Model Checking Through Symbolic Reachability Graph

Jean Michel Ilié[*+] and Khalil Ajami[*]

[*] MASI-CNRS URA 818
Univ. Pierre et Marie Curie-PaisVI
4, pl. Jussieu, 75252 Paris

[+] Univ. René Descartes-ParisV-IUT
143, av. de Versailles 75016 Paris

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

Abstract. A Symbolic Reachability Graph (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. In this paper, we show how to make operational the CTL* formal checking system presented in [1]. Our technique consists in exploiting the SRG by taking into account the object symmetries only if they leave the formula invariant. The difficulty to bypass is that SRG does not preserve explicitly the behavior of the objects specified within formulas. This 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.

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. Effectively, a global state-transition graph of a system composed of many identical (isomorphic) processes, exhibits a great deal of symmetry reflected in the group of permutations of processes. In the same way, any formula exhibits a certain degree of symmetry, reflected in the group of permutations of processes that leave the formula invariant. The reduction technique consists in gathering into equivalence classes, the states which cause the same behavior by building a quotient structure defined on both graph and formula symmetry groups. In [1], a formal approach of model checking through such quotient structure is proved.

The aim of this paper is to present a technique which makes the former approach operational. Our method consists in exploiting the theory of Well-formed net and the associated symbolic reachability graph, proposed in [2][3]. Well-formed nets (WN) are Colored Petri Nets (CPN) 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) abstract the state space of a system specified via a Well-formed net, by representing classes of states and actions. The equivalence relation between states 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. Effectively, SRG are built according to definitions of symmetrical object groups for which the identity of objects is not preserved, i.e. only the nature of objects and the cardinality of the groups are known. With respect to a given formula which specifies a property, the starting point of our method consists in determining the groups of symmetrical objects that leave the formula invariant in order to isolate objects specified within this formula from their classes in the WN model. 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 groups of symmetrical objects. The WN defined on such refined groups can be used to build a suitable SRG, through which the formula can be checked. 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 their major properties as well as the difficulties to perform model checking through it; part 3 presents the refinement approach and the new system specification that results from this refinement. This specification is used to build a new SRG; part 4 defines the verification process of a CTL* formula through the resulting SRG; part 5 contains the formal proof of the validity of the presented work; part 6 is our conclusion. We assume that the reader knows the basic theories of CPN, reachability graph and temporal logic. However, some known notions 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) [2]. Such nets are colored Petri nets but their color domains and the associated functions are defined from *classes* and *static subclasses* of primitive objects. Classes gather 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 to preserve the successor relation of objects. For example, one may define class Process=$\{p_1, p_2, p_3\}$ in order to model three ordered processes, and may split Processes in two static subclasses the first is Interactive=$\{p_1,p_2\}$ and the second is 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. The colors belonging to a place (with respect to the place domain) form the place marking. A state of a system is a vector of marked places called marking.
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 we use unordered classes to bring out the reducing effects of using symmetries.

Example 1: *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 having 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: $m(Thinking) = Ph_1$; $m(Waiting)=<Ph_2,f_1>$; $m(Eating)=<Ph_3,f_2+f_3>=<Ph_3,f_2>+<Ph_3,f_3>$; $m(Forks) = 0$.*

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

2.1. Symbolic Marking

Roughly speaking, a symbolic marking is a representative of an equivalence class of markings, for which the equivalence relation is deduced from a set of *admissible symmetries* of colors. Such symmetries operate on the classes of the studied WN. They preserve the static subclasses and the successor relation on ordered classes. Let $\mathcal{N}$ be a WN.

Definition 2.1.1: Group of Admissible Symmetries
Let $CD=\{C(r) | r\in P\cup T\}$ be the set of color domains attached to either places or transitions in $\mathcal{N}$. A symmetry S on a color domain C(r) of CD is a permutation on C(r). A set of symmetries, ζ, on $\mathcal{N}$ is defined as the family of symmetries S on the elements of CD. (ζ, o) forms a group called 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: $\forall s\in AS(\mathcal{N}), \forall c\in D_{i,q}$, $s(c)\in D_{i,q}$.

