition nor the implementation of the symbolic reachability graph building (see Chiola et al., 1991 for more details). Actually, we will focus on the representation of nodes and on the building of the successor relation.

3.1. PRESENTATION OF THE WELL-FORMED PETRI NET MODEL

Description of the system and properties to verify. The system which is described here consists of n processes $\{P_i\}_{i=1..n}$ accessing to a critical section. Like in the former model, we consider that the processes run the same critical section access protocol. They are now synchronised by shared variables.

This protocol is described for a process P_i in Figure 1. Initially, the process is idle (local state Re) then it may ask for an access to the critical section (local state De). Before accessing this section (local state Sc) the process follows a sequel of two local states : Transmitting (Tr) and Waiting (At). We now detail the state transitions : (1) A requesting process can be in state Tr if there is no process in waiting state. The scheduler considers as privileged all the processes in state De. This is represented in the figure by a bold arc. (2) A process changes from state Tr to At without specific condition. (3) To enter the critical section, there must be no process in this section, but also no process in state Tr.

We aim at demonstrating that the system is (weakly) fair : Every process which asks for the critical section will obtain it in a finite time. Intuitively, this property holds due to the conditions which ensures that requests are treated *wave after wave*. A wave starts from the moment when there is a requesting process and is completed as soon as one process enters the waiting state.

Hence, a wave contains all the processes in state Tr and At, and these processes will enter the critical section without specific order. Due to

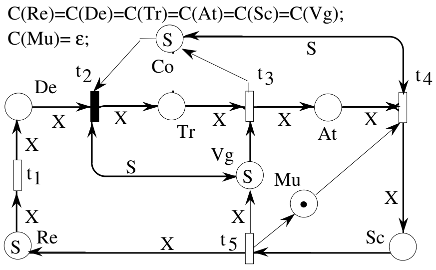

Figure 2 Modelling of the symmetrical behaviour of the system

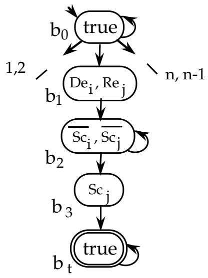

Figure 3 Büchi automaton of $\neg f$

the privilege of the change from De to Tr, a requesting process belongs either to the current wave or to the following one. This ensures the fairness property.

In a second stage, the system is modified so that within a wave, the executions of the critical section are ordered according to a priority relation, induced by the identity number of processes. Actually, The first system is symmetric, while the second is not.

With respect to this new specification, the two expected properties are the followings :

- Every requesting process will finally obtain the access
- The processes in state Tr or At will be served according to the order induced by the specified priority relation.

In contrast, the strong fairness property "*A requesting process will be served before every process in state idle*" is false.

Formal modelling for the System and its properties.

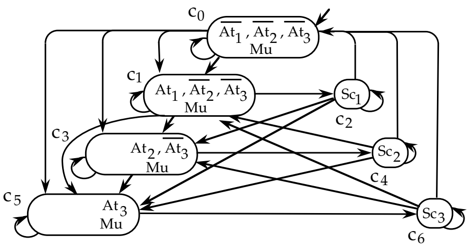

Figure 4 control automaton w.r.t. a priority relation

In order to represent our system, we proceed in two stages : a symmetric system is specified first in terms of the well-formed Petri nets of Figure 2, then the runs which do not preserve the priority relation are prohibited by means of the additional automaton of Figure 4, namely control automaton. Actually, the system formally corresponds to the synchronised product of the well-formed Petri net and the control automaton.

In a well-formed Petri net, a colour domain is attached to each place and transition. Each token in a place is typed by a colour of the place colour domain. Each colour of a transition indicates a different way to fire the transition, hence, the labels of the arcs are no more integers but colour functions which denote for each firing the quantities of tokens to be consumed and produced per each colour. In well-formed Petri nets, the colour functions are written under a specific syntax which allows one to exploit the system symmetries. Function S represents a constant function, called a diffusion. Function S is a sum of tokens s.t. there is one token per colour of the domain, e.g. one for each process. Function X represents the colour which is used to instantiate the firing of a transition in the net.

In addition, a priority number is associated with each transition of the well-formed Petri nets. Thus, among the transitions that are enabled from the current marking of the net, only the ones which have the highest priorities can be fired.

