Exploitation des Symétries Dans La Vérification des Propriétés des Systèmes Concurrents

K. Ajami*   S. Haddad**   J-M. Ilié*

* LIP6 - CNRS ERS 587
Univ. Pierre & Marie Curie,
Tour 65-66, Bureau 204,
4, place Jussieu,
75252 Paris Cedex 05

e.mail:
Khalil.Ajami@lip6.fr,
Jean-Michel.Ilie@lip6.fr

** LAMSADE - CNRS URA 825
Univ. Paris Dauphine
Pl. du Maréchal De Lattre de Tassigny,
75775 Paris

e.mail:
haddad@lamsade.dauphine.fr

Résumé

La vérification des propriétés des systèmes concurrents peut être effectuée par la spécification et la vérification des formules de logique temporelle sur un graphe états-transitions modélisant le système. Le problème connu de l'explosion combinatoire, en espace et en temps, nécessite le développement de techniques efficaces pour réduire (par rapport à certaines propriétés) la taille du graphe à construire.

Concernant la logique temporelle, l'une des techniques les plus prometteuses fut initiée par Emerson & al [6]. Comme en [1,12,14], elle exploite les symétries de description existant entre les objets du système, cependant elle introduit l'idée que les symétries ne sont pas seulement detectables dans le système mais aussi dans toute formule décrivant une propriété de ce système.
De telles symétries sont définies par un groupe de permutations sur les objets du système de façon à assurer la préservation de la formule et le graphe d'états-transitions du système. Une relation d'équivalence entre les états du système peut donc être déduite, autorisant la construction de graphes quotients très réduits sur lesquels les vérifications sont assurées. Néanmoins, cette méthode de vérification exploite les symétries du système de façon très restrictive puisque seules les formules dont les symétries forment un groupe peuvent être vérifiées avec efficacité.

Notre but est de définir et d'utiliser des symétries plus grossières afin d'obtenir une meilleure réduction des graphes d'états-transitions représentés.

** Version étendue de l'article "Exploiting Symmetries In Linear Time Temporal Logic Model Checking: One Step Beyond" Publié dans: TACAS'98, Lisbon, Portugal, Avril 1998, Springer-Verlag, LNCS 1384.*

L'ensemble des symétries considérées ne constitue donc pas nécessairement un groupe mais permet d'exploiter le maximum de symétries possibles du système.

Dans notre approche nous commençons par analyser l'automate de Büchi modélisant les séquences valides vis à vis d'une propriété LTL à vérifier. Nous déterminons l'ensemble des symétries entre les états de l'automate en considérant que deux états sont symétriques s'ils induisent le même comportement courant et futur à une permutation près. De la même façon nous détectons, dans le modèle représentant le système, les symétries du modèle en considérant que deux états sont symétriques s'il sont égaux à une permutation près, à calculer en fonction des valeurs courantes des variables du système.
En appliquant ces relations sur le produit synchronisé de l'automate de la formule et du graphe qui modélise le système, nous construisons une structure quotient sur laquelle les séquences invalidant l'automate de la formule, sont détectées.

Dans cet article, nous prouvons que la vérification utilisant cette approche est équivalente à la vérification de la formule sur le graphe complet. Néanmoins, le calcul général des permutations a une complexité exponentielle ce qui réduit en quelques sorte les bénéfices réalisés par la réduction du graphe. C'est pourquoi nous proposons une approche alternative consistant à restreindre le temps de calcul des symétries à un temps polynomial en ne calculant qu'un sous ensemble des symétries exploitables. Bien que le produit synchronisé résultant puisse être d'une taille accrue par rapport à la structure quotient, nous prouvons qu'il possède les même capacités de vérification et trouvons qu'elle réalise dans la plupart des cas pratiques des réductions exponentielles.

L'article que nous présentons ici est une version étendue de celui présenté à TACAS'98. En particulier, une application a été ajouté pour confirmer l'intérêt de notre technique.

Domaine: Méthodes Formelles, Vérification.
Mots clés: Logique Temporelle, LTL, Symétries, Automates de Büchi, Vérification, Vérification du modèle.

Exploiting Symmetry In Linear Time Temporal Logic Model Checking: One Step Beyond

K. Ajami* S. Haddad** J-M. Ilié*

* LIP6 - CNRS ERS 587
Univ. Pierre & Marie Curie,
Tour 65-66, Bureau 204,
4, place Jussieu,
75252 Paris Cedex 05

e.mail: Khalil.Ajami@lip6.fr,
Jean-Michel.Ilie@lip6.fr

** LAMSADE - CNRS URA 825
Univ. Paris Dauphine

Pl. du Maréchal De Lattre de Tassigny,
75775 Paris

e.mail: haddad@lamsade.dauphine.fr

Abstract. Model checking is a useful technique to verify properties of dynamic systems but it has to cope with the state explosion problem. By simultaneous exploitation of symmetries of both the system and the property, the model checking can be performed on a reduced quotient structure [2,6,7]. In these techniques a property is specified within a temporal logic formula (CTL*) and symmetries correspond to permutations of objects which are obtained by either a syntactical checking [7] or semantical one through the state of the corresponding automaton [6]. We introduce here a more accurate method based on what we call the partial symmetries of the formula. Such symmetries are detected within the Büchi automaton of the formula and form a set of symmetries even if the fomula is not globally symmetrical i.e. there is no group of permutation (excepting the identity) operating on the formula. We define an appropriate quotient structure for the synchronized product of the Büchi automaton and the global state transition graph. We prove that model checking can be performed over this quotient structure leading to efficient algorithms.

Topic: Formal Methods.
Keywords: Temporal Logic, LTL, Symmetries, Büchi automata, Model Checking, Verification.

1. Introduction

Checking system correctness can be performed by the specification and the verification of temporal logic formulas over a state transition graph which models the system behavior. The well-known combinatorial explosion problem in space and time requires the development of efficient techniques in order to reduce the size of the graph to be built, with respect to some desired properties.

One of the most promising technique has been initiated by Emerson & al [6,7]. It exploits the symmetries of both the system and formula. Such a technique builds a quotient graph in which each node represents an equivalent class of states. The relation is induced by a subgroup of permutations preserving the state graph and the formula. In practice, the permutations act on a set of system processes with identical behavior. Previous works have been already developed focusing on the safeness properties [1,14,17,19]. Other developments include model checking algorithms [2,13], model checking under fairness constraints [8] and application to system bisimulation [16].

Looking carefully at the technique described in [6,7], it appears that currently, the CTL* model checking can make profit from the symmetries but in a restrictive way. Roughly speaking, the previous methods detect symmetries with respect to a subgroup of symmetries acting on either state formula like $f = \vee_{i \in I} f_i$ or temporal formulas like $f = \vee_{i \in I} \mathrm{F} f_i$ where f_i is a propositional formula involving the process *i*.

We generalize the previous methods by showing how these approaches are inefficient for formulas like $f = \wedge_{i,j \in I}^{i<j} (f_i \mathrm{U} f_j)$. Roughly speaking, formula describing such a property is not completely asymmetrical although its group of symmetries is reduced to the identity. However, this formula contains symmetries since it is applied on a set of identical processes. Such symmetries are called partial symmetries.

Hence, we show how partial symmetries can be exploited inside path subformulas of a CTL* formula. In this work, we limit the presentation to the case of LTL formulas. The general framework for branching time model checking can be developed using the iterated method of [10].

Unlike the approach presented in [6,7], *the considered Büchi automaton is not necessarily globally symmetric with respect to a predefined symmetry group*.

The starting point of our method is the analysis of the Büchi automaton associated with any LTL formula to be verified (see for instance [11]). Then we relate two states of the Büchi automaton if they represent the same current and future behavior up to a permutation of processes. Given a permutation, this state relation can be computed in polynomial time. Similarly, in the system model, two states are related by a permutation with respect to their current value of the system variables. By applying these relations on the synchronized product of the Büchi automaton and the global state transition graph, we define an appropriate quotient structure. We prove that model checking over this quotient is equivalent to model checking over the synchronized product however, the general computation of permutations wipes out the benefit of having a quotient structure (exponential complexity of computation). Therefore, we propose an alternative approach which computes, in polynomial time, an intermediate size structure. Such a structure has the same equivalence property as the quotient one and leads, in practical cases, to significant savings of space (even exponential).

