Une technique Automatique de Vérification de propriétés des Systèmes Symétriques*

K.Ajami, J-M Ilié

LIP6 (**L**aboratoire d'**I**nformatique de **P**aris**6**)
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

Résumé

Dans cet article, nous proposons une technique de vérification des propriétés des systèmes symétriques à partir du graphe de marquages symboliques. Un Graphe de Marquages Symboliques (GMS) est une représentation condensée du graphe d'états construit automatiquement à partir d'une spécification en termes de réseaux colorés. La construction d'un tel graphe utilise les symétries structurelles du système pour agréger les états et les actions entre les états (les transitions) dans des classes d'équivalences en choisissant des représentant des classes pour décrire l'évolution du système. Notre technique consiste à adapter et à utiliser, dans un modèle opérationnel, l'approche formelle, de la vérification des propriétés exprimées en CTL* (computational Temporal Logic Star), proposée par Emerson & al [1]. Nous proposons une logique temporelle CPN-CTL* (computational Temporal Logic Star for Colored Petri Nets), dérivée de CTL* et adaptée syntaxiquement à nos spécifications en termes de réseaux bien formés. Nous montrons, comment la vérification des propriétés peut être réalisée directement, sur le graphe de marquages symboliques malgré sa construction à partir des représentants des classes d'équivalences des états et des transitions. En effet, dans une telle représentation, les identités des objets spécifiés par la formule n'apparaissent pas dans les marquages du graphe, seule une notion de groupe d'objets est préservée indiquant la nature des objets dans le groupe ainsi que sa cardinalité. Cette technique donne lieu à une nouvelle spécification du système permettant la vérification directe des propriétés sur le graphe de marquages symboliques. Cette vérification est équivalente à la vérification des mêmes propriétés sur le graphe d'états ordinaire (graphe d'accessibilité).

Sujet: Méthodes Formelles, Vérification.

Mots Clés: Logique Temporelle, CTL*, Symétries, Graphe de Marquages Symboliques, Réseaux de Petri bien formés.

** Version intégrale de l'article "Model Checking through Symbolic Reachability Graph" Publié dans: TapSoft'97, Lille, France, Avril 1997, Springer-Verlag, LNCS 1214.*

An automatic technique for CTL* Model Checking

K.Ajami, J-M Ilié

LIP6 (**L**aboratoire d'**I**nformatique de **P**aris**6**)
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

Abstract

In this paper, we propose an automatic technique for CTL* Model checking through the Symbolic Reachability Graph (SRG). SRG is a highly condensed representation of system state space built automatically from a specification of system in terms of Well-formed net. The building of such graph profits from the presence of object symmetries to aggregate either states or actions within symbolic representatives. Our technique consists in making the formal approach of CTL* model checking presented in [1], operational. We propose a derived temporal logic CPN-CTL* (computational Temporal Logic Star) equivalent to CTL* and built to express properties of systems specified by means of a Well-formed net. We show how to perform the model checking of such a formula directly through SRG by retrieving the behavior of the objects specified within these formulas. Effectively, SRG are built according to definitions of representatives of symmetrical object groups for which the identity of objects is not preserved, i.e. only the nature of objects and the cardinality of groups are known. This technique leads to a new specification of system, from which we can prove that model checking through a state space is equivalent to model checking through the symbolic reachability graph.

Topic: Formal Methods, Model Checking.

Keywords: Temporal Logic, CTL*, Symmetries, Symbolic Reachability Graph, Well-formed Net.

1. Introduction

Checking system correctness can be performed by the verification of CTL* formulas through a state-transition graph which models the system behavior. Such verification has to cope with combinatorial explosion problem in space and time, and several works aim at reducing the size of the graph to be built, with regards to some desired properties.

One of the most promising technique has been initiated by Emerson & al [1]. It exploits the symmetries of both the system and the 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 (with BDD) [3,5,9,10]. Further developments include operational algorithms [2,15], model checking under fairness constraints [7] and application to system bisimulation [4].

The aim of this paper is to present a technique which makes the formal approach presented in [1] operational. It consists in exploiting the theory of well-formed net and the associated symbolic reachability graph, proposed in [5][6]. Well-formed nets (WN) are colored Petri Nets which enable one to specify systems in a parametric form on the basis of object classes and related action types. WN inherits from the concision of CPN since the same structure can be used to describe the behavior of similar objects.

Symbolic reachability graphs (SRG) is a highly condensed abstraction of the state space of a system specified in terms of a well-formed net. This abstraction is performed by representing classes of equivalent states and equivalent actions. The equivalence relation between states or actions is based on structural symmetries which are directly read off from the types of objects defined in the system specification. By defining convenient types of actions for these types of objects, it can be ensured that states which are equivalent let the future behavior of the system unchanged. SRG gathers the following advantages: to be built automatically from the well-formed net specification, and, to enable efficient symbolic approach by defining canonical symbolic representatives of states and actions.

Our contribution shows how to specify a system in order to perform the model checking, directly, through the SRG. The difficulty to bypass is to retrieve the behavior of the objects specified within temporal logic formulas. The starting point of our method consists in determining, for a given formula, the groups of symmetrical objects that leave the formula invariant in order to isolate the objects specified within this formula. The isolation process is based on the intersection between the two symmetry groups of the graph and formula. This intersection allows the construction of a new group of structural symmetries which is expressed by means of the refinements of the original groups of symmetrical objects. The WN defined on the refined groups can be used to build a suitable SRG, through which the formula can be checked directly. The proposed methodology allows one to evaluate in advance (structurally) the efficiency of using the system symmetries in the model checking of a formula. Effectively, two cases appear in limits: the best one where the structural symmetries of the WN are entirely used, causing a maximal condensation of the represented state space and the worse case in which any group of symmetrical objects is reduced to a singleton, leading the new SRG to be large as well as the standard reachability graph.In our context, the formal approach presented in [1] must be adjusted since the correspondence lemma between the quotient and the ordinary structures is not respected by the original SRG.

The next sections are organized as follows: part 2 briefly recalls the technique to build SRG and highlights the major properties of SRG as well as the difficulties to perform model checking through it; part 3 presents the syntax and the semantic of the propositional logic CPN-CTL* used here to express properties of systems specified by means of well-formed nets; part 4 presents the problems of checking properties through SRG and the refinement approach (with respect to a formula); part 5 contains the algorithms used to verify a formula through the new SRG; part 6 presents the formal proof of the validity of the presented work; part 7 is our conclusion.

We assume that the reader knows the basic theories of colored Petri nets, reachability graph and temporal logic. However, some known notions which must be expressed within symbolic approach are defined again.

2. Symbolic Reachability Graph

The building of a SRG starts from the specification of a system in terms of well-formed nets (WN) [5]. Such nets are colored Petri nets but their color domains and associated functions are defined from *classes* of primitive objects and from *static subclasses* of these objects. Classes gather (ordered or unordered) objects having the same nature, while static subclasses gather objects having the same nature and behavior. Moreover, in the case of ordered objects, static subclasses are ordered so as to preserve the successor relation of objects.

Example 1:
One may define class C_1=Process=$\{p_1,p_2,p_3\}$ in order to model three ordered processes, and may split Process in two static subclasses the first is $D_{1,1}$=Interactive=$\{p_1,p_2\}$ and the second is $D_{1,2}$=Batch=$\{p_3\}$.

Like in colored Petri net, a color domain is attached to each node of the net (place or transition). In well-formed nets, color domains are defined as cartesian products of either object classes or static subclasses. Any state of the system is represented by a marking of the net. Effectively, any place is marked by a collection of colors (possibly empty) defined with respect to the place color domain.