In the well-formed Petri net of Figure 2, the local states of the processes are represented by means of the corresponding places. The set $C = \{\langle 1\rangle), \langle 2\rangle), \langle 3\rangle\}$ is the colour domain of processes. It is the domain of all the places (except Mu) and the domain of all transitions of the net. The marking of place Re by the (constant) diffusion function corresponds to the fact that, initially, all processes are idle. Three more places allow one to implement the protocol mechanism : (1) Place Mu is used to specify the control of accesses to the critical section; initially, it is set to an (uncoloured) token and prohibits the firing of transition t_4, whenever place Sc is marked. (2-3) Places Vg and Co (initially set to S) are used to specify the wave mechanism. Vg prohibits the firing of transition t_2, whenever one of the At or Sc places is marked. Place Co prohibits the firing of transition t_4 whenever place Tr is not empty. At last, t_2 is a transition the firing of which is privileged with respect to the others transition, so that the expected behaviour of the scheduler is obtained by the following priority relation : $\forall t \neq t_2$, $Pri(t_2) > Pri(t)$.

The specification of the well-formed Petri net is completed by the control automaton A_c given in Figure 4. The first state of A_c specifies that there is no process currently in the critical section, and that there is no request for that. The three states at the left side specify the different cases where a process must be privileged, provided the critical section is empty. Moreover, the three processes in the right side determine which processes is in the critical section. The arcs from left to right ensure the priority mechanism. The others follow the behaviour of the net.

The semantic of the system is defined by the synchronised product between the control automaton and the reachability graph of the net.

In order to illustrate our method, we wish evaluate the following strong fairness property : "*A requesting process will be served before every idle process*". The corresponding LTL formula is the following (Pnueli, 1977) : $f = \wedge_{i,j \in I} G[(De_i \wedge Re_j) \Rightarrow X(\overline{Sc_j} U Sc_i)]$

The Büchi automaton $A_{\neg f}$ partially represented by Figure 3 models the possible runs of the system for which the negation of property f holds. For sake of simplicity, only one branch (i, j) is represented. In this elementary case, the structure of the automaton is immediately deduced from the formula to invalidate. With the help of the first state of the automaton, one waits to the moment when $(De_i \wedge Re_j)$ holds, which will cause the invalidation of the property. Then, the next states of the automaton ensures that process j will enter the critical section before i. The last state represents the remaining part of the infinite run, without meaning according to the considered property.

We highlight now (for this kind of specification) that *the checking of a formula on an asymmetric system can be reduced to the checking of an asymmetric formula on a symmetric system.* The search of a path must be computed through the synchronised product of the three structures : the reachability graph of the well-formed Petri net, the control automaton and the automaton of the negation of the formula. However, the synchronised product is a commutative and associative operation. It is thus possible to perform the synchronised product of the two automata, A_c and $A_{\neg f}$, in order to obtain a new one, then to apply our construction on this new automaton and the well-formed Petri net (a symmetric model).

On can check that the following path in the synchronised product invalidates the desired formula :

$$(m_0, (c_0, b_0)) \stackrel{t_1(\langle 1 \rangle)}{\longrightarrow} (m_1, (c_0, b_1)) \stackrel{t_2(\langle 1 \rangle)}{\longrightarrow} (m_2, (c_0, b_2)) \stackrel{t_1(\langle 2 \rangle)}{\longrightarrow} (m_3, (c_0, b_2)) \stackrel{t_2(\langle 2 \rangle)}{\longrightarrow}$$

$$(m_4, (c_0, b_2)) \stackrel{t_3(\langle 2 \rangle)}{\longrightarrow} (m_5, (c_3, b_2)) \stackrel{t_3(\langle 1 \rangle)}{\longrightarrow} (m_6, (c_3, b_2)) \stackrel{t_4(\langle 2 \rangle)}{\longrightarrow} (m_7, (c_4, b_3))$$

Each node (m,(c,b)) of such a sequence corresponds respectively to one marking of the net and one pair obtained by a state of the Büchi automaton and a state of the control automaton. Observe that, the label of an arc represents some firing in the net. For instance, $\xrightarrow{t_1(\langle 1\rangle)}$ corresponds to the firing of transition t_1 with respect to process $\langle 1\rangle$.

3.2. BUILDING OF THE SYMBOLIC SYNCHRONISED PRODUCT

The symbolic synchronised product is a representation of the *quotient synchronised product*. We call them symbolic due to the analogy with the symbolic reachability graph representation (Chiola et al., 1991). Its major peculiarity is that objects are never more represented explicitly, but are viewed under a symbolic form. For instance, one can say symbolically that two processes require the same resource, instead of saying that processes 1 and 2 require resource 1. From such a symbolic representation, a symbolic firing rule can be defined which allows one to compute the symbolic reachability graph, automatically and directly from the object specification within the well-formed net.