The next sections are organized as follows: part 2 presents the model of computation and briefly recalls the temporal logic used to specify properties, it also presents the representation of a linear temporal logic formula by means of a Büchi automaton; part 3 presents the definition of system symmetries while part 4 presents the symmetries reflected in a temporal logic formula; part 5 is the analysis of the model checking using symmetries and the proof of its validity; part 6 contains the operational model checking approach using symmetries; part represents a realistic application; part 7 contains our conclusion and perspectives.

2. Model of Computation and Temporal Logic

We can apply our work on any system where symmetries are defined within a set of permutations. So, let us consider a simple model of system.

2.1. The Model

We deal with finite state concurrent systems composed of many processes. Processes are identified by indices. They may share global variables but differ from local ones. The structure of such a system is defined as follows:

Definition 2.1.1: Finite State Concurrent System
We present a finite state system using the temporal structure $M=(S, \Delta, I, V, D, L, S_0)$ where:
- S is the finite set of the states; $S_0 \subseteq S$ is the set of initial states;
- $\Delta \subseteq S \times S$ is the possible changes between states;
- I is the set of process indices;
- V is the set of system variables; it is composed of two distinct subsets, V_G, the set of global variables and V_L the set of local variables.
- D is the definition domain of variables.
- L is the state labeling function, L: $S \times (V_G \cup V_L \times I) \rightarrow D$ such that:
  - *(i)* $L(s,v_g)$ is the value of variable v_g of V_G in state s;
  - (ii) $L(s,v_l,i)$ is the value of variable v_l of V_L of process i in state s.

Atomic propositions are built from the association of a value to a variable.

Remark:
The structure of a system depends only on the value of the variables i.e. two different states must have at least one variable with different values.

Definition 2.1.2: Global and Local Atomic propositions
A global atomic proposition, is a pair $(v_g, d) \in V_G \times D$ whereas a local atomic proposition is a triplet $(v_l, i, d) \in V_L \times I \times D$ that depends on a process i.

We define $AP = \{p | \ p \in V_G \times D \cup V_L \times I \times D\}$ the set of atomic propositions built on the global and local variables.

We define $prop_S$: $S \rightarrow 2^{AP}$ such that $prop(s)$ is the set of propositions associated with s.

Definition 2.1.3: Atomic propositions holding in a state
Global (respectively local) atomic propositions hold at state s of S (noted $\models$) as follows: $s \models (v_g, d) \Leftrightarrow L(s, v_g) = d$; (respectively $s \models (v_l, i, d) \Leftrightarrow L(s, v_l, i) = d$).

In the following we recall some notions of temporal logic used to specify system properties.The translation of linear temporal formulas to Büchi automata is also presented.

2.2. Temporal Logic

In a propositional Temporal Logic, the non temporal portion of the logic is propositional logic. Thus formulas are built up from atomic propositions, which intuitively express, atomic facts about the underlying state of the concurrent system, truth-functional connectives and the temporal operators. Furthermore, when defining a system of temporal logic, two possible views of the system, can be considered, regarding the nature of time.

One is that the course of time is linear: at each moment there is only one possible future moment. The other is that time has a branching tree-like nature: at each moment, time may split into alternate courses representing different possible futures. In linear time, one reasons about sets of infinite sequences, while in branching time, one reasons about the possible futures of the current state leading to branching tree like structure.
In our work we are mainly interested by the linear time temporal logic formulas. However, the notion of branching time temporal remains the general framework in which our model checking can be extended. We use here two kinds of operators, temporal operators presented later and path quantifiers using the two symbols, A, E, to indicate respectively all or some paths.

2.2.1. Linear Temporal Logic (LTL)

A well-formed linear-time temporal logic, dealing with our system, is constructed from the set of atomic propositions *AP*, the standard boolean operators V (Or), ¬ (Not), and the temporal operators X (neXttime) and U (strong Until). Precisely, formulas are defined inductively as follows: (1) Every member of *AP* is a formula; (2) if φ and ψ are formulas then so are $\neg\varphi, \varphi \vee \psi$, Xφ, φ U ψ.
An interpretation for a linear-time temporal logic formula is an infinite word $\xi = x_0 x_1 ...$ over an alphabet 2^{AP}. For more precision, the elements of 2^{AP} are interpreted as assigning truth values to the elements of *AP*: elements in the set are assigned *true*, elements not in the set are assigned *false*. We note ξ_i the suffix of ξ starting at x_i. The semantics of LTL is defined in the following:

- $\xi \models \alpha$ iff $\alpha \in x_0$, for $\alpha \in AP$.
- $\xi \models \neg\varphi$ iff $\neg(\xi \models \varphi)$.
- $\xi \models \varphi \vee \psi$ iff ($\xi \models \varphi$ or $\xi \models \psi$).
- $\xi \models X\varphi$ iff $\xi_1 \models \varphi$.
- $\xi \models \varphi \, U \, \psi$ iff $\exists i \geq 0$ such that $\xi_i \models \psi$ and $\xi_j \models \varphi$ $0 \leq \forall j < i$.

As some abbreviations, one can introduce additional linear operators: the *eventuality* operator F where Fφ= *true* U φ, the *always* operator G where Gφ=¬F¬φ.

2.2.2. From LTL to Büchi automata

A Büchi automaton is a finite automaton which accepts infinite sequences. A sequence is accepted if, and only if, it is recognized by the automaton and meets infinitely often one of the accepting states (called also designated states).
It has been shown that any LTL formula can be translated to a Büchi automaton in order to perform efficient model checking. Indeed, Büchi automata are strictly more expressive than LTL formulas and equivalent to linear-time Mu-calculus [5,18,21].

Definition 2.2.3: Büchi automata
A Büchi automaton [6] is a tuple $A=(AP, B, \rho, B_0, Prop_B, F)$ where:

- B is a set of states. Each state b of B is defined by the set $Atom(b) \subseteq AP$.
- ρ: $B \rightarrow 2^B$ is a nondetermistic transition function.
- $B_0 \subseteq B$ is a set of starting states.
- $Prop_B$: $B \rightarrow 2^{AP}$ is the set of atomic propositions holding in B.
- $F \subseteq B$ is a set of accepting states.

3. Symmetries on Models

Given a permutation π: $I \to I$ on the set of process indices, we want to determine whether two states of the state transition graph are symmetric up to this permutation. Effectively, a permutation is said to be a symmetry if and only if it preserves the possible changes between states. We define the symmetries on the model represented by the structure *M=(S, Δ, I, V, D, L,* S_0*)*.

Definition 3.1: Symmetry on a State Transition Graph
A permutation π on *I*, is a symmetry iff:
(1) For each state $s \in S$, there is a unique state s' denoted $\pi(s)$ which satisfies:
(i) $\forall v_g \in V_G, L(\pi(s), v_g) = L(s, v_g)$;
(ii) $\forall i \in I, \forall v_l \in V_L, L(\pi(s), v_l, i) = L(s, v_l, \pi(i))$.
(2) Permutation π satisfies the following condition:
$(\forall s_1 \in S), (\forall s_2 \in S), ((s_1 \to s_2) \in \Delta \Leftrightarrow (\pi(s_1) \to \pi(s_2)) \in \Delta)$.

The group of symmetries defined on *M* is called the automorphisms group of *M* and denoted *Aut(M)*.

4. Symmetries on Formulas

In [6], the symmetries of a temporal logic formula to be verified are obtained by a syntactical checking while in [7], they result from the analysis of the corresponding Büchi automaton. By looking carefully at this method, it appears that symmetries of a CTL* formula are obtained in a restrictive way. Roughly speaking, many techniques are proposed based on the detection of a group of symmetries:

(1) *State symmetries* obtained from (sub)formulas like $V_{i \in I} f_i$ where f_i is propositional involving process *i*. Effectively, the symmetries resulting from formulas like $f = \mathrm{EF}(V_{i \in I} f_i)$, $f = \mathrm{EF}(\Lambda_{i \in I} f_i)$ constitute the group *Sym(I)*, the group of all the permutations between the elements of *I*. Those computed for formula like $f = \mathrm{EF} f_i$ constitute the group *Stab(i)* (the group of all the permutation between the elements of $I \setminus \{i\}$).
(2) The former approach fails to capture *Path symmetries* in LTL subformulas like $f = V_{i \in I} \mathrm{F} f_i$, $f = \Lambda_{i \in I} \mathrm{F} f_i$. Thus, the method of [6] introduces a complementary framework by detecting a group of symmetries acting on the states of Büchi automaton.