The dining philosophers is a good example of resource sharing process, with possible deadlocks, that we can use to present a model of WN. It is used also as a case study for the verification of CTL* formulas using our method. In the standard presentation, the considered classes are ordered, however, we introduce an alternative version in which unordered classes are defined in order to bring out the reducing effects of using symmetries.

Example 2:

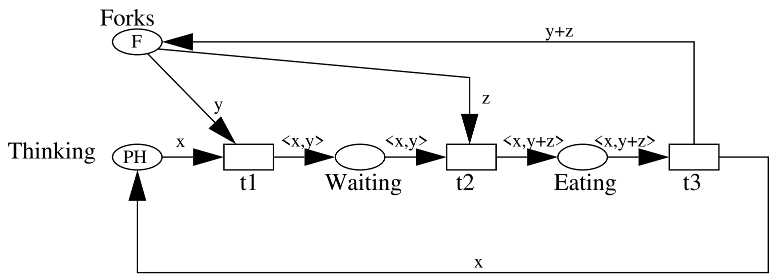


Fig. 1. The Well-formed Net of the Philosophers System with the Initial Marking

Let us consider a finite set of philosophers who spend their time thinking and eating around a circular table. Initially, any philosopher has a direct access to a set of free forks that contains as many forks as the number of philosophers. A philosopher can pick up one fork or two forks if they are free. However, he needs two forks to eat. After eating, he returns the two forks together. In any case, he does not relax the forks before eating, therefore, deadlocks appear when all the philosophers have taken one fork in the same time. From a modelling point of view, our philosophers can be in one of the following three states: "thinking while ignoring the forks", "waiting for a fork but hav-

ing another", or "eating". In terms of well-formed net, three places are used to represent these states and a fourth place must be added to model the unused forks. The color domains attached to these places are defined from the two following basic classes: philosophers PH and forks F. By noting C(r) the color domain of node r, we have:
C(Thinking)=PH, C(Waiting)= $PH \times F$ *, C(Eating)=* $PH \times F$ *, C(Forks)=F.*
For instance, the marking which models three philosophers and three forks such that the first thinks, the second waits for a fork but detains the fork number one, and the third eats with forks number two and three, is the following:
m(Thinking) = Ph_1*; m(Forks) = 0; m(Waiting)=* $<Ph_2,f_1>$*;*
m(Eating)= $<Ph_3,f_2+f_3> = <Ph_3,f_2> + <Ph_3,f_3>$*;*

The set of collections of colors representing the possible markings of place p is noted Bag(C(p)) according to the color domain of place p, and any marking is expressed as a linear combination of tuples wherein each component relates to a simple color from a class. In a color domain, a class may appear one or many times in the domain:
$M(p_i) = \left(\sum_k \alpha_k m_k\right) \in Bag(C(p_i)), m_k = \prod_i \prod_j c_i^j$, the indexes i,j of a simple color c_i^j denote respectively the class number and the occurrence number of color c_i^j in m_k.

Basically, the construction of the symbolic reachability graph is defined from the notion of symbolic marking.

2.1. Symbolic Marking

Roughly speaking, a symbolic marking is a representative object of an equivalence class of markings for which the equivalence relation is deduced from a set of "*admissible symmetries*". Such symmetries operate on the object classes of the studied WN. They preserve the static subclasses and the successor relation on ordered classes when defined.

Let $\mathcal{N}$ be a WN and CD={C(r)|$r \in P \cup T$ } be the set of the color domains attached to either places or transitions in $\mathcal{N}$. Let C={C_i: C_i class of colors} be the set of all the classes of $\mathcal{N}$. Let us consider any color domain C(r) from CD.

> **Definition 2.1.1: Symmetry and Group of Symmetries**
> A symmetry S on C(r) is a permutation on the color domain C(r). A set of symmetries, ζ, on $\mathcal{N}$ is defined as the family of symmetries S which operate on the elements of CD. (ζ, o) forms the group of symmetries of $\mathcal{N}$. The set of *admissible symmetries* of $\mathcal{N}$, AS($\mathcal{N}$), is a subset of ζ that satisfies the two conditions: (1) (AS($\mathcal{N}$),o) is a subgroup of (ζ,o), (2) Let C_i be an object class and $D_{i,q}$ one of its static subclasses, then we have: $\forall s \in AS(\mathcal{N}), \forall c \in D_{i,q}$, $s(c) \in D_{i,q}$. The set of static subclasses of the net $\mathcal{N}$ can be formally defined as the equivalence classes that result from the quotient structure $\hat{C} = C/AS(\mathcal{N})$.

It must be noted that, admissible symmetries on ordered classes are restricted to rotations in order to preserve the order of colors. In consequence, admissible symmetries of ordered classes, composed of many static subclasses, are restricted to identity.

In WN, due to the restricted (but well chosen) operators defined on object classes, it has been proved that symmetrical colors in a given marking leads the system to have symmetrical behaviors (They lead to define equivalent markings and firing sequences). Therefore for a given state, symmetrical colors can be aggregated in equivalence classes in such a way that only representatives are used. Such representatives express quantity of objects from their static subclass while forgetting identities of objects Hence, markings having the same representation can be represented only once. They correspond to the notion of dynamic subclasses, from which symbolic marking is expressed.

Let $\mathcal{M}$ be the set of all the markings of $\mathcal{N}$, the symbolic marking is defined as follows:

Definition 2.1.2: Dynamic Subclasses and Symbolic Markings
A dynamic subclass of C_i represents a subset of colors belonging to a static subclass of C_i. It is featured by the name of the concerned static subclass and by the cardinality of the subset. We note Z_i^j the j^{th} dynamic subclass of C_i.

According to AS($\mathcal{N}$), $\hat{\mathcal{M}} = \mathcal{M}/AS(\mathcal{N})$ is the set of equivalence classes of markings. Each class of $\hat{\mathcal{M}}$ is represented by a canonical representative called *Symbolic Marking*. It is expressed in terms of vector of marked places where colors are symbolically represented by dynamic subclasses. Its useful notation is $\hat{m}$.

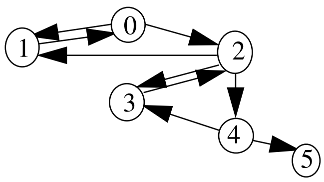
We define a morphism $\alpha_{\hat{m}}$ on Bag(C) (C is the set of color classes) where $\alpha_{\hat{m}}(c)$ denotes the representative of a color c in $\hat{m}$. Mainly, this morphism is used, by extension, to compare the representatives of symmetrical markings. Effectively, two markings m, m' such that $s(m) = m'$ for a symmetry $s \in (AS(\mathcal{N}), o)$ have the same representative. We have: $\alpha_{\hat{m}}(m) = s(\alpha_{\hat{m}}(m)) = \alpha_{\hat{m}}(s(m)) = \alpha_{\hat{m}}(m') = \hat{m}$.

Example 3:
The marking of Example 2 can be expressed symbolically by considering that one philosopher thinks, one waits for a second fork and one eats. The corresponding symbolic marking is deduced by introducing convenient dynamic subclasses defined on the static subclasses (the class of philosophers presents a static subclass as well as the class of forks). The class of philosophers is divided in three dynamic subclasses, the cardinality of each is one. The class of forks must be split in two dynamic subclasses, the first is associated with the philosopher who eats and represents two forks, the second is associated with the philosopher who waits and represents one fork:
$\hat{m}$ *(Thinking)=* Z_1^1 *;* $\hat{m}$ *(Forks)= 0;* $\hat{m}$ *(Waiting) =* $\langle Z_1^2, Z_2^1 \rangle$ *where* $|Z_2^1| = 1$ *;* $\hat{m}$ *(Eating)=* $\langle Z_1^3, Z_2^2 \rangle$ *where* $|Z_2^2| = 2$ *and* $|Z_1^1| + |Z_1^2| + |Z_1^3| = 3$
In fact, $\hat{m}$ *represents eighteen markings obtained by operating the admissible permutations on the class PH and the associated class F when they are represented in* $\hat{m}$.