Computation of local partitions of colours. Before computing the quotient synchronised product, one must compute and represent the isotropy group of each state of the automaton. The most easy way to proceed consists in *detecting* the colours which marks the same propositions in the states. These colours are gathered in subsets and these subsets form a partition, named local partition. The isotropy group is implicitly defined as the group of permutations which left each set of the partition globally invariant. Let us give three examples : (1) For a state where the atomic propositions are $\{\overline{At_1}, \overline{At_2}, \overline{At_3}\}$, the partition is composed by a unique singleton $\{\langle 1\rangle, \langle 2\rangle, \langle 3\rangle\}$ (i.e. all the permutations are admissible). (2) For a state where the atomic propositions are $\{\overline{At_2}\}$, the partition is composed of a singleton $\{\langle 2\rangle\}$ and a set composed of the remaining colours, here $\{\langle 1\rangle, \langle 3\rangle\}$ (the permutations must left $\langle 2\rangle$ invariant). (3) For a state where the atomic propositions are $\{At_2, \overline{At_3}\}$, the partition is composed of three elements, $\{\langle 1\rangle\}, \{\langle 2\rangle\}, \{\langle 3\rangle\}$. The isotropy group is reduced to the identity permutation.

States of the symbolic synchronised product. Let us recall that we want to represent triples (H,O,(b,c)), such that O is subset of markings which is unchanged under the action of the subgroup H ($O = H.o$, $\forall o \in O$). The subgroup H will be implicitly represented by a local partition, which may differ from the partition attached to the state of the automaton (H may be different from G_b). The subset of markings will be represented in a symbolic way : the colour domain is partitioned in abstract subsets, called *dynamic subclasses* (i.e. the composition is not known). However for each subset, the cardinality and the reference to a local partition is preserved. The markings of the net are now symbolic since their tokens are no more colour but dynamic subclasses. *Symbolic markings* are denoted like $\widehat{m}$.

In our example, there is one initial symbolic node, such that : $(\widehat{m}_0, (b_0, c_0))$ where $\widehat{m}_0$ is a symbolic marking of the net such that :

- $\widehat{m}_0$ is defined from a unique local partition $\{\langle 1\rangle, \langle 2\rangle, \langle 3\rangle\}$ composed of a unique dynamic subclass (denoted Z_1 s.t. $\mid Z_1 \mid= 3$),
- $\widehat{m}_0 = (Z_1.Re + Z_1.Co + Z_1.Vg) + Mu$, meaning that there is one token of each process in places Re, Co and Vg and that there is one uncoloured token in place Mu.

The atomic propositions of b_0 and c_0 $(\{\overline{At}_i\}_{i=1..3}, Mu)$ are satisfied by $\widehat{m}_0$. Observe that $\widehat{m}_0$ is a symbolic representation for both subgroup H of permutations and subset O of markings.

Symbolic firing in the quotient synchronised product. In order to build a successor of a node $(H_0, O_0, (b_0, c_0))$ within the symbolic synchronised product, a successor of the current state in the automaton is selected. For instance from (b_0, c_0), one can choose state (b_1, c_0) as a possible successor. Then a new local partition is computed, by intersecting the local partition of the state currently considered within the symbolic synchronised product and the local partition of the state newly considered within the automaton. This is equivalent to consider the group $H_0 \cap G_{(b_1,c_0)}$.

For state $(\widehat{m}_0, (b_0, c_0))$, the local partition in the symbolic synchronised product is given by $\{\langle 1\rangle, \langle 2\rangle, \langle 3\rangle\}$ and the one of the state within the automaton is given by $\{\langle 1\rangle\}, \{\langle 2\rangle\}, \{\langle 3\rangle\}$. Hence, a decomposition identical to colours is obtained. Let $\widehat{m'}_0$ be the symbolic representation $\widehat{m}_0$ after the above decomposition:

- $\forall i \in \{1, 2, 3\}$ Z_i is the unique dynamic subclass of $\{\langle i\rangle\}$
- $\widehat{m'}_0 = (Z_1.Re + Z_2.Re + Z_3.Re) + (Z_1.Co + Z_2.Co + Z_3.Co) + (Z_1.Vg + Z_2.Vg + Z_3.Vg) + Mu$

Then, a symbolic firing is performed in the net which is similar to the ordinary one (see Chiola et al., 1991 for more details). For instance, the firing of $t_1(Z_1)$ leads to symbolic marking $\widehat{m'}_1 = (Z_1.De + Z_2.Re + Z_3.Re) + (Z_1.Co + Z_2.Co + Z_3.Co) + (Z_1.Vg + Z_2.Vg + Z_3.Vg) + Mu$. Additionally, one must verify that $\widehat{m'}_1$ satisfies the atomic properties of (b_1, c_1) : $(\{\overline{At}_i\}_{i=1..3}, Mu, De_1, Re_2)$.