All these approaches are inefficient for formulas like $f = \Lambda_{i,j \in I}^{i<j} (f_i \mathrm{U} f_j)$ because the group of symmetries is reduced to the identity. However, the former formula contains symmetries since it is applied on a set of identical processes. Such symmetries are called partial symmetries and can be reflected in some states of its Büchi automaton.
In this section, we propose a more accurate method based on the exploitation of partial symmetries computed for some states of the automaton. Hence, we show that the existence of a group is not required to exploit symmetries.

We compute, the symmetries on a Büchi automaton, *A=(AP, B,* ρ*,* B_0*, E, F)*. The states equivalence can be detected using the relation defined as follows:

Definition 4.1: Permutation on a set of atomic propositions
Let π be a permutation on I. Let AP_1 be a set of atomic propositions, there is a set AP_2 denoted $\pi(AP_1)$ which satisfies:

$$\pi(AP_1)=AP_2=\{ (v_g,d') \mid \forall (v_g,d) \in Atom(b), \exists (v_g,d') \in Atom(b')\, where\, (d' = d)\} \cup$$
$$\{(v_l,j,d') \mid \forall i \in I, \forall (v_l,i,d) \in Atom(b), \exists j \in I, \exists (v_l,j,d') \in Atom(b')\, where\, (d' = d)\}$$

Definition 4.2: Equivalence of two states of a Büchi automaton
A relation $\mathcal{R}_\pi$ is the coarsest relation that defines the equivalence of two states of a Büchi automaton. It fulfills the following two requirements: $\forall b, b' \in V, b\mathcal{R}_\pi b'$ iff:
(1) There is a permutation π that satisfies the two conditions:

(i) $b \in \mathcal{F} \Leftrightarrow b' \in \mathcal{F}$;

(ii) $Atom(b') = \pi(Atom(b))$.

(2) $\forall b_1 [b \rightarrow b_1], \exists b'_1 [b' \rightarrow b'_1] \mid (b_1 \mathcal{R}_\pi b'_1)$.

Generally, $\mathcal{R}_\pi$ is not an equivalence relation. It can be computed in a polynomial time using a fixed-point computation starting with condition (1) and by applying (2).

Example 1: *Let us consider the following Büchi automaton representing the formula* $f = [(p_1 U p_3) \vee (p_2 U p_3)] \wedge (p_1 U p_2)$ for a system of three processes P_1, P_2, P_3, where p_1, p_2, p_3 are three atomic propositions.

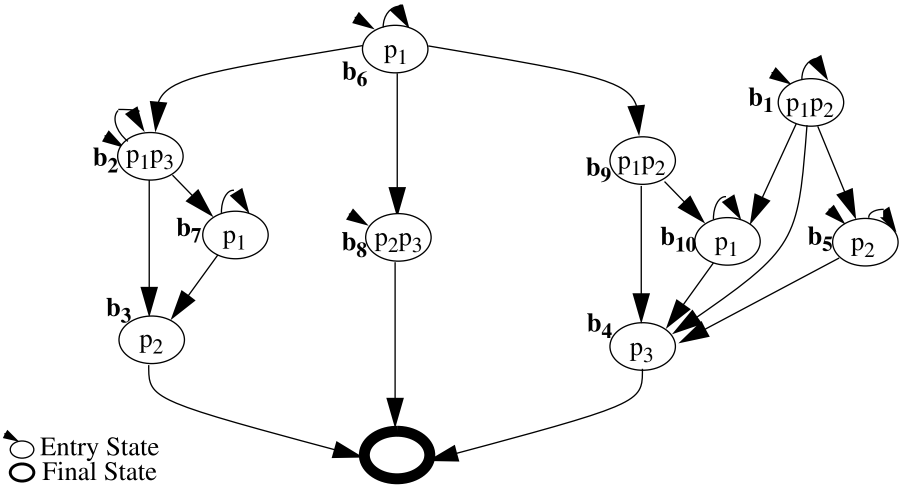

Fig. 1. Büchi automaton of the formula f

In this automaton, the only global symmetry group acting on the states is the identity. However, one can observe that states b_7 *and* b_{10} *are symmetrical with respect to definition 4.2 (permutation* π *such that* $\pi(1)=1$, $\pi(2)=3$, $\pi(3)=2$ *is used). Similarly, other symmetries can be detected between* b_3 *and* b_4, b_5 *and* b_{10} *etc. Conversely,* b_7 *and* b_6 *are identically labelled but not symmetrical.*

In the next section, we show how to perform an efficient model checking using the Büchi automaton representation and the proposed symmetries.

5. Analysis of Model Checking using symmetries

Classically, model checking is realized by (1) considering the Büchi automaton, $A_{\neg f}$ of the negation of formula f to be verified; (2) building the synchronized product of this automaton and the one which models the behavior of the system; (3) searching in the synchronized product a sequence which has an accepting state repeated infinitely often in order to prove that the negation of the formula holds. The meaning of such algorithm is that one must verify that any behavior of the system validates the formula. This algorithm can work in an "on-the-fly" fashion [11] so as to avoid the construction of the whole graph of the strongly connected components.

5.1. Synchronized Product

The synchronized product of M and $A_{\neg f}$ is noted $M \times A_{\neg f}$ and is defined as follows:

Definition 5.1.1: Synchronized Product

The synchronized product of $M=(S, \Delta, I, V, D, L, S_0)$ and $A_{\neg f}=(AP, B, \rho, B_0, E, F)$ is the automaton $M \times A_{\neg f}=(AP, \Theta, \Gamma, \Theta_0, \Phi)$ defined below:

- $\Theta=\{(s,b) \mid s \in S \wedge b \in B \wedge (prop_B(b) \subseteq prop_S(s))\}$;
- $\Theta_0= \{(s,b) \in \Theta \mid s \in S_0 \wedge b \in B_0 \wedge (prop_B(b_0) \subseteq prop_S(s_0))\}$;
- $((s_i,b_i) \to (s_j,b_j)) \in \Gamma$ iff $(s_i \to s_j) \in \Delta \wedge (b_i \to b_j) \in \rho$;
- $\Phi = \{(s,b) \in \Theta \mid s \in S \wedge b \in F \wedge (prop_B(b) \subseteq prop_S(s))\}$.

By means of such a product, we have to check if the required formula holds through M. Generally, the verification algorithm become less complex using the negation of the formula. Effectively, we can search if there is a path $M \times A_{\neg f}$ in which an accepting state is repeated infinitely often. In such an algorithm we do not have to search all the strongly connected component to verify the formula. It is sufficient to find one of them to prove that the negation holds and consequently the formula does not hold. We call such a method, the on-the-fly verification fashion.

Formally, the satisfaction relation of a formula is expressed as follows:

$M \models \neg f \Leftrightarrow \exists \pi = (s_0,b_0) \ldots (s_l,b_l) \ldots (s_m,b_m) \ldots (s_n,b_n)$ in $M \times A_{\neg f}$ where $l \le m < n$

such that: (1) $b_m \in F$;

(2) $(\forall i,j \in I, i \neq j, ((s_i,b_i) = (s_j,b_j) \Leftrightarrow \{i,j\} = \{l,n\}))$

Roughly speaking, we present the path π as a strongly connected component using the following equivalence: for any different indices i,j, $(m_i,b_i)=(m_j,b_j)$ means that $i=l$ and $j=n$.

5.2. Quotient Structure

In order to reduce the size of the synchronized product structure, we only consider canonical representatives of the symmetrical states instead of all the states.

Consequently, we build a graph of representatives with respect to a symmetry relation, $\mathcal{R}$, defined on $M \times A_{\neg f}$ as follows:

Definition 5.2.1: Symmetry Relation, $\mathcal{R}$, Defined on $M \times A_{\neg f}$

$\forall s, s' \in S, \forall b, b' \in V$ such that $s \models$Atom(b) and $s' \models$Atom(b'),

$(s, b)\, \mathcal{R}\, (s', b')$ iff $\exists \pi \in Aut(M)$ such that $b \mathcal{R}_\pi b'$ and $\pi(s) = s'$.