2.2. Symbolic Reachability Graph Construction

In [5], a symbolic firing rule is introduced in order to compute a new symbolic marking directly from a current one. The classical notion of instance of transition is replaced by the notion of symbolic instance which corresponds to a splitting of the dynamic subclasses of the current marking in order to isolate quantities of colors that can be used for the firing. The definitions of symbolic marking and symbolic firing rule allow us to build symbolic reachability graphs. In this graph, the nodes are the symbolic markings expressed in a canonical form.

Example 4:
Figure 2 represents the SRG for three philosophers. It contains 6 nodes and 9 arcs while the corresponding reachability graph contains 88 nodes and 207 arcs. It is worth noting that important properties like deadlock can be directly verified (e.g. state 5 has no successor):



The meaning of the markings is:
(0) Three philosophers think.
(1) Two philosophers think and one philosopher eats.
(2) Two philosophers think, one philosopher waits.
(3) One philosopher thinks, one waits and one eats.
(4) One philosopher thinks, two philosophers wait.
(5) The three philosophers wait.

Fig. 2. SRG for three Philosophers

3. The Temporal Logic CPN-CTL*

In order to perform our model checking, we use the Propositional form of the Computational Tree Logic star (CTL*) proposed in [1], [7] and [8] that respect a specification of the system in terms of a colored petri net. Classically, the linear temporal operators are introduced as follows: F (sometimes), G (always), X (next time) and U (strong until) [11]. Moreover, path quantifiers are represented either by symbol A for all full paths or E for some full paths. Like in [7] too, two types of formulas are considered: state formulas (which holds in a specific state) and path formulas (which holds along a specific path). Let *AP* be the set of atomic propositions. A state formula is either [7]:
- a if $a \in AP$;
- if f and g are state formulas, then $\neg f$ and $f \vee g$ are state formulas.
- if f is a path formula, then E(f) is a state formula.
A path formula is either:
- a state formula;
- if f and g are path formulas, then $\neg f$, $f \vee g$, Xf and fUg are path formulas.
By definition, CTL* is the set of state formulas generated by the above rules.
It must be noted that any CTL* formula can be transformed to be temporally expressed by the two boolean operators, the negation and the disjunction and the temporal operators E, X, U presented previously using the general transformations [8]:
$(1) f \wedge g \equiv \neg(\neg f \vee \neg g)$, $f \rightarrow g \equiv \neg f \vee g$; $(2) A(f) \equiv \neg E(\neg f)$; $(3) Gf \equiv \neg F \neg f$; $(4) Ff \equiv True U f$.

In the current context of Petri nets, formulas must be expressed in terms of markings. We call such expression CPN-CTL* and present its syntax in the next section.

3.1. CPN-CTL* Syntax

In order to specify properties of WN, formulas must be expressed in terms of colors of classes and quantities of colors. Moreover, they must refer to places since colors in places represent the system variables assigned to specific values. Hence, state formulas express that tokens (i.e. markings) exist in places (i.e. mark the places). For more clarity, quantities are assumed to be always equal to one in this paper. In fact, considering distinct values do not represent any theoretical difficulties.

Depending on the fact that a color domain of a place p can be built on an object or a cartesian product of object classes, we now introduce the form of our formulas on a simple color domain: (i) a(c) where $a(c) = c \bullet p$ tests if a color, from a color domain C_i, mark place p; (ii) $\exists x, f(x)$ where f is a CPN-CTL* formula tests if there is a color x from a color domain C_i over which f is verified. Similarly, the expression $\forall x, f(x)$ needs to test the verification of f for all the colors of C_i. Observe that the form used for complex color domain is a straightforward extension since such a domain is directly composed of simple color domains.

For more clarity, we use indexed CTL*. Consequently, universal and existential quantifiers are now re-expressed in terms of disjunction or conjunction operators. The form of the previous formulas can be re-expressed as: $V_i\, f(c_i)$ or $\Lambda_i\, f(c_i)$ where V and Λ are respectively the disjunction and the conjunction operators. It must be noted that for a color b from color domain C_i we have the following equivalence: $f(b) \Leftrightarrow \exists x, f(x) \Leftrightarrow V_i f(c_i)$. Moreover, the conjunction operator in a state formula corresponds to the (+) operator used in marking expressions. So we define: $\left(\sum_k c_k\right) \bullet p \Leftrightarrow [c_1 \bullet p \wedge \ldots \wedge c_i \bullet p \wedge \ldots] \Leftrightarrow f = \Lambda_k f(c_k)$ where $f(c_k) = c_k \bullet p$. Further, this allows us to simplify the presentation by comparing atomic propositions and marking of the form $c \bullet p$.

3.2. CPN-CTL* Semantics

We define the semantics of CPN-CTL* with respect to the structure of a reachability graph $R = <\mathcal{M}, \mathcal{R}, \mathcal{E}>$ where $\mathcal{M}$ is the set of the system markings, $\mathcal{R} = \mathcal{M} \times \mathcal{M}$ is the set of edges between markings and $\mathcal{E}$ is the evaluation function defined on $\mathcal{M}$ and used to verify the correctness of an atomic formula. A path in R is an infinite sequence of markings $\pi = m_0 m_1 \ldots$ such that for every i, $(i \geq 0)$ we have $(m_i, m_{i+1}) \in \mathcal{R}$. π_i denotes the suffix of π starting at m_i. We use the standard notation to indicate that a state formula f holds in a structure: $R, m \models f$ means that f holds at the marking m, similarly, $R, \pi \models f$ means that the path formula f holds along path π.

To present the semantic of CPN-CTL* formula through a R structure, we start by defining a projection function that can be used to express $\mathcal{E}$, the evaluation function. With respect to a marking m of the net, let us consider an atomic proposition a defined as a marking of place p and have one of the two forms expressed previously. We define the subdomain SD_a of a_p as $SD_a: Bag(C(p)) \rightarrow Bag(C(p))$ such that:

$SD_a\left(C_{i_1} \times \ldots \times C_{i_n}\right) = SD_{a,1}(C_{i_1}) \times \ldots \times SD_{a,n}(C_{i_n})$ where $SD_{a,i}(C_{i_j}) = C_{i_j}$ if the i^{th} component of the tuple expressed in a_p belongs to C_{i_j} and $SD_{a,i}(C_{i_j}) = \emptyset$ if not.

We note $Prj[SD_a](M)$ the projection of marking M with respect to the subdomain SD_a, hence: $a \in \mathcal{E}(m) \Leftrightarrow a = Prj[SD_a](m)$.

The next rules of presents the semantics of CPN-CTL*. We assume that f_1, f_2 are two state formulas, g_1, g_2 are two path formulas. The relation $\models$ is defined inductively as follows [8]:

(1) $R,m \models a$ iff $a \in \mathcal{E}(m)$.

(2) $R,m \models \neg f_1$ iff Not($R,m \models f_1$).

(3) $R,m \models f_1 \vee f_2$ iff $R,m \models f_1$ or $R,m \models f_2$.

(4) $R,m \models E(g_1)$ iff $\exists\pi$ starting from m such that $R,\pi \models g_1$.

(5) $R,\pi \models f_1$ iff m is the first marking of π and $R,m \models f_1$.

(6) $R,\pi \models \neg g_1$ iff Not($R,m \models g_1$).