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 cause the same behavior:

Property 2.1.2: Behavioral Equivalence of Symmetrical Colors
Colors of a static subclass in a given marking cause the same behaviors. In consequence, they cause equivalent markings and firing sequences.

So, for a given state, symmetrical colors can be aggregated and represented by their quantity and their static subclass while forgetting their identities. Such representation corresponds to the notion of dynamic subclass that express symbolic marking.

Definition 2.1.3: Dynamic Subclasses and the Associated Symbolic Markings
Let C_i be a class of $\mathcal{N}$. A dynamic subclass of C_i represents a set of colors belonging to a static subclass of C_i. It is featured by its nature and its behavior (static subclass) as well as its cardinality. We note Z_i^j the jth dynamic subclass of C_i. A symbolic

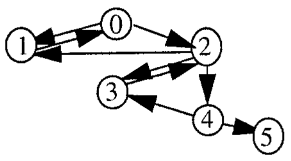
marking is a representative of an equivalence class of markings according to AS($\mathcal{N}$). It is expressed in terms of vector of marked places where colors are symbolically represented by dynamic subclasses. Its useful notation is $\hat{m}$.

Example 2: *The marking of the example 1 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:* $\hat{m}(Thinking)=Z_1^1$; $\hat{m}(Waiting) = <Z_1^2, Z_2^1>$ *where* $|Z_2^1| = 1$; $\hat{m}(Eating)=<Z_1^3, Z_2^2>$ *where* $|Z_2^2| = 2$; $\hat{m}(Forks)= 0$. *In fact,* $\hat{m}$ *represents nine markings obtained by operating the nine possible permutations on the philosophers and the associated forks presented in* $\hat{m}$.

2.2. Symbolic Reachability Graph Construction

In [2], a symbolic firing rule is introduced in order to compute directly a new symbolic marking 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 firing rule allow us to build SRG. In this graph, the nodes are the symbolic markings expressed in a canonical form.

Example 3: *Figure 1 represents the SRG for three philosophers. It contains 6 nodes and 9 arcs while the corresponding reachability graph contains 46 nodes and 81 arcs.*



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

Fig. 1. SRG for three Philosophers

2.3. Checking properties through SRG

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. Fortunately, since SRG is built using admissible symmetries, a new property can be deduced from 2.1.2 concerning the representation of color behaviors:

Property 2.3.1: Symbolic Representation of Symmetrical Color Behaviors
Colors of a static subclass in a given symbolic marking cause the same behaviors. In consequence, they cause the same symbolic markings and firing sequences.

Consequently, it is sufficient to prove that a property is verified for a given color to prove that it is verified for its static subclass. In this case, we can solve the problem of path properties expressed symbolically by checking the property for any arbitrary color of the concerned class. 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 all the colors that have the same behavior. In consequence, several interesting properties which are not color dependency can be checked like the absence of deadlock, the existence of home space (resp. unavoidable home space) or infinite path. The next section copes with the SRG advantages and drawbacks by presenting our operational model which enables the model checking through SRG.