Observe that $\mathcal{R}$ is an equivalence relation since it is defined on the group *Aut(M)*. Therefore, we can define the quotient structure of the synchronized product $M \times A_{\neg f}$ denoted $\overline{M \times A_{\neg f}} = (M \times A_{\neg f}) / \mathcal{R}$ as follows:

Definition 5.2.2: The Quotient Structure $\overline{M \times A_{\neg f}}$

The quotient structure $\overline{M \times A_{\neg f}}$ is defined by means of the representatives of the state orbits of $M \times A_{\neg f}$. The orbit of $(s, b) \in \Theta$ is defined by the set:

$\theta(s, b) = \{ (s', b') \mid \exists \pi \in Aut(M), (\pi(s) = s') \wedge (b \mathcal{R}_\pi b')\, where, s' \models Atom(b') \}$.

From each orbit $\theta(s,b)$, we pick an arbitrary representative denoted $(\overline{s, b})$.

The representative can be efficiently implemented by defining a canonical representation based on a lexicographical order [1].

5.3. Model Checking Correctness

In this section we validate our approach by showing that the model checking based on the proposed quotient synchronized product is equivalent to the one performed by means of the ordinary structure. Intuitively, we prove that the existence of an accepting state repeated infinitely often in the quotient structure is equivalent to the existence of an accepting state repeated infinitely often in the ordinary synchronized product. Hence, we can prove the satisfaction of temporal logic formulas by using our approach of symmetry.

We start our proof by the correspondence between both the quotient and the ordinary structures of the synchronized product.

Let $M \times A_{\neg f}$ represents the structure resulting from the synchronized product of the state transition graph and the automaton and let $\overline{M \times A_{\neg f}} = (M \times A_{\neg f}) / \mathcal{R}$ be its quotient structure with respect to the relation $\mathcal{R}$ introduced in definition 5.2.1. For each symbolic path in the quotient structure there is an ordinary path in the synchronized product such that the corresponding states of the two paths are symmetrical with respect to $\mathcal{R}$:

Lemma: Correspondence Lemma

$$\exists (\overline{s_0, b_0}), \ldots, (\overline{s_n, b_n}) \in \overline{M \times A_{\neg f}} \Leftrightarrow \exists (s'_0, b'_0), \ldots, (s'_n, b'_n) \in M \times A_{\neg f} \text{ such that } 0 \le \forall i \le n, (s'_i, b'_i)\, \mathcal{R}\, (s_i, b_i)$$

Proof:

the $\Leftarrow$ direction is immediate from the definition of quotient structure.

For the $\Rightarrow$ direction, we proceed by induction on, n, the length of the path:

(i) $n=0$, This case is very simple since for any (s'_0, b'_0) such that $(s'_0, b'_0)\, \mathcal{R}\, (s_0, b_0)$, the lemma is proved.

(ii) We assume that the lemma is proved for a given length equal to *n* and we verify that it is proved for a length equal to *n+1*. Let us consider $(\overline{s_0, b_0}), \ldots, (\overline{s_n, b_n})$ in $\overline{M \times A_{\neg f}}$, and let us recall that by assumption, we have $(s'_0, b'_0), \ldots, (s'_n, b'_n)$ in $M \times A_{\neg f}$ such that $0 \le \forall i \le n, (s'_i, b'_i) \mathcal{R} (s_i, b_i)$. Let us consider an edge $(\overline{s_n, b_n}) \to (\overline{s_{n+1}, b_{n+1}})$ of $\overline{M \times A_{\neg f}}$, thus, $\exists (s''_n, b''_n) \to (s''_{n+1}, b''_{n+1})$ in $M \times A_{\neg f}$ such that $(s''_n, b''_n) \mathcal{R} (s_n, b_n)$ and $(s''_{n+1}, b''_{n+1}) \mathcal{R} (s_{n+1}, b_{n+1})$. We have already $(s'_n, b'_n) \mathcal{R} (s_n, b_n)$. So, $(s'_n, b'_n) \mathcal{R} (s''_n, b''_n)$ and from the definition of the relation $\mathcal{R}$: $\exists \pi, s'_n = \pi(s''_n)$ and $b'_n \mathcal{R}_\pi b''_n$. Hence, there is a state $s'_{n+1} = \pi(s''_{n+1})$ and an automaton state b'_{n+1} such that $b'_{n+1} \mathcal{R}_\pi b''_{n+1}$ where:

$s'_{n+1} = \pi(s''_{n+1}) \models \pi(Atom(b''_{n+1})) = Atom(b'_{n+1})$.

Therefore, (s'_{n+1}, b'_{n+1}) is a state of $M \times A_{\neg f}$ and since, there is an arc $(s''_n, b''_n) \to (s''_{n+1}, b''_{n+1})$, the arc $(s'_n, b'_n) \to (s'_{n+1}, b_{n+1})$ is an arc of the same structure also.

From this lemma, we now prove the equivalence of the existence of paths verifying the formula, in both the quotient and the ordinary synchronized product:

Theorem: The two following statements are equivalent:

(i) There is a path $(\overline{s_0, b_0}) \ldots (\overline{s_l, b_l}) \ldots (\overline{s_m, b_m}) \ldots (\overline{s_n, b_n})$ in $\overline{M \times A_{\neg f}}$ where $l \le m < n$ such that: (1) $\overline{b_m} \in F$; (2) $\forall i \ne j, ((\overline{s_i, b_i}) = (\overline{s_j, b_j})) \Leftrightarrow (\{i, j\} = \{l, n\})$.

(ii) There is a path $(s'_0, b'_0) \ldots (s'_l, b'_l) \ldots (s'_m, b'_m) \ldots (s'_n, b'_n)$ in $M \times A_{\neg f}$ where $l \le m < n$ such that: (1) $b_m \in F$; (2) $\forall i \ne j, ((s'_i, b'_i) = (s'_j, b'_j)) \Leftrightarrow (\{i, j\} = \{l, n\})$.

Proof:

(**1**) (i) $\Rightarrow$ (ii): Let us consider $(\overline{s_0, b_0}) \ldots (\overline{s_l, b_l}) \ldots (\overline{s_m, b_m}) \ldots (\overline{s_n, b_n})$ the shortest path that verifies (i1) in $\overline{M \times A_{\neg f}}$. From the correspondence lemma there is a corresponding path $(s'_0, b'_0), \ldots, (s'_l, b'_l), \ldots, (s'_m, b'_m), \ldots, (s'_n, b'_n)$ in $M \times A_{\neg f}$. We prove that it verifies the following two statements:

(ii1) $b'_m \in F$, effectively, symmetries built on the automaton preserve accepting states;

(ii2) Two directions have to be proved. The $\Rightarrow$ direction is straightforward since $(s'_i, b'_i) = (s'_j, b'_j) \Rightarrow ((\overline{s_i, b_i}) = (\overline{s_j, b_j}))$, consequently we deduce from (i2 that *{i,j}={l,n}*. For the $\Leftarrow$ direction, we have *{i,j}={l,n}*, so from (i2), $(\overline{s_i, b_i}) = (\overline{s_j, b_j})$. By assumption, *{i,j}={l,n},* so, $(\overline{s_l, b_l}) = (\overline{s_n, b_n})$ from (i2). The proof is now made by contradiction assuming that $(s'_l, b'_l) \ne (s'_m, b'_m)$. In this case, $(s'_l, b'_l) \mathcal{R} (s'_m, b'_m)$ since their representatives are equals. Consequently, from the correspondence lemma, we have an infinite path in which there is an infinite number of subpaths of the form $(s'_{n+k(n-l)}, b'_{n+k(n-l)}) \dots (s'_{n+(k+1)(n-l)}, b'_{n+(k+1)(n-l)})$ where *k=0,1,...* and for $l \le \forall i \le n$, $(s'_{i+k(n-l)}, b'_{i+k(n-l)}) \mathcal{R} (s'_i, b'_i)$. Because we deal with finite state transition graph, this infinite path must contain a circuit. Let us consider $(s'_0, b'_0), \dots, (s'_x, b'_x), \dots, (s'_y, b'_y)$ the shortest path that meets twice the same state where: $(s'_x, b'_x) = (s'_y, b'_y)$ and $x \ge l, y \ge n$. To simplify the proof we consider x<y and we denote *f(x)=x-k(n-l), f(y)=y-k(n-l)* such that $f(x), f(y) \in [l, n]$. Since $((s'_x, b'_x) = (s'_y, b'_y)) \Rightarrow ((\overline{s_x, b_x}) = (\overline{s_y, b_y})) \Rightarrow (\overline{s_{f(x)}, b_{f(x)}}) = (\overline{s_{f(y)}, b_{f(y)}})$ we have two cases. In the first one: *f(x), f(y)* are from *[l,m[* or *]m,n]*. By considering