(7) $R,\pi \models g_1 \vee g_2$ iff $R,\pi \models g_1$ or $R,\pi \models g_2$.

(8) $R,\pi \models X(g_1)$ iff $R,\pi_1 \models g_1$.

(9) $R,\pi \models g_1 U g_2$ iff $\exists k \geq 0$ such that: $R,\pi_k \models g_2$ and $\forall i$ such that $k > i \geq 0$ we have $R,\pi_i \models g_1$.

In the next section we present the difficulties that we must bypass in order to perform model checking directly through the symbolic reachability graph, then we introduce a way to built a highly condensed but convenient SRG.

4. Symbolic Reachability graph built with Respect to a Formula

Using CPN-CTL* formulas, two kinds of properties can be checked, using a reachability graph which presents the state space of a system specified in terms of well-formed net. State properties and path properties. State properties presents properties verified by a set of colors or a tuple of colors in a given marking of the graph. In consequence, they are presented as a boolean combination of atomic propositions. Path properties presents properties that are verified, through a path of the graph, either by a set of colors or tuple of colors. They are represented in terms of both boolean and temporal combination of atomic propositions.

In this section we present the properties which can be directly verified through the symbolic reachability graph, as well as properties which need additional informations. Finally, we present the unfolding process that exploit symmetries in order to add sufficient informations for performing general model checking while saving a highly condensed representation of the state space.

4.1. Model Checking Problem

Due to color representations in the symbolic approach, two major difficulties appear when performing model checking through SRG. The first consists in checking properties based on color identities since identities of colors are not preserved. The second is due to the fact that path properties expressed symbolically (i.e. expressed for an arbitrary quantities of symmetrical colors) cannot be checked. Effectively, it can be proved that a symbolic path between two symbolic markings may represent not only real paths but also wrong paths. Here again, path properties which require the verification of color dependencies between some markings cannot be checked even if they are expressed symbolically. Anyhow, SRG allows direct model checking of state properties expressed symbolically. Effectively, the quantity of colors represented by a dynamic subclass in a marking represents a set of colors that have the same behavior. In consequence, several properties which are not color dependent can be checked directly like the absence of deadlock, the existence of home space (resp. unavoidable home space) or the existence of an infinite path.

Example 5:
In the example of philosophers, we mentioned that deadlock are directly detected by searching the states that do not have outcoming edges, we can detect deadlocks, like state 5 in the graph presented in figure 2. Contrarily, properties like fairness cannot be proved through the SRG. For example, let us consider the formula $f = \Lambda_{Ph \in PH} \mathrm{A}G[Ph \bullet Thinking \Rightarrow F(Ph \bullet \mathrm{Eating})]$ *, which means that, for all the paths of the graph a philosopher who is thinking will eat. Further, the verification of such a formula needs to span the graph depicted in figure 2 in such a way that f is satisfied through all of its paths. Unfortunately, this verification is impossible since identities are not preserved in the markings of the graph. For example, let us consider the infinite path of the symbolic reachability graph, "2,4,3,2,4,3...". Through this path, it is not possible to determine that the philosopher who thinks in symbolic marking 2 is the same who eats in symbolic marking 3.*

Hopefully, as a partial solution for the model checking through symbolic paths, the next section will show that checking properties on groups can be reduced to check them on representatives of the groups.

4.2. Studying the Behaviors of Symmetrical Colors

In order to perform model checking of properties expressed for a color or a subset of colors from the same static subclass, we prove that a property is verified for a color if and only if it is verified for all the colors of the same static subclass under the assumption that all those colors have symmetrical initial states (which is our assumption here in this paper). In this case, the verification of a symbolic path property expressed for all the objects can be substituted by the verification of this property for one color or a subset of colors from the same static subclass. This leads to isolate the chosen colors, only, in order to trace their behaviors. Such a partial process of isolation corresponds to a partial unfolding of the SRG.

It is worth noting that there is no need to isolate colors for the verification of symbolic state properties since the verification of the reachability is performed through the SRG directly [5].

Let us consider an atomic proposition g expressed as a marking of a WN, $\mathcal{N}$, two markings m_1, m_2 of $\mathcal{N}$ and s a symmetry from AS($\mathcal{N}$). We prove that symmetrical markings yields symmetrical projections with respect to an atomic proposition:

Lemma 1: $\mathrm{Prj}[\mathrm{SD}_g](m_1) = m_2 \Leftrightarrow \mathrm{Prj}[\mathrm{SD}_g](s(m_1)) = s(m_2), \forall s \in AS(\mathcal{N})$

Proof: This proof considers first a marking build with a unique color of a simple color domain, then, it is extended to complex color domain. Firstly, let us consider any place p of the net and let us assume that $m_1(p)=c$ where $c \in C$ is a simple color and let us suppose that H: $Prj[SD_g](m_1(p)) = m_2(p)$ holds. We have either:

(1) $\forall c \in C, c \notin g$ i.e $m_2(p)=0$. This is equivalent to $s(c) \notin g, \forall s \in AS(\mathcal{N})$ car s(c) is an object of C. Since $s(m_1(p))=s(c)$, then we have:

$Prj[SD_g](s(m_1(p))) = s(m_2(p)) = 0$.

(2) $\exists c \in C, c \in g_p$. From H and $Prj[SD_g](m_1(p)) = m_2(p)$ we deduce that $m_2(p)=c$. This is equivalent to $s(m_1(p)) = s(m_2(p)) = s(c)$. Since s(c) is also a color then we have: $Prj[SD_g](s(m_1(p))) = s(c) = s(m_2(p))$.

In case of complex color domains the former proof can be extended to deal in a similar way with each component of the tuple.

From the previous lemma, we can conclude that state properties are preserved for symmetrical colors.

Proposition 1:

$\forall s \in AS(\mathcal{N})$, $(R,m \models g_c) \Leftrightarrow (R, s(m) \models s(g_c))$ where R is the state space and $s(g_c)=g_{s(c)}$, g_c is an atomic proposition.

Proof: Starting from the semantic rules of section 3.2:

$R,m \models g_c \Leftrightarrow (g_c \in \mathcal{E}(m)) \Leftrightarrow (Prj\left[SD_{g_c}\right](m) = g_c)$.

From lemma 1: $Prj\left[SD_{g_c}\right](m) = g_c \Leftrightarrow Prj\left[SD_{g_c}\right](s(m)) = s(g_c)$, which is equivalent to $s(g_c) \in \mathcal{E}(s(m))$. So we have $R,s(m) \models s(g_c)$ where $s(g_c)=g_{s(c)}$.

Proposition 1 makes it easy to prove that path properties are preserved for symmetrical colors. Path properties are expressed in terms of CPN-CTL* path formulas. We start our proof for a restricted set of formula EXf and E(fUg) (f, g are atomic propositions) from which the generalization to CPN-CTL* formulas is made using the rules of section 3.2:

Proposition 2:

(1) $R,\pi \models \Lambda_{c \in D}, EXg_c \Leftrightarrow R, \pi \models EXg_a$ where object a, is chosen arbitrary from D_i.

(2) $R,\pi \models \Lambda_{c \in D_i, c' \in D_j}, E(f_c U g_{c'}) \Leftrightarrow R, \pi \models E(f_a U g_b)$.

where f_c, g_c are atomic propositions, D_i, D_j are static subclasses or cartesian products of static subclasses and a,b are respectively chosen arbitrary from D_i and D_j.