3. SRG Built with Respect to a Formula

In this section, we show how to check formulas directly through the symbolic reachability graph. Formulas are expressed with Computational Tree Logic star (CTL*) proposed in [1], [4] and [5]. In such logic, there are two types of formulas: state formulas (which are true in a specific state) and path formulas (which are true in the states along a specific path) [7]. Linear temporal operators are introduced as follows: F (sometimes), G (always), X (next time) and U (strong until). Moreover, path quantifiers are represented either by symbol A for all full paths or symbol E for some full paths.
In order to specify properties of WN, formulas must be expressed in terms of classes and colors of classes. Moreover, they must refer to places since colors in places represent the system variables assigned to specific values. In fact, state formulas express that tokens (i.e. markings) exist in places (i.e. mark the places). Depending on the fact that a color domain of a place p can be built on an object class or a cartesian product of object classes, we introduce two kinds of atomic formulas:
(i) $\alpha c \in D_{i,q}, a = c \bullet p$ tests if colors c of $D_{i,q}$ mark p according to quantifier α (universal or existential) where $D_{i,q}$ is a static subclass of a class C_i;
(ii) $\alpha_{i_1} c_{i_1} \in D_{1,q_1}, \ldots, \alpha_{i_n} c_{i_n} \in D_{n,q_n}, a = \langle c_{i_1}, \ldots, c_{i_n} \rangle \bullet p$ tests if the tuples of colors $\langle c_{i_1}, \ldots, c_{i_n} \rangle$ of a color domain $D_{1,q_1} \times \ldots \times D_{n,q_n}$ mark p according to quantifiers α_{i_j}, where $D_{1,q_1}, \ldots, D_{n,q_n}$ are static subclasses.
Since our model checking is based on propositional formulas, universal and existential quantifiers are re-expressed in terms of conjunction and disjunction operators. The previous types of atomic formulas become respectively: (i) $a = \theta_{c \in D}(c \bullet p)$, (ii) $a = \theta_{c_{i_1} \in D_{1,q_1}} \ldots \theta_{c_{i_n} \in D_{n,q_n}} \left(\langle c_{i_1}, \ldots, c_{i_n} \rangle \bullet p \right)$ where θ is either the disjunction or conjunction operator. At last, it must be noted that within some formulas, the colors that mark a place can be restrained in order to refer to a subdomain of the place color domain. Therefore, with respect to a place p appearing in a formula f, we introduce a projection function which restrains the marking of p to the subdomain of p expressed in f and noted $C(p)|_f$: $Prj(f,p): Bag(C(p)) \rightarrow Bag(C(p)|_f)$. Bag(C) denotes the set of markings that can be built on class C. The projection function on place p, with respect to f, can be generalized for all marked places appearing in f: $Prj_f = \langle Prj(f,p_1), Prj(f,p_2), \ldots \rangle$.

***Example 4:** Let $f = \Lambda_{Ph \in PH}(Ph \bullet Eating)$ be an atomic state formula and consider the symbolic marking $\hat{m}$ of example 2. We have $Prj_f(\hat{m}) = \hat{m}(Eating)|_f$, so, the verification must be processed on $\hat{m}$ (Eating). Moreover, only the class of philosophers represented in $\hat{m}$ by the dynamic subclass Z_1^3, is taken into account.*

The next subparagraph presents the transformation process of the SRG, with respect to a formula, in order to check the former atomic propositional formulas directly.

3.1. Transformation of SRG with respect to 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 those 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 (admissible symmetries that leave the formula invariant). More practically, the former three points leads us to succeed the two following stages: (1) the first stage consists in determining the group that reflects the symmetries expressed by the isolated colors; (2) the second stage consists in considering 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 specification of static subclasses. It corresponds to a refinement of static subclasses in order to isolate the formula colors. Let us assume the existence of $\mathcal{N}$, a given WN. The determination of a subgroup of $(AS(\mathcal{N}),o)$ leaving a formula invariant requires to express the structural symmetries reflected in the formula. Such symmetries can be defined by the notion of automorphism group of a formula.

Definition 3.1.1: Automorphism Group of a Formula f
Aut(f), is the group of permutations of colors that leave f invariant.

The former definition of automorphism group means that: $\forall s \in Aut(f), s(f) = f$, but it does not always ensure that are respected neither the splitting of colors in static subclasses nor the restrictions imposed on symmetries for ordered classes. Therefore we must consider a subgroup of $AS(\mathcal{N}) \cap Aut(f)$ which expresses the admissible symmetries that leave f invariant. Of course, the largest subgroup, $AS(\mathcal{N}_f) = AS(\mathcal{N}) \cap Aut(f)$, is desirable for maximal compression. $AS(\mathcal{N}_f)$ is a restriction of the admissible structural symmetries enabled in $\mathcal{N}$, therefore, it is always possible to form a new WN, $\mathcal{N}_f$ 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 3.1.2: Group of Admissible Symmetries with Respect to a Formula f
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: $AS(\mathcal{N}_f) = AS(\mathcal{N}) \cap Aut(f)$.
(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) the set of all the permutations on C_i. In consequence, if Aut(f) = $\bigcup_i$Sym(C_i) then, AS($\mathcal{N}_f$)=AS($\mathcal{N}$). The SRG built from $\mathcal{N}_f$, 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 3.1.3: Correspondence property
Let $\hat{m}$ and $\hat{\pi}$ be respectively a marking and a path of $SRG_{\mathcal{N}f}$, we have two properties: (1)$\forall s \in$ AS($\mathcal{N}_f$), f holds in $\hat{m} \Leftrightarrow$ f holds in s($\hat{m}$).