$(s'_0, b'_0), \dots, (s'_l, b'_l), \dots, (s'_{f(x)}, b'_{f(x)}), (s'_{f(y)+1}, b'_{f(y)+1}), \dots, (s'_m, b'_m), (s'_n, b'_n)$

$(s'_0, b'_0), \dots, (s'_l, b'_l), \dots, (s'_m, b'_m), (s'_{f(x)}, b'_{f(x)}), (s'_{f(y)+1}, b'_{f(y)+1}), \dots, (s'_n, b'_n)$

with respect to the position of *f(x), f(y)* in the domain, we have a shortest path than $(s'_0, b'_0), \dots, (s'_l, b'_l), \dots, (s'_m, b'_m), (s'_n, b'_n)$ that verifies the two conditions (ii1) and (ii2) which is opposite to the initial assumption. In the second case: *f(x)* is from *[l,m[* and *f(y)* from *[m,n]*. By considering the following path that verify (ii1) and (ii2), $(s'_0, b'_0), \dots, (s'_{f(x)}, b'_{f(x)}), \dots, (s'_m, b'_m), \dots, (s'_{f(y)}, b'_{f(y)})$, we have a shortest path than $(s'_0, b'_0), \dots, (s'_l, b'_l), \dots, (s'_m, b'_m), (s'_n, b'_n)$ which is contradictory with the initial assumption. Consequently $(s'_l, b'_l) = (s'_m, b'_m)$.

(**2**) (ii) $\Rightarrow$ (i): Let us consider the set of paths:

$\Pi = \{(s_0,b_0)\dots(s_l,b_l)\dots(s_m,b_m)\dots(s_n,b_n)$ *such that* $(s_l,b_l)\ \mathcal{R}\ (s_m,b_m)$ *and* $b_m \in F\}$.

$\Pi \ne \varnothing$ from the assumption (i1). Let us consider $\pi = (s_0,b_0)\dots(s_n,b_n)$ one of the shortest path of Π. From the correspondence lemma, $\exists \bar{\pi} = (\overline{s'_0, b'_0}) \dots (\overline{s'_n, b'_n})$ a representative path where $(s_i, b_i) R (s'_i, b'_i)$. Consequently, $\bar{\pi}$ verifies (i1) $(\overline{s'_l, b'_l}) = (\overline{s'_n, b'_n})$ and (i2) $b'_m \in F$. We have to prove that, $\forall \{i,j\} \ne \{l,n\} \Rightarrow (\overline{s'_i, b'_i}) \ne (\overline{s'_j, b'_j})$. We suppose that $\exists \{i,j\} \ne \{l,n\}, (\overline{s'_i, b'_i}) = (\overline{s'_j, b'_j})$, hence, either, $(s'_i, b'_i) = (s'_j, b'_j)$ which is impossible from (ii2), or, $(s'_i, b'_i) \mathcal{R} (s'_j, b'_j)$. Hence, three cases can appear. in the first one we have $(i<j\le l) \vee (l \le i<j<m) \vee (m<i<j<n)$: by considering the path $(s_0,b_0)\dots(s_i,b_i)(s_{j+1},b_{j+1})\dots(s_n,b_n)$ we have a shortest path than π belonging to Π which is opposite to the initial assumption. In the second case we have $i<l<j<m<n$: by considering $(s_0,b_0)\dots(s_i,b_i)(s_{j+1},b_{j+1})\dots(s_n,b_n)(s_{l+1},b_{l+1})\dots(s_j,b_j)$ we have a shortest path than π belonging to Π which is opposite to the initial assumption. In the last case we have $l<j<m<j<n$: by considering $(s_0,b_0)\dots(s_i,b_i)\dots(s_m,b_m)\dots(s_j,b_j)\dots(s_n,b_n)$ we have also a shortest path than π belonging to Π which is contradictory for the initial assumption. Consequently, $\forall \{i,j\} \ne \{l,n\} \Rightarrow (\overline{s'_i, b'_i}) \ne (\overline{s'_j, b'_j})$.

5.4. Consistent Graph

The quotient structure is the smallest structure that can be built to perform model checking using symmetries. In the worst case, $\{\mathcal{R}_\pi\}_\pi$ requires an exponential time construction therefore, we propose a new approach based on the construction of an intermediate structure, called *consistent graph* which does not require the computation of all the relations induced by the symmetries.

In such a graph, (1) reachability is preserved with respect to the ordinary synchronized product; (2) the transition relation of the ordinary synchronized product is preserved accordingly to the symmetry relation $\mathcal{R}$ in the consistent graph; (3) the transition relation of the consistent graph product is preserved accordingly to the symmetry relation $\mathcal{R}$ in the ordinary synchronized. Hence, All paths are preserved with respect to $\mathcal{R}$.
Such a consistent graph will be used in section to propose an efficient model checking in polynomial time.

Definition 5.4.1: Consistent Graph
Let $G = M \times A_{\neg f} = (AP, \Theta, \tau, \Theta_0, \Phi)$ and let $G' = (AP', \Theta', \tau', \Theta'_0, \Phi')$ we call G' consistent with G iff:
(1) $\forall (s,b) \in \Theta$ such that *(s,b)* is reachable from $(s_0, b_0) \in \Theta_0$, $\exists (s', b') \in \Theta'$ reachable from $(s'_0, b'_0) \in \Theta'_0$ such that $(s', b') \mathcal{R} (s, b)$.
(2) $\forall (s,b) \in \Theta$ such that *(s,b)* is reachable from $(s_0, b_0) \in \Theta_0$, $\forall (s', b') \in \Theta'$ reachable from $(s'_0, b'_0) \in \Theta'_0$ such that $(s', b') \mathcal{R} (s, b)$:

if $(s,b) \to (s_1, b_1) \in \tau$ then $\exists (s'_1, b'_1) \in \Theta'$ where $(s'_1, b'_1) \mathcal{R} (s_1, b_1)$ such that $(s', b') \to (s'_1, b'_1) \in \tau'$.

(3) $\forall (s,b) \in \Theta$ such that *(s,b)* is reachable from $(s_0, b_0) \in \Theta_0$, $\forall (s', b') \in \Theta'$ reachable from $(s'_0, b'_0) \in \Theta'_0$ such that $(s', b') \mathcal{R} (s, b)$:

if $(s', b') \to (s'_1, b'_1) \in \tau'$ then $\exists (s_1, b_1) \in \Theta$ where $(s_1, b_1) \mathcal{R} (s'_1, b'_1)$ such that $(s,b) \to (s_1, b_1) \in \tau$.

The following lemma highlights the correspondence between the quotient structure $\overline{M \times A_{\neg f}}$ and the graph consistent with $M \times A_{\neg f}$.

Lemma: Consistent Graph Correspondence

$\exists (\overline{s_0, b_0}), \ldots, (\overline{s_n, b_n}) \in \overline{M \times A_{\neg f}} \Leftrightarrow \exists (s'_0, b'_0), \ldots, (s'_n, b'_n) \in G'$ consistent with $G = M \times A_{\neg f}$ such that $0 \le \forall i \le n$, $(s'_i, b'_i) \mathcal{R} (s_i, b_i)$.

Proof: It is similar to the one used for the correspondence lemma (section 5.3).

Hence, we can prove the model checking equivalence between the quotient structure and the graph consistent with the ordinary structure using correspondence lemma of the consistent graph.

Theorem: The two following statements are equivalent:
(i) There is a path $(\overline{s_0, b_0}) \ldots (\overline{s_l, b_l}) \ldots (\overline{s_m, b_m}) \ldots (\overline{s_n, b_n})$ in $G = \overline{M \times A_{\neg f}}$ where

$l \le m < n$ such that:

(1) $\overline{b_m} \in F$; (2) $\forall i \ne j, ((\overline{s_i, b_i}) = (\overline{s_j, b_j})) \Leftrightarrow (\{i,j\} = \{l,n\})$.