Proof:

(1) the $\Rightarrow$ direction is simply proved because $\Lambda_{c \in D}, EXg_c$ means that EXg_{a1} holds and... EXg_{an} holds where $a_1,...a_n$ are the objects of D_i. For the $\Leftarrow$ direction, EXg_a holds by assumption, i.e. $R,\pi \models EXg_a$ where $\pi=m_1m_2...m_n$. From the semantic rules of 3.2 we have: $R,m_2 \models g_a$ so, $Prj\left[SD_{g_a}\right](m_2(p)) = g_a$. From proposition 1, this is equivalent to $Prj\left[SD_{g_a}\right](s(m_2(p))) = s(g_a), \forall s \in AS(\mathcal{N})$ so, $\forall s \in AS(\mathcal{N}), s(g_a) = g_{s(a)}$ holds in $s(m_2(p))$. In consequence, $s(EXg_a)=EX(g_{s(a)})$ holds in $s(\pi)$ for each s and consequently for each element of the static subclass.

(2) Like in the former proof, the $\Rightarrow$ direction is straightforward, since $\Lambda_{c \in D_i, c' \in D_j}, E(f_c U g_{c'})$ holds means that $E(f_{a1} U g_{b1})$ holds and.... where a_i, b_j are objects respectively chosen from D_i and Dj. For the $\Leftarrow$ direction , $E(f_a U g_b)$ holds, by assumption and, from the semantic rules presented in section 3.2, we deduce: $\exists(\pi = m_1, \ldots, m_k, \ldots)$, $\mathrm{Prj}\left[SD_{f_a}\right](m_l(p)) = f_a, \forall l \leq k$, $\mathrm{Prj}\left[SD_{g_b}\right](m_t(q)) = g_b$, $\forall t \geq k$. Like in part (1) of this proposition and for each marking of the path we use proposition 2 to prove that: $\forall s \in AS(\mathcal{N}), s(f_a) = f_{s(a)}$ holds in $s(m_l(p)), \forall l \leq k$ and $s(g_b) = g_{s(b)}$ holds in $s(m_t(q)), \forall t \geq k$. This is equivalent to: $s(E(f_a U g_b))=E(f_{s(a)} U g_{s(b)})$ holds in $s(\pi)$, for each s and consequently for each object of the static subclass.

Finally, the former proposition can be generalized to any path formula of CPN-CTL* by using the semantic rules of the temporal logic mentioned in section 3.2.

The next section shows how to refine the WN description so as to define the subgroup of the admissible symmetries that leaves the formula invariant. The propositions of the current section are used to drive the net transformation.

4.3. Subgroup of Admissible Symmetries that Respect a Formula

Roughly speaking, the conditions that allow the building of a SRG through which a formula can be checked are the three followings: detect the colors that appear in the formula; find symmetries between that colors in order to form the group of symmetries that leave the formula invariant; and save the admissible symmetries of colors that do not appear in the formula as well as those of colors which appear in the formula (such admissible symmetries correspond to those that leave the formula invariant). More practically, the former three points lead us to succeed the two following stages:
(1) Determine the automorphism group of the formula that reflects the symmetries expressed by the isolated colors;
(2) Consider only a subgroup of the group of admissible symmetries that leave the formula invariant. SRG will be built on the basis of such subgroup.
In WN, this subgroup is determined statically since admissible symmetries can be deduced, directly, from the description of static subclasses. It corresponds to a refinement of static subclasses in order to isolate the colors of the formula within the considered net specification.

4.3.1. The Unfolding Formal Approach

Let us assume the existence of $\mathcal{N}$, a given WN. The determination of a subgroup of $(AS(\mathcal{N}),o)$ that leaves formula invariant requires to express the structural symmetries reflected in the formula. Such symmetries can be formally defined by the notion of automorphism group of a formula.

Definition 4.3.1.1: Automorphism Group of a Formula
The automorphism group of f, Aut(f), is the group of permutations of colors that leave f invariant.

The former definition of automorphism group means that: $\forall s \in \mathrm{Aut}(f), s(f) = f$, but it does not always ensure that neither the splitting of colors in static subclasses nor the restrictions imposed on symmetries for ordered classes are respected. Therefore we must consider a subgroup of $AS(\mathcal{N}) \cap \mathrm{Aut}(f)$ which expresses the admissible symmetries that leave f invariant [1]. Of course, the largest subgroup, $AS(\mathcal{N}_f) = AS(\mathcal{N}) \cap \mathrm{Aut}(f)$, is desirable for maximal compression of state space. $AS(\mathcal{N}_f)$ is a restriction of the admissible structural symmetries enabled in $\mathcal{N}$, therefore, it is always possible to form a new well-formed net, $\mathcal{N}_f$ according to this subgroup.

The static subclasses of $\mathcal{N}_f$ are obtained by refinement of static subclasses of $\mathcal{N}$. This leads to a new definition of admissible symmetries.

Definition 4.3.1.2: Admissible Symmetries with Respect to a Formula
The group of admissible symmetries with respect to f, $(AS(\mathcal{N}_f),o)$, is a subgroup of (ζ, o) that satisfies one of the two following equivalent conditions:

1. $(AS(\mathcal{N}_f),o)$ is a subgroup of (ζ, o) such that: $AS(\mathcal{N}_f) = AS(\mathcal{N}) \cap \mathrm{Aut}(f)$
2. Let C_i be an object class and $D_{i,q}$ one of its static subclass, we have: $\forall s \in AS(\mathcal{N}_f), \forall c \in D_{i,q}, (s(c) \in D_{i,q} \wedge s(f) = f)$

One may note that no refinement is needed for a given class C_i when Aut(f)=Sym(C_i). In consequence: if $(\mathrm{Aut}(f) = \bigcup_i \mathrm{Sym}(C_i)) \Leftrightarrow (AS(\mathcal{N}_f) = AS(\mathcal{N}))$.

Let us consider R as the structure of the reachability graph deduced from $\mathcal{N}$. The SRG, built from the specification of $\mathcal{N}_f$, enables model checking for the formula f. Such SRG denoted $SRG_{\mathcal{N}, f}$ represents the quotient structure which saves the largest symmetries for $\mathcal{N}$ that leave f invariant.

Property 4.3.1.3: Correspondence property
Let $\hat{m}$ and $\hat{\pi}$ be respectively a marking and a path of $SR_{\mathcal{N}, f}$, we have the following two properties for symbolic markings and symbolic paths:

(1) $\forall s \in AS(\mathcal{N}_f), \forall m, \alpha_{\hat{m}}(s(m)) = \hat{m}$, $SRG_{\mathcal{N}, f}, \hat{m} \models f \Leftrightarrow R, m \models f$;

(2) $\forall s \in AS(\mathcal{N}_f), \forall \pi, \alpha_{\hat{\pi}}(s(\pi)) = \hat{\pi}$, $SRG_{\mathcal{N}, f}, \hat{\pi} \models f \Leftrightarrow R, \pi \models f$.

This important property enables one to perform the model checking of f, directly through $SRG_{\mathcal{N}, f}$. The proof of the former property is included in the proof of model checking equivalence presented in Section 6. The computation of $AS(\mathcal{N}_f)$ consists, mainly, in determining Aut(f) since $AS(\mathcal{N})$ is given initially by $\mathcal{N}$. The determination of Aut(f) is presented in the next section.

4.3.2. Determination of a Formula Automorphisms Group

In [1], some rules which concern the determination of automorphism group, are presented in the context of CTL* formulas model checking through a state transition graph. Many difficulties appear when one performs model checking through SRG due to the particularity of color representations in the symbolic approach (see section 4.1). In consequence, the rules must be adapted not only to respect the conditions presented in the introduction of section 3.2 but also to cope with those difficulties. Such an adaptation allows the building of a SRG through which a formula can be checked directly.