(2)$\forall s \in$ AS($\mathcal{N}_f$), f holds through $\hat{\pi} \Leftrightarrow$ f holds through s($\hat{\pi}$).

Those property enable 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 5. The computation of AS($\mathcal{N}_f$) consists, mainly, in determining Aut(f) since AS($\mathcal{N}$) is given initially by $\mathcal{N}$.

3.2. Determination of Aut(f)

In [1], the rules which determine the automorphism group are presented in the context of CTL* formulas model checking through a state transition graph. Unfortunately, These rules can not be applied directly in our context and many difficulties appear when we perform model checking through SRG due to the particularity of color representations in the symbolic approach (see section 2.3). In consequence, the rules are adapted to cope with those difficulties and to allow the building of a SRG through which a formula can be checked directly. Let θ be either $\wedge$ or V. In the following, we consider a formula f built on a class C or a static subclass D of C or a subset B of D.

Rules 3.2.1: Generic rules to determine Aut(f)

(1) If f is trivial (f or $\neg$f is a validity) then Aut(f) = $\bigcup_i$Sym(C_i) for all C_i.

(2) If f= $g_{b \in C}$ built for a specific color b then Aut(f)=Sym(C\{b}).

(3) If $f = \theta_{(c_i \neq c_j) \in C} g_{c_i, c_j}$ then $\forall c_a, c_b \in C$ Aut(f)=Aut(g_{c_a, c_b}).

(4) If $f = \theta_{c_i \in D} g_{c_i}$ or $f = \theta_{c_i \in B} g_{c_i}$ then (a) Aut(f)=Sym(D) if f is a state formula (b) $\forall c \in D$ Aut(f)=Sym(D\{c}), if f is a state formula.

(5) If $f=\theta_{c_{i_1} \in D_1}, \ldots, \theta_{c_{i_n} \in D_n} g_{c_{i_1}, \ldots, c_{i_n}}$ or $f= \theta_{c_{i_1} \in B_1}, \ldots, \theta_{c_{i_n} \in B_n} g_{c_{i_1}, \ldots, c_{i_n}}$ then (a) Aut(f)=$\bigcup_i$Sym(D_i) if f is a state formula (b) $\forall c_{a_1} \in D_1, \ldots, \forall c_{a_n} \in D_n$ Aut(f)=Sym($D_1 \setminus \{c_{a_1}\}$)$\cup \ldots \cup$Sym($D_n \setminus \{c_{a_n}\}$).

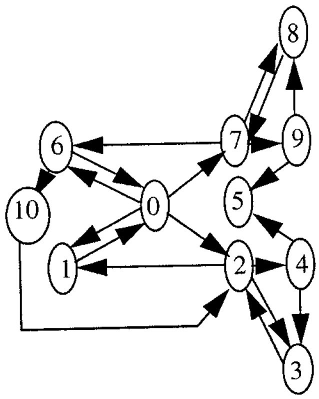
(6) If f is a temporal formula that has one of the forms EXg, EFg, EGg, where g has one of the forms presented by the current rules, then Aut(f)=Aut(g).

(7) 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).

(8) If f is a formula built on many static subclasses from many classes of colors the former rules are applied separately for each static subclass.