(ii) There is a path $(s'_0, b'_0) \dots (s'_l, b'_l) \dots (s'_m, b'_m) \dots (s'_n, b'_n)$ in G' consistent with $G = M \times A_{\neg f}$ where $l \le m < n$ such that:

(1) $b_m \in F$; (2) $\forall i \ne j, ((s'_i, b'_i) = (s'_j, b'_j)) \Leftrightarrow (\{i,j\} = \{l,n\})$.

Proof: This proof is similar to the one presented in the theorem of model checking equivalence (section 5.3) using the consistent graph correspondence lemma.

6. Operational Approach

The construction of a quotient structure is performed by checking, for each node built during the synchronized product, whether it is symmetrical with an already computed one. For this, an equivalence test has to be performed in an exponential time *O(n!)* (in the worst case), where n is the number of processes. Clearly, this would damage the benefit to have a condensed structure.

The following section introduces an operational approach in order to reduce the complexity of the construction algorithms. Nevertheless, the resulting state transition graph may have a larger size than the quotient structure because we compute only a reduced subset of symmetries. However, it is a consistent structure with both the ordinary and the former quotient structure, thus the model checking can be performed equivalently.

This section aims at presenting efficient algorithms to compute symmetries and to construct the consistent graph.

6.1. *ij-Symmetry* on Büchi automata

We now define a set of symmetries called the *ij-symmetries* in such a way that it represents a subset of $\mathcal{R}$.

6.1.1. *ij-Symmetry* Definition

Let $A=(AP, B, \rho, B_0, E, F)$. A relation $\mathcal{R}_{i,j}$ is a coarse relation that defines the equivalence of two states of a Büchi automaton with respect to two given processes. It fulfills the following definition:

A relation $\mathcal{R}_{i,j}$ is the relation $\mathcal{R}$ defined in 5.2.1 with respect to a the permutation $s = s_{ij}: I \to I$ such that $s_{ij}(i)=j$, $s_{ij}(j)=i$ and for each k where $k \ne i, j, s_{ij}(k) = k$.

From this definition we define a symmetry relation, $\mathcal{R}_{in}$, (The inner symmetry relation) with respect to I as follows: $\forall i,j \in I, (b,i)\mathcal{R}_{in}(b',j) \Leftrightarrow b\mathcal{R}_{i,j}b'$ where $\mathcal{R}_{in} = \left(\bigcup_{i,j} \mathcal{R}_{i,j}\right)^*$.

Hence, the inner symmetries presented by the relation $\mathcal{R}_{in}$ constitutes a subset of the set of symmetries presented by $\mathcal{R}$ such that: $b\mathcal{R}_{in}b' \Rightarrow b\mathcal{R}b'$.

We define the function: M_{ij}: $B \times B \to \{0,1\}$ such that $M_{ij}(b_k,b_l)=1$ if $b_k\mathcal{R}_{ij}b_l$ otherwise $M_{ij}(b_k,b_l)=1$. We use boolean matrix called the ij-matix constructed on the states of the automaton in ordre to represent the functions M_{ij}, $\forall i,j \in I$. Such matrix are used in the construction of the reduced graph.

Example 2:
As an example we can detect the ij-symmetries defined on the automaton of figure 2. We represent ij-symmetries by the following ij-matrix:

$$\mathcal{R}_{12} = \begin{array}{c|cccccccccc}
M_{12} & b_1 & b_2 & b_3 & b_4 & b_5 & b_6 & b_7 & b_8 & b_9 & b_{10} \\
\hline
b_1 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
b_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
b_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
b_4 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 \\
b_5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} \\
b_6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
b_7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
b_8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
b_9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
b_{10} & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0
\end{array}
, \mathcal{R}_{23} = \begin{array}{c|cccccccccc}
M_{23} & b_1 & b_2 & b_3 & b_4 & b_5 & b_6 & b_7 & b_8 & b_9 & b_{10} \\
\hline
b_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
b_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 \\
b_3 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 \\
b_4 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
b_5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
b_6 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 \\
b_7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} \\
b_8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\
b_9 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
b_{10} & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0
\end{array}$$

The computation of the inner symmetry using the transitive closure is made as follows:
$\forall m, n \in I, \mathcal{R}_{mn} \in \left(\underset{i,j \in I}{\cup} \mathcal{R}_{ij} \right)^2 \Leftrightarrow \exists k \in I, \mathcal{R}_{mn} = \mathcal{R}_{mk} \times \mathcal{R}_{kn}$. *For example:*

$$\mathcal{R}_{13} = \mathcal{R}_{12} \times \mathcal{R}_{23} = \begin{array}{c|cccccccccc}
M_{23} & b_1 & b_2 & b_3 & b_4 & b_5 & b_6 & b_7 & b_8 & b_9 & b_{10} \\
\hline
b_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
b_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
b_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
b_4 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
b_5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\
b_6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
b_7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
b_8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
b_9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
b_{10} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}$$

Based on $\mathcal{R}_{in}$, the next section proposes an efficient algorithm for the determination of the symmetries which are reflected in a Büchi automaton.

6.1.2. *ij-Symmetry* Computation Algorithms

The computation of the inner symmetry starts from the computation of the *ij-symmetry* presented in the definition .
Let f be a temporal specification formula and let $A_{\neg f}$ be the representation of its negation in terms of Büchi automaton.

(1) Firstly, we calculate, for a pair (i,j) of process indices, an initial partition of the states of $A_{\neg f}$ using the definition of *ij-permutation*. This results in a set of pairs of state that verify the *ij-permutation*.
(2) Secondly, we restrain the computed *ij-permutations* to $\mathcal{R}_{i,j}$ symmetry by checking the preservation of the transition relation (using definition 5.2.1). For each state, the set of $\mathcal{R}_{i,j}$ symmetries is saved.
(3) This algorithm is repeated for each pair i,j of process indices.

Let $B=\{b_1 \ldots b_m\}$ be the states of $A_{\neg f}$ where $|B|=m$ is the number of states. Let I be the set of process indices such that $|I|=n$.
We construct the set of ij-matrices that represent the symmetry relation between the states of B. We note Mat_{ij} the matrix representing the symmetry $\mathcal{R}_{i,j}$.

In the following, we present the function R_{ij} which compute the *ij-symmetry* in the automaton. This function calls two functions:

(1) function *ij-permut*(b_1,b_2) which checks the *ij-permutation* for two given states b_1 and b_2.

(2) The second one $R_{ij}(b_1,b_2)$ which checks if the two states, b_1, b_2, are *ij-symmetrical*.

Observe that The function *Succ(b)* computes all the successors of a given state *b*.

Algorithm 1: $R_{ij}()$
Input : The set states of the automaton
Output : The set of ij-matrix.

```
BEGIN
    FOR each paire b_d,b_k  from V DO Mat_ij[b_d, b_k] = R_ij({(b_d, b_k)},b_d, b_k);
```

Algorithm 2: $R_{ij}(Path, b_d, b_k)$
Input : Path used to collect states already visited; Two states of the automaton.
Output : Boolean Value that determines if the two states are ij-symmetrical.

```
BEGIN
    IF ij-permut(b_d,b_k) THEN
    BEGIN
            IF (Succ(b_d)=∅ AND Succ(b_k)=∅ ) Then          return 1;
            FOR [each b_d' in Succ(b_d) and each b_k' in Succ(b_k)] DO
                    IF ( b_d' , b_k' ) is not in Path THEN
                    BEGIN          Path=Path ∪ ( b_d' , b_k' );
                                   return R_ij(Path,b_d, b_k);
                    END;
                    ELSE           return 1;
    END;
    return 0;
END;
```

It worth noting that the complexity of the previous algorithm is $O(m^5)$ for a given *i,j*. Hence, the determination of all the *ij-symmetry* have a complexity of $O(n^2 \times m^5)$ which means a polynomial complexity. Furthermore, the computation of $\mathcal{R}_{in} = \left(\bigcup_{\mathrm{i,j}} \mathcal{R}_{i,j}\right)^*$ can be restricted to have, also, a polynomial complexity.

6.2. Construction of the Consistent Graph

We compute the symmetries on the synchronized product in order to build the consistent graph. Such symmetries are symmetries of the model and must be an inner symmetries with respect to the considered Büchi automaton.