Let θ be one of the two boolean operators Λ or V. In the following, we consider a formula f built on a subset of colors *K*. Such a subset may be either a class, a static subclass or a subset of a static subclass. We define Sym(*K*) as the set of all the permutations on *K*. We now define, τ, the transformation function of static subclasses and its correlated function I used to isolate an arbitrary object that isolate an arbitrary object or set of objects by splitting the concerned static subclass:

$\tau: CD \rightarrow CD^2$ and for a cartesian product of static subclasses $D_1 \times \ldots \times D_n$ we have: $\tau(D_1 \times \ldots \times D_n) = \langle I(D_1 \times \ldots \times D_n), D'_1, \ldots, D'_n \rangle = \langle I_1(D_1), \ldots, I_n(D_n), D'_1, \ldots, D'_n \rangle$

where: $D'_i = D_i \backslash \{I_i(D_i)\}$ and $I_i(D_i)$ is the i^{th} component of the isolation function applied on the i^{th} component of the cartesian product such that:

$I_i: C \rightarrow C$, where C is a class of colors, such that for $D_i \subseteq C, I_i(D_i) = \{c_i\} \subseteq D_i$.

Moreover, the isolation function defined over a tuple must take into account additive integrity constraint. This can be formalized by defining a general isolation function as follows: I: $C^n \rightarrow C^n$, I=$<I_1,...,I_n>$ such that:

(1) $(I_i \neq I_j) \Leftrightarrow (D_i \neq D_j) \vee \{ (D_i = D_j) \wedge (I_j(D_j) = I_i(D'_i)) \}$;

(2) $(I_i = I_j) \Leftrightarrow \{ I_i(D_i) = I_j(D_j) = \{c_i\} \wedge (D'_i = D'_j) \}$.

Using such definitions, we can present the rules that determine the automorphisms group of the formula and use isolation and transformation functions to calculate this group.

Rules 4.3.2.1: Generic rules to determine Aut(f)

(1) If f is trivial (f or $\neg$f is a validity) then $\mathrm{Aut(f)} = \bigcup_i \mathrm{Sym}(C_i)$ for all the classes of colors C_i.

(2) If f= $g_{b \in D}$ built for a specific color, b, or a tuple of colors b=$<b_1,.,b_m>$ from a static subclass or a cartesian product of static subclasses D then Aut(f)=Sym(D\{b}) where $\tau(D) = \langle I(D), D' \rangle$ and I(D)=b.

(3) If $f = \theta_{c_{i_1} \in D_1} \ldots \theta_{c_{i_n} \in D_n} g_{\langle c_{i_1} \ldots c_{i_n} \rangle}$ built for a set of tuples of colors from a color domain $D_1 \times \ldots \times D_n$ then Aut(f)=Aut($g_{I(D_1 \times \ldots \times D_n)}$) where $\tau(D_1 \times \ldots \times D_n) = \langle I(D_1 \times \ldots \times D_n), D'_1, \ldots, D'_n \rangle$

(4) If $f = \theta_{c \in D} g_c$ where D is a static subclass then Aut(f)=Sym(D) if f is a state formula, if not, Aut(f)=Aut($g_{I(D)}$) where $\tau(D) = \langle I(D), D' \rangle$.

(5) If f is a temporal formula of the form f = Xg where g has one of the forms presented by the current rules then Aut(f)=Aut(g).

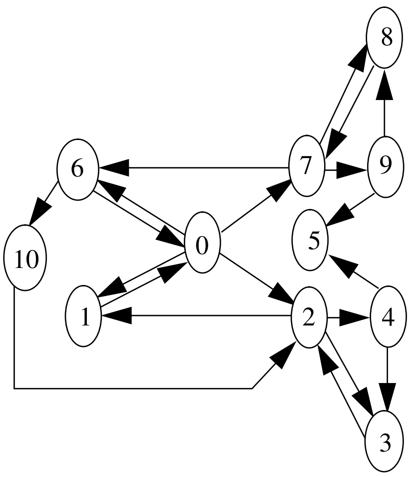
(6) If f is a temporal formula of the form f = g U h where g, h has one of the forms presented by the current rules then Aut(f)=Aut(g) $\cap$ Aut(h) .

Example 6:

Again, let us consider the following fairness property [2]*:*

$f = \Lambda_{Ph \in PH} AG[Ph \bullet Thinking \rightarrow F(Ph \bullet \text{Eating})]$, *meaning that "for each path, it is always possible to eat for a philosopher who thinks". Initially, PH has only one static subclass, PH itself. Formula f is a path formula which corresponds to rule 4-b of rules 4.3.2.1, then the automorphism group of f is: Aut(f)=Sym($f_{\tau(PH)}$). We suppose that $\tau(PH)$=<{Ph_1}, {Ph_2, Ph_3}).*

We have AS($\mathcal{N}$)=Sym(PH), in consequence, the new group of admissible symmetries is AS($\mathcal{N}_f$)=Sym(PH\{Ph_1}). In order to check the formula, PH must be partitioned in two static subclasses: FirstPhilosopher={Ph_1} and OtherPhilosophers={Ph_2, Ph_3}. The new SRG built on such admissible symmetries is presented in Figure 3. It worth noting that, the advantage of such isolation process is that static subclasses which do not appear in the formula are saved, like the class Forks in our example. The presented graph remains highly condensed (11 nodes and 20 arcs instead of 88 nodes and 207 arcs).



The meaning of the markings is:
(0) Three philosophers think.
(1) The one of FirstPhilosopher thinks. One of OtherPhilosophers thinks and one eats.
(2) The one of FirstPhilosopher thinks One of OtherPhilosophers thinks and one waits.
(3) The one of FirstPhilosopher thinks. One of OtherPhilosophers waits and one eats.
(4) The one of FirstPhilosopher thinks. Each philosopher of OtherPhilosophers waits.
(5) The three philosophers wait.
(6) The one of FirstPhilosopher eats The two of OtherPhilosophers Think.
(7) The one of FirstPhilosopher waits. The two of Otherphilosophers Think.
(8) The one of FirstPhilosopher waits. One of OtherPhilosophers thinks and one eats.
(9) The one of FirstPhilosopher waits. One of OtherPhilosophers waits and one thinks.
(10) The one of the FirstPhilosopher eats. One of the OtherPhilosophers Thinks, One waits.

Fig. 3. *SRG built with respect to formula f for three philosopher*

In the next section we present the model checking process of a formula f through the symbolic reachability graph $SRG_{\mathcal{N},f}$ built with respect to f.

5. Model Checking through SRG with respect to a formula

The verification of formula f through $SRG_{\mathcal{N},f}$ is explained first for an atomic proposition formed by simple colors, then we extend it to the case of atomic propositions expressed with tuples. Finally, we consider general CPN-CTL* formulas. Let us consider first a formula f built on a class of colors C or a static subclass D of C. The verification of either conjunctive or disjunctive forms is processed according to the refinement method imposed by AS($\mathcal{N}_f$). The atomic proposition $f = V_{c \in D}(c \bullet p)$ holds in a symbolic marking of p if at least one color of D marks p. It must be noted that atomic propositions

$f = c \bullet p$ form a particular case of the disjunctive form since colors specified in such an atomic proposition are isolated in static subclasses. Similarly, formula $f = \Lambda_{c \in D}(c \bullet p)$ holds, with respect to a symbolic marking, if all the colors of D mark p. In both cases, the verification process must take into account that place color domains of a WN can be complex (i.e. cartesian product of classes). We consider first the verification of atomic propositions defined on a simple and a complex domain, then, we generalize the verification to basic path formula expressed in proposition 6. Finally, we generalize the verification to any formula CPN-CTL* by the application of the semantics rules of the temporal logic presented in section 3.2.