The last stage consists in grouping some local partitions which contain a single dynamic subclass, provided the distributions of the concerned dynamic subclasses over the places are the same. Actually, this does not modifies the set of ordinary markings associated with the symbolic marking and this means that the group of admissible permutations $H_0 \cap G_{(b_1,c_0)}$ is extended. For instance in $\widehat{m'}_1$, sets $\{\langle 2\rangle\}$ and $\{\langle 3\rangle\}$ can be grouped yielding symbolic marking $\widehat{m}_1$: Z_1 is the unique dynamic subclass of $\{\langle 1\rangle\}$ and Z_2 is the unique dynamic subclass of $\{\langle 2\rangle, \langle 3\rangle\}$; moreover, $\widehat{m}_1 = (Z_1.De + Z_2.Re) + (Z_1.Co + Z_2.Co) + (Z_1.Vg + Z_2.Vg) + Mu$.

4. CONCLUSION

Complexity reduction of the verification by use of the symmetries is now well established. However most of the proposed techniques do not handle (or in a limited way) the partially symmetric systems. In this work we have proposed an efficient way to deal with asymmetric models and/or formulas. Our method does not depend on a particular formalism and *a priori* can be applied to any one. Here we have illustrated the method on well-formed Petri nets in order to clarify the principles based on actions of groups on transition systems.

This work needs to be completed in two ways. At first, it must be integrated in a tool. So the implementation of the presented work is under progress. We have chosen to develop our method in the GreatSPN tool (Chiola et al., 1997) as symmetries are already exploited for well-formed Petri nets. To this implementation, must succeed a stage of evaluation in order to characterise systems for which our algorithm is efficient.

Moreover, we plan to combine our method with the global analysis of symmetries of the automaton as done in (Ajami et al., 1998). At last, we investigate the way to extend this method to performance evaluation of stochastic well-formed Petri nets.

References

Ajami, K., Haddad, S., and Ilié, J.-M. (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*, volume 1384 of *LNCS*, pages 52–67, Lisbon - Portugal. Springer Verlag.

Chiola, G., Dutheillet, C., Franceschinis, G., and Haddad, S. (1991). On Well-Formed Coloured Nets and Their Symbolic Reachability Graph. In Jensen, K. and Rozenberg, G., editors, *High-Level Petri Nets. Theory and Application*, pages 373–396. Springer Verlag.

Chiola, G., Dutheillet, C., Franceschinis, G., and Haddad, S. (1993). Stochastic Well-Formed Colored Nets and Symmetric Modeling Applications. *IEEE Transactions on Computers*, 42(11):1343–1360.

Chiola, G., Franceshinis, G., Gaeta, R., and Ribaudo, M. (1997). GreatSPN1.7: GRaphical Editor and Analyzer for Timed and Stochastic Petri Nets. *Performance Evaluation, North Holland Journal*, 24.

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

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

Emerson, E. A. and Trefler, R. J. (1999). From Asymmetry to Full Symmetry: New Techniques For Symmetry Reduction in Model Checking. In *Proc of CHARME'99*, Lecture Notes in Computer Science, pages 142–156, Bad Herrenalb - Germany. Springer Verlag.

Gerth, R., Peled, D., Vardi, M., and Wolper, P. (1993). Simple On-the-fly Automatic Verification of Linear Temporal Logic. In *Proc. Int Conf. on Protocol Specification Testing and Verification*.

Godefroid, P. and Wolper, P. (1994). A Partial Approach to Model Checking. *Information and Computation*, 110(2):305–326.

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

Holzmann, G. J. (1997). The Spin Model Checker. *IEEE Transaction on Software Engineering*, 23(5):279–295.

Jensen, K. (1986). Coloured Petri Nets. In Brauer, W., Reisig, W., and Rozenberg, G., editors, *Petri Nets: Central Model and their Properties, Advances in Petri Nets, Part 1*, volume 254 of *Lecture Notes in Computer Science*, pages 248–299, Bad Honnef, Germany. Springer Verlag.

Kozen, D. (1983). Results on the propositional mu-calculus. *Theoretical Computer Science*, 27:333–354.

Lang, S. (1977). *Algebra*. 7th printing. Addison Wesley.

Pnueli, A. (1977). The temporal logic of programs. In *Proceedings of the 18th IEEE Symposium on Foundations of Computer Science*, pages 46–57.

Vardi, M. (1996). An Automata-theoretic Approach to Linear Temporal Logic. *Lecture Notes in Computer Science*, 1043:238–266.