Let $A_{\neg f}$ be a Büchi automaton and let *M* be the structure of the state transition graph. In the following, we define the symmetries of the system used to compute the reduced structure. They let the state of the automaton invariant in each step the product $M \times A_{\neg f}$:

Definition 6.2.1: Symmetry $\mathcal{R}'$ defined on $M \times A_{\neg f}$

$\forall s, s' \in M, \forall b \in B$ such that $s \models Atom(b)$ *and* $s' \models Atom(b)$ we have
$(s, b)\mathcal{R}'(s', b) \Leftrightarrow \exists \pi \in Aut(M), \mathcal{R}_\pi \in \mathcal{R}_{in}, b\mathcal{R}_\pi b \wedge \pi(s) = s'$.

The relation, $\mathcal{R}'$, is used to build a reduced graph $G' = (AP', \Theta', \tau', \Theta'_0, \Phi')$ consistent with $M \times A_{\neg f}$ =(AP, Θ, τ, Θ_0, Φ). The algorithm presented in the following.

Algorithm 3: Consistent Graph Constructor

BEGIN

<u>/ Symmetry Computation */</u>*

- FOR each i,j from I DO Rij();

- Polynomial computation of $\mathcal{R}_{in}$;

- FOR each equivalence class of $A_{\neg f}$ states DO Choose a representative $\hat{b}$;

<u>/ Consistent Graph Construction: */</u>*

FOR each (s_0,b_0) from Θ_0 such that $s_0 \models \hat{b}_0$ DO

BEGIN

Compute the symbolic representative $\overline{(s_0, b_0)}$ using the $\mathcal{R}'$;

$\Theta' = \Theta' \cup \overline{(s_0, b_0)}$;

Push($\overline{(s_0, b_0)}$);

END; / FOR */*

rs = Pop();

WHILE $rs \neq \emptyset$ DO / Stack is not empty */*

BEGIN

WHILE rs in Θ' DO rs = Pop();

FOR each arc $(rs \rightarrow (s, b)) \in \rho$ DO

BEGIN

Compute $\overline{(s, b)}$;

IF $\overline{(s, b)}$ is not in Θ' THEN

BEGIN

$\Theta' = \Theta' \cup \overline{(s, b)}$;

Push($\overline{(s, b)}$);

END; / IF */*

$\rho' = \rho' \cup (rs \rightarrow \overline{(s, b)})$;

END; / FOR */*

rs = Pop();

END; / WHILE */*

END. / ALGORITHM */*

The complexity of our model checking using symmetries is strongly dependent on the complexity of the former algorithm which have a polynomial complexity.

7. Application: Example of ij-symmetries detected on a formula

Let us consider a set of processes such that each one may send a request for accessing a critical section. Every process knows the other requests so as to cancel its proper request, in the case when another process reach the critical section. We model the problem by the following three states for any process i of the system: *Idle*$_i$, *Request*$_i$, *Access*$_i$ (I_i, R_i, A_i respectively). We assume that the system is fair, but the request will be served according to the minimum value of the indices. Therefore, the property to check is:

$$f = G[\Lambda_{i \neq j} \neg(C_i \wedge C_j)] \wedge G[\Lambda_i (R_i \rightarrow R_i U C_i)] \wedge G[\Lambda_{i<j} (R_i \wedge R_j \rightarrow R_j U C_i)]$$

In such formula, the first of the conjuction describes the mutual exclusion property (When a process has the access to the critical section, the other ones cannot issue a request). The second one describes the fairness property (each request of the critical section will be served). While the last part describes the access management according the minimum value of the indices.

Our verification procedure is based on the synchronized product between the state space representing the system and the automaton representing the negation of the formula.

In the following we present the automaton of the previous formula in order to compute the reduced synchronized product by means of the symmetries detected on both the state space of the system and the automaton:

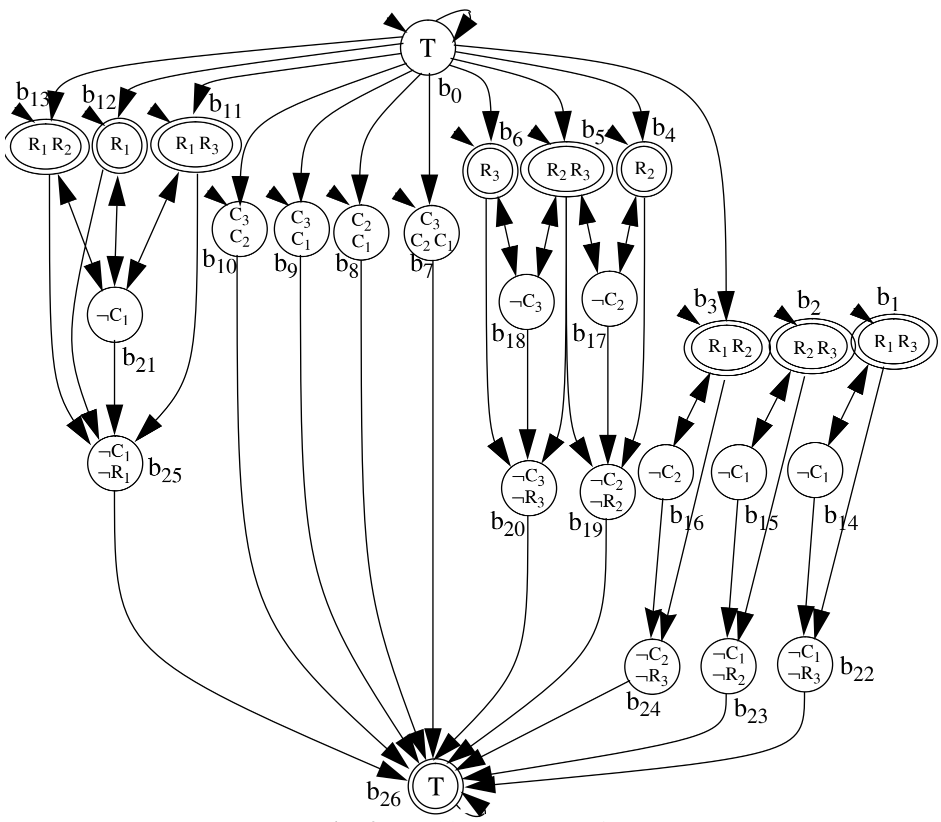

Fig. 2. Büchi automaton of $\neg f$

Examples of ij-symmetries that can be detected on this automaton:

- $b_4\,\mathcal{R}_{12}\,b_{12}$, - $b_9\,\mathcal{R}_{12}\,b_{10}$, - $b_{14}\,\mathcal{R}_{12}\,b_{16}$, - $b_{17}\,\mathcal{R}_{12}\,b_{21}$, - $b_{22}\,\mathcal{R}_{12}\,b_{24}$, ...
- $b_6\,\mathcal{R}_{13}\,b_{12}$, - $b_8\,\mathcal{R}_{13}\,b_{10}$, - $b_{18}\,\mathcal{R}_{13}\,b_{21}$, - $b_{20}\,\mathcal{R}_{12}\,b_{25}$, ...
- $b_{17}\,\mathcal{R}_{23}\,b_{18}$, - $b_8\,\mathcal{R}_{23}\,b_9$, - $b_4\,\mathcal{R}_{23}\,b_6$, - $b_5\,\mathcal{R}_{23}\,b_5$, ...