Proposition 3: Disjunctive Atomic Proposition on a Simple Color Domain

$SRG_{\mathcal{N},f}, \hat{m} \models f = V_{c \in D}(c \bullet p)$ iff $\exists Z \subseteq D$ such that $Z \in Prj[SD_f](\hat{m}(p))$.

Proof: the $\Rightarrow$ direction is proved by definition of symbolic markings. Effectively, since formula f holds in $\hat{m}$, we can be sure that there is a dynamic subclass representing some colors of D in $\hat{m}$ (p). Moreover, since projection is achieved according to formula f, that dynamic subclass is present in the symbolic marking $Prj[SD_f](\hat{m}(p))$. the $\Leftarrow$ direction is proved since the presence of a dynamic subclass of D in $Prj[SD_f](\hat{m}(p))$, means that a quantity of colors of D exists in $\hat{m}$ with respect to p.

Proposition 4: Conjunctive Atomic Proposition on a Simple Color Domain

$SRG_{\mathcal{N},f}, \hat{m} \models f = \Lambda_{c \in D}(c \bullet p)$ iff the two following conditions hold:

(1) $\exists Z, \left(Z = \{Z^j \subseteq D \middle| Z^j \in Prj[SD_f](\hat{m}(p))\}\right)$. (2) $\forall Z^j \in Z$ we have: $\sum_j |Z^j| = |D|$.

Proof: The proof is similar to the one of proposition 3 with the exception that condition (2) must be taken into account. For the $\Rightarrow$ direction, in order to prove the second condition, we must consider that all the colors of D are in $\hat{m}$. In this case, the cardinality of the union of dynamic subclasses of D, in $\hat{m}$, is equal to the cardinality of D. Moreover, since projection is made according to formula f, that dynamic subclass is present in $Prj[SD_f](\hat{m}(p))$. the $\Leftarrow$ direction uses the same reasoning for the second condition.

Let us consider now atomic proposition for tuples, propositions 3 and 4 can be simply extended as follows:

Proposition 5: Extension to Atomic Propositions on a Complex Color Domain

$SRG_{\mathcal{N},f}, \hat{m} \models \left[f = \theta_{c_1 \in D_1} \ldots \theta_{c_n \in D_n}(\langle c_1, \ldots, c_n \rangle \bullet p)\right]$ iff the two conditions hold:

(1) for each component $c_q \in D_q, \exists Z_q, \left(Z_q = \{Z_q^j \subseteq D_q \middle| Z_q^j \in Prj[SD_f](\hat{m}(p))\}\right)$.

(2) for each q where $\theta_q = \Lambda_q$, we have: $\sum_j |Z_q^j| = |D_q|$.

Proof: The proof is directly derived from those of 3 and 4.

Finally, we present our verification rules in the case of model checking of conjunctive and disjunctive forms of path formula. We have the two following propositions concerning the verification of a path formula expressed by simple colors:

Proposition 6: Conjunctive and Disjunctive Path Formulas

(1) $SRG_{\mathcal{N},f}, \hat{\pi} \models \Lambda_{c \in D}, EXg_c \Leftrightarrow SRG_{\mathcal{N},f}, \hat{\pi} \models EXg_{\tau(D)}$.

(2) $SRG_{\mathcal{N},f}, \hat{\pi} \models \Lambda_{c \in D_i, c' \in D_j}, E(f_c U g_{c'}) \Leftrightarrow SRG_{\mathcal{N},f}, \hat{\pi} \models E(f_{\tau(Di)} U g_{\tau(Dj)})$.

where f_c, g_c are atomic propositions, D_i, D_j are static subclasses or cartesian products of static subclasses and a,b are chosen arbitrary from D_i and D_j respectively. Let us recall that $\tau(D_i)$ (resp. $\tau(D_j)$) represents the isolation and the transformation processes of the static subclasses D_i (resp. D_j)

Proof: The prove can be deduced from proposition 2 of Section 4.2.

Effectively, the former proposition can be generalized also to complex color domain. and in the case of simple or complex color domain, the verification is restricted to the verification of the property for the colors isolated by means of the function τ. Since this colors are isolated in new static subclasses that appear in the markings of the graph. In consequence, the verification of the formula can be performed directly by the application of both the verification rules depending on the linear and boolean operators (proposition 7) and the verification rules of the atomic subformula contained in the formula (propositions 3, 4 and 5).

Now, we can deal with CPN-CTL* formulas, by applying the semantic rules depicted in section 3.2. Let us consider f_1, f_2 be two state formulas and g_1, g_2 two path formulas, let $\hat{\pi} = \hat{m}_0, \ldots, \hat{m}_n$ a path in the $SRG_{\mathcal{N},f}$. We note $\hat{\pi}_i = \hat{m}_i, \ldots, \hat{m}_n$. The following proposition introduces the general rules used to check CTL* formula through $SRG_{\mathcal{N},f}$:

Proposition 7: Generalization to CPN-CTL* Formulas

(1) $SRG_{\mathcal{N},f}, \hat{m}_k \models \neg f_1$ iff Not $SRG_{\mathcal{N},f}, \hat{m}_k \models f_1$.

(2) $SRG_{\mathcal{N},f}, \hat{m}_k \models f_1 \vee f_2$ iff $SRG_{\mathcal{N},f}, \hat{m}_k \models f_1$ or $SRG_{\mathcal{N},f}, \hat{m}_k \models f_2$.

(3) $SRG_{\mathcal{N},f}, \hat{m}_k \models E(g_1)$ iff $\exists \pi$ starting from m_k such that $SRG_{\mathcal{N},f}, \hat{m}_k \models g_1$.

(4) $SRG_{\mathcal{N},f}, \hat{\pi} \models \neg g_1$ iff it is false that $SRG_{\mathcal{N},f}, \hat{\pi} \models g_1$.

(5) $SRG_{\mathcal{N},f}, \hat{\pi} \models g_1 \vee g_2$ iff $SRG_{\mathcal{N},f}, \hat{\pi} \models g_1$ or $SRG_{\mathcal{N},f}, \hat{\pi} \models g_2$.

(6) $SRG_{\mathcal{N},f}, \hat{\pi} \models X(g_1)$ iff $SRG_{\mathcal{N},f}, \hat{\pi}_1 \models g_1$.

(7) $SRG_{\mathcal{N},f}, \hat{\pi} \models g_1 \, U \, g_2$ iff $\exists k \geq 0$ such that: $SRG_{\mathcal{N},f}, \hat{\pi}_k \models g_2$ and $\forall i$ such that $k > i \geq 0$ we have $SRG_{\mathcal{N},f}, \hat{\pi}_i \models g_1$.

Proof: The proof of the rules can be based simply on the definitions of the operators used in the former formulas.

Example 7:

Let us perform model checking of $f = \Lambda_{Ph \in PH} AG[Ph \bullet Thinking \Rightarrow F(Ph \bullet \mathrm{Eating})]$ *through* $SRG_{\mathcal{N},f}$ *depicted in Figure 3.*

Formula f can be normalized by using the standard transformation rules presented in [8] *and recalled in the introduction of section 3. This yields the following form of f:* $f = \Lambda_{Ph} A\neg\{TrueU\neg[(\neg Ph \bullet Thinking) \vee (TrueU(Ph \bullet \mathrm{Eating}))]\}$ *. Such a formula is verified using the propositions of section 5: from proposition 4, we can reduce the model checking of f to the model checking of* f_1 *expressed by* Ph_1 *selected by using rule* 5 *of rules 4.3.2.1 and the associated transformation function:*
$f_1 = A\neg\{TrueU[(Ph_1 \bullet Thinking) \wedge \neg(TrueU(Ph_1 \bullet \mathrm{Eating}))]\}$. *Then formula* f_1 *is checked recursively using the rules presented in [7] which correspond, inductively, to the verification of the subformulas. Finally, Proposition 3 is applied in order to check the atomic subformulas,* $f_{1,1} = Ph_1 \bullet Thinking$ *and* $f_{1,2} = Ph_1 \bullet Eating$*, at the end of the recursion loop. In consequence, by scanning* $SRG_{\mathcal{N},f}$ *of figure 3 we can find an infinite path* $\hat{\pi} = \hat{m}_2, \hat{m}_4, \hat{m}_3, \hat{m}_2, \ldots$ *through which f does not hold (*$\hat{m}_i$ *is the symbolic marking corresponding to node i)*

6. The Model Checking Equivalence

We prove that the proposed verification method through structure $\mathrm{SRG}_{\mathcal{N},f}$ is equivalent to the one performed through the reachability graph of $\mathcal{N}$ noted $\mathrm{RG}_{\mathcal{N}}$.