In the following we present the construction of the reduced synchronized product. We make the application on a portion of the state space presented in the following. To use the same notation, we substitute proposition I_i meaning that process *i* is in *Idle* state by proposition $\neg C_i \neg R_i$ which means that the process is neither in the request state nor in the critical section:

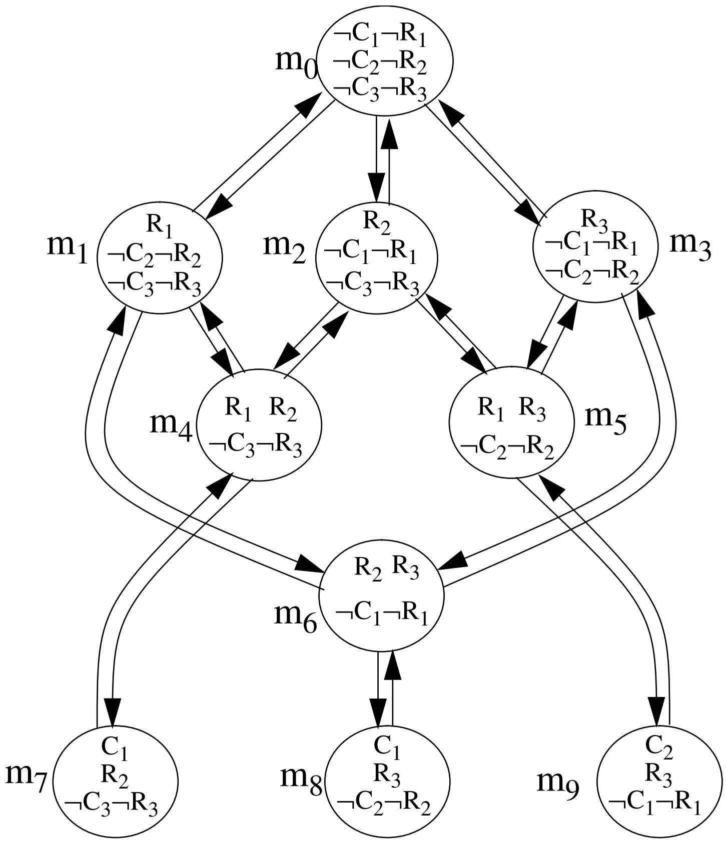

Fig. 3. A portion of the state space of the system

A portion of the ordinary synchronized product (without considering symmetries) is presented in figure 4. The reduced structure obtained when using symmetries is presented in the same figure but in bold lines:

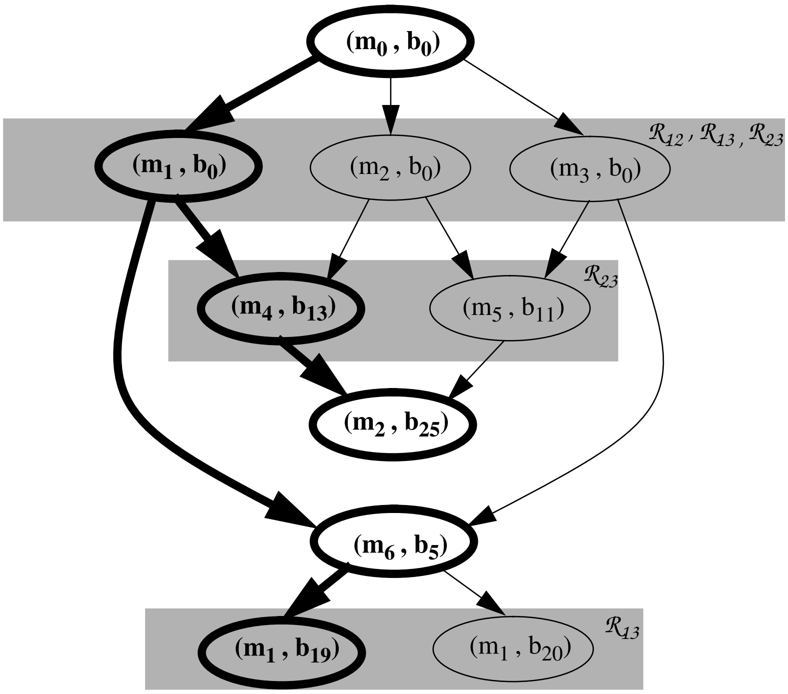

Fig. 4. an ordinary and a reduced (in bold) portion of synchronized product

8. Conclusion and Perspectives

We have described two frameworks for performing efficient LTL model checking. Both of them exploits the existence of symmetries reflected in the system and in the specification formula to be checked. With the first one, we show how to build the most aggregated structure by using the largest available symmetry relation. Such technique could be computed using algorithms which, in the worst case, would have an exponential complexity. The second framework computes a subset of symmetries with polynomial complexity algorithms inducing a less condensed structure.

In comparison, the method proposed in [6,7] can be considered as a restrictive case requiring the definition of a symmetry group.

Using the symmetry approach, two cases appear as the two extreme limits: the best one where the structural symmetries of the system are entirely used, causing a maximal aggregation of states and the worst case in which any set of symmetrical objects is reduced to a singleton, leading the reduced structure to be as large as the ordinary one.

We now aim at extending our methods to deal with specifications having nothing but partial symmetries [12]. In such specifications, runs sometimes depend on the process identities (i.e. static priorities based on identities), and sometimes not.

The implementation of this work is derived from GreatSPN2.0 developed by Chiola & Gaëta from the university of Torino-Italy. It will be integrated into the CPN-AMI tool developed by the group of distributed and cooperative systems of LIP6.

9. References

[1] G. Chiola, C. Dutheillet, G. Franceschinis, S. Haddad, “On Well-formed Colored Nets and their Symbolic Reachability Graph”, proc. of 11th International Conference on Application and Theory of Petri Nets, Paris-France, June 1990.

[2] E. Clarke, T. Filkorne, S. Jha, “Exploiting Symmetry In Temporal Logic Model Checking”, 5th Computer Aided Verification (CAV), June 1993.

[3] E. Clarke,O. Grumberg, D. Long,"Verification Tools for Finit-State Concurrent Systems", "A Decade of Concurrency-Reflections and Perspectives", LNCS vol 803, 1994.

[4] C. Courcoubetis, M. Vardi, P.Wolper, M. Yannakakis, "Memory Efficient Algorithms for the Verification of Temporal Properties", In proceedings of CAV’90, North Holland, DIMACS 30. 1990.

[5] M. Dam. "Fixed points of Büchi automata", In R. Shymanasundar, editor, Foundations of Software Technology and theoretical Computer Science, volume 652 of LNCS, pages 39-50, Springer-Verlag, 1992.

[6] E.A. Emerson, A. Prasad Sistla, “Symmetry and Model Checking”, In Formal Methods and System Design 9, pp 105-131, 1996.

[7] E.A. Emerson, A. Prasad Sistla, “Symmetry and Model Checking”, 5th conference on Computer Aided Verification (CAV), June 1993.

[8] E.A. Emerson, A. Parsad Sistla, "Utilizing Symmetry when Model Checking under Fairness Assumptions: An Automata-theoric Approach", 7th CAV, LNCS 939, pp. 309-324, Liège, Belgium, July 1995.

[9] E.A. Emerson, "Temporal and Modal Logic", HandBook of Theoretical Computer Science, Volume B, J. van Leeuwen (eds), 1990.

[10] E.A. Emerson and Chin-Laung Lei, "Modalities for Model Checking: Branching Time Stricks Back", In Proc of 12h Annual Symposium on Principles of Programming Languages, New-Orleans, Louisiana, January 1985.

[11] R. Gerth, D. Peled, M. Vardi, P. Wolper, "Simple On-the-fly Automatic Verification of linear Temporal Logic", Protocol Specification Testing and Verification, 1995, Warsaw, Poland.

[12] S. Haddad, JM. Ilié, B. Zouari, M. Taghelit, "Symbolic Reachability Graph and Partial Symmetries", In Proc. of the 16th ICATPN, Torino, Italy, June 1995.

[13] J-M. Ilié, K. Ajami, "Model Checking through the Symbolic Reachability Graph", in Proc of TapSoft’97 - CAAP, pp 213-224, Lille, France, Springer-Verlag, LNCS 1214, Avril 1997.

[14] K. Jensen, G. Rozenberg (eds), "High Level Petri Nets, Theory and Application", Springer-Verlag, 1991.

[15] Z. Manna, A. Pnueli. "The Temporal Logic of Reactive and Concurrent Systems: Specification", Springer-Verlag, 1992.

[16] F. Michel, P. Azéma, F. Vernadat. "Permutable Agents and Process Algebra", In Proc. of TACAS’96, Passau, Germany, 1996, Springer-Verlag, LNCS 1055.

[17] C. Norris IP and D. Dill, "Better Verification Through Symmetry", In Formal Methods in System Design, Vol 9, August 96, pp 41-76.

[18] D. Park,"Concurrency and Automata on Infinite Sequences", LNCS vol 114, 1984.

[19] K. Schmidt, "Symmetry Calculation", Workshop CSP Warschau 1995.

[20] M.Y. Vardi, "Alternating Automata and Program Verification", Computer Science Today: Recent Trends and Developments.LNCS,Vol.1000, Springer-Verlag 1995.

[21] M. Y. Vardi, "An Automata-theoretic approach to linear temporal logic (banff’94), LNCS, 1043, 1996.