Theorem: Model checking equivalence for atomic formulas

(i) state formulas:

$\mathrm{RG}_{\mathcal{N}}, m' \models f \Leftrightarrow \mathrm{SRG}_{\mathcal{N},f}, \hat{m} \models f,\ \forall m' = s(\hat{m})$ where $s \in \mathrm{AS}(\mathcal{N}_f)$.

(ii) path formulas: $\mathrm{RG}_{\mathcal{N}}, \pi \models f \Leftrightarrow \mathrm{SRG}_{\mathcal{N},f}, \hat{\pi} \models f$

(a) From M, if $\pi = m_0, m_1, \ldots, m_n$ is a path of markings for which $M, \pi \models f$ then there is an image $\hat{\pi} = \hat{m}_0, \ldots, \hat{m}_n$ of corresponding representatives where $\mathrm{SRG}_{\mathcal{N},f}, \hat{\pi} \models f$.

(b) From $\mathrm{SRG}_{\mathcal{N},f}$, if $\hat{\pi} = \hat{m}_0, \ldots, \hat{m}_n$ is a symbolic path for which f holds then for every marking m_0' such that $s(m_0') = \hat{m}_0$ where $s \in \mathrm{AS}(\mathcal{N}_f)$ in $\mathrm{RG}_{\mathcal{N}}$ there exists a corresponding path $\pi = m'_0, m'_1, \ldots, m'_n$ through which the formula f, holds.

Proof:

For (i), the equivalence is proved by the following reasoning: assume that s is a symmetry of $\mathrm{AS}(\mathcal{N}_f)$ such that $s(m') = \hat{m}$. In consequence $\mathrm{RG}_{\mathcal{N}}, s(m') \models s(f)$ Since $s(f) = f$ by definition of $\mathrm{AS}(\mathcal{N}_f)$, we have $\mathrm{SRG}_{\mathcal{N},f}, \hat{m} \models f$.

For (ii)-(a) The proof is immediate since for any firing sequence there is a symbolic firing sequence in the associated SRG [6].

For (ii)-(b) Let us consider symbolic paths of the form $\hat{\pi} = \hat{m}_0, \hat{m}_1, \ldots, \hat{m}_n$ we have the two following cases:

(1) f=EX(g), from rules 3 and 6 of proposition 7 we have a path $\hat{\pi} = \hat{m}_0, \hat{m}_1$ of length 1, i.e. edge in $\mathrm{SRG}_{\mathcal{N},f}$, such that $\mathrm{SRG}_{\mathcal{N},f}, \hat{m}_1 \models g$. From part (i) of this theorem

we have $RG_{\mathcal{N}}, m' \models g \Leftrightarrow SRG_{\mathcal{N},f}, \hat{m}_1 \models g,\ \forall m' = s(\hat{m}_1)$ where $s \in AS(\mathcal{N}_f)$. Since the g holds for all the marking represented by $\hat{m}_1$, the formula f holds for all the corresponding ordinary paths represented by the symbolic path $\hat{\pi} = \hat{m}_0, \hat{m}_1$. In consequence the verification of such formula is equivalent through the two graphs.
(2) $f=E(g_1 U g_2)$ from rule 3 and 7 $\exists k \geq 0$ such that: $SRG_{\mathcal{N},f}, \hat{\pi}_k \models g_2$ and $\forall i$ such that $k > i \geq 0$ we have $SRG_{\mathcal{N},f}, \hat{\pi}_i \models g_1$. In each state of $\hat{\pi}_i = \hat{m}_i, \ldots, \hat{m}_n$ the atomic formula g_1 holds and from (i) it holds for all the ordinary markings represented by a symbolic marking. In consequence, the formula g_1 holds for all the corresponding ordinary paths represented by the symbolic path $\hat{\pi}_i = \hat{m}_i, \ldots, \hat{m}_n$. Same proof for g_2. In consequence the verification of f is equivalent through both the ordinary and the symbolic graph.

7. Conclusion

The proposed technique to verify CTL* formulas is derived from the symbolic theory, based on well-formed nets and the formal approach of model checking in [1]. Due to the ability of refining static subclasses in order to take the symmetries expressed in a formula into account, we have shown that CTL* formula are able to be checked through a symbolic reachability graph built on the refined static subclasses. The main advantage of our method is that it can lead to a complete automatic verification, due to the automatic building of SRG that takes the structural symmetries of system objects into account. Like in [1][7][8], a graph is built for a class of properties specified by a class of formulas which correspond to the same automorphism group. The proposed methodology allows one to evaluate in advance (structurally) the efficiency of using the system symmetries in the model checking of a formula. Effectively, two cases appear in limits: the best one where the structural symmetries of the WN are entirally used, causing a maximal condensation of the represented state space and the worse case in which any group of symmetrical objects is reduced to a singleton, leading the new SRG to be large as well as the standard reachability graph.In our context, the formal approach presented in [1] must be adjusted since the correspondence lemma between the quotient and the ordinary structures is not respected by the original SRG.Currently, we aim at extending this method in order to deal with specifications, in terms of well-formed nets, based on partial symmetries and the associated Extended Symbolic Reachability Graph (ESRG) [12]. This correspond to the case of a system, the behaviors of which sometimes depend on the process identities (i.e. static priorities based on identities), and sometimes not. However, our perspective is to enforce the efficiency of model checking process by relaxing the dependency of the formula on the graph computations.

8. References

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

[2] E. Allen Emerson, A. Parsad Sistla, "Utilizing Symmetry when Model Checking under Fairness Assumptions: An Automata-theoric Approach", 7th conference on Computer Aided Verification (CAV), LNCS 939, pp. 309-324, Liège, Belgiume, July 1995.

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

[4] F. Michel, P. Azéma, F. Vernadat. "Permutable Agents and Process Algebra", In Proc. of TACAS’96, pages 187-207, Passau, Germany, 1996, Springer-Verlag, LNCS 1055.

[5] 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.

[6] G. Chiola, R. Gaeta, "Efficient Simulation of Parallel Architectures Exploiting Symmetric Well-formed Petri Net Models", Sixth International Workshop on Petri nets and Performance Models, Durham, NC, USA, IEEE Computer Society Press, October 1995.

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

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

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

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

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

[12] S. Haddad, JM. Ilié, B. Zouari, M. Taghelit, "Symbolic Reachability Graph and Partial Symmetries", In Proc. of the 16th International Conference on Application and Theory of Petri Nets, pp 238-257, Torino, Italy, June 1995.

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

[14] Claude Dutheillet, "Symmetries dans les Reseaux Colorés, Définition, Analyse et Application a l’Evaluation des Performances", PH-D thesis, MASI Laboratory, University of ParisVI, Paris, France, march 1992.

[15] 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.