The major modifications of Aut(f) determination appear in rules (3), (4), (5). Despite the presence of symmetries in the corresponding formulas, we must isolate arbitrary colors from the concerning classes or static subclasses to perform model checking. The isolation process aim to bypass difficulties previously mentioned in section 2.3: In (3), the verification of the associated predicate $c_i \neq c_j$ requires the knowledge of the identity of the concerned colors, in consequence, it is sufficient to isolate two arbitrary colors from the class in order to detect their behavior. Due to property 2.3.1 the behaviors of such colors is equivalent to the behavior of each couple of colors of their static subclasses. In (4) and (5), the whole symmetries of the colors of D can be saved in case of state formulas since the verification of a state property by any arbitrary quantity of colors from the same static subclass (a dynamic subclass) is sufficient to prove that the property is verified by all the static subclass (property 2.3.1). Contrary, in case of path properties, we must isolate an arbitrary chosen color in order to detect its presence along a symbolic path. In fact, nothing can ensure that a color, from a static subclass, follows a symbolic path between two symbolic markings even if the color marks those two markings. Consequently, it is sufficient to isolate an arbitrary color from the concerned static subclass. Due to property 2.3.1, the behavior of such color through the required symbolic path describes the behavior of the static subclass through that path.

The following formula f expresses that, there is a path through which it is always possible for any philosopher who waits, to turn back in the future, to a state in which he thinks: $f = \wedge_{Ph \in PH} EG[Ph \bullet WaitingFork \rightarrow F(Ph \bullet Thinking)]$. *Initially, PH has only one static subclass, PH itself. Formula f is a path formula which corresponds to rule 4-b of rules 3.2.1, then the automorphism group of f is:* $Aut(f)=Sym(PH \setminus \{Ph_1\})$ *where* Ph_1 *is chosen arbitrary from PH. We have* $AS(\mathcal{N})=Sym(PH)$*, in consequence, the new group of admissible symmetries is* $AS(\mathcal{N}_f)=Sym(PH \setminus \{Ph_1\})$*. 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 2. 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 46 nodes and 81 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. 2. SRG built with respect to formula f for three philosopher

The graph depicted in figure 2 contains the SRG presented in figure 1 (nodes 0 to 5). However, the isolated color Ph_1 of the static subclass FirstPhilosopher adds, by its behavior, new paths to the graph (nodes 6 to 10). The property specified by formula f will be verified for Ph_1, however, the result will be generalized to all the colors of the static subclass due to property 2.3.1. In fact, since the static subclass FirstPhilosopher contains Ph_1 only, any of its dynamic subclass is a representative of Ph_1.

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

4. 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 formulas expressed with tuples. Finally, we consider general CTL* formulas. We use the standard notation $SRG_{\mathcal{N},f}, \hat{m}_k \models f$ to indicate that a state formula f holds at a symbolic marking $\hat{m}_k$ (state of SRG) in the structure $SRG_{\mathcal{N},f}$; similarly, $SRG_{\mathcal{N},f}, \hat{\pi} \models f$ means that path formula f holds along $\hat{\pi}$. Let us consider atomic formulas 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 formula $f = \vee_{c \in D}(c \bullet p)$ holds in a symbolic marking of p if at least one color of D marks p. Similarly, formula $f = \wedge_{c \in D}(c \bullet p)$ 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 the color domains of a place, in a WN, can be complex (i.e. cartesian product of classes), therefore, a pattern matching must be introduced against the tuples which are expressed in the formula and those expressed in the marking of places. This matching is processed according to the application of Prj_f (see introduction of section 3) on the corresponding marking.

Proposition 1: Verification of Disjunctive Atomic Formula on a Simple Domain
$SRG_{\mathcal{N},f}, \hat{m} \models f$ where $f = \vee_{c \in D}(c \bullet p)$ iff $\exists Z \subseteq D$ such that $Z \in Prj_f(\hat{m}(p))$.

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 $Prj_f(\hat{m}(p))$. the $\Leftarrow$ direction is proved since the presence of a dynamic subclass of D in $Prj_f(\hat{m}(p))$, means that a quantity of colors of D exists in $\hat{m}$ with respect to p.

Proposition 2: Verification of Conjunctive Atomic Formula on a Simple Domain
$SRG_{\mathcal{N},f}, \hat{m} \models f$ where $f = \wedge_{c \in D}(c \bullet p)$ iff the two conditions hold:

(1) $\exists Z$ such that $\left(Z = \{Z^j \subseteq D | Z^j \in Prj_f(\hat{m}(p))\}\right)$, (2) $\forall Z^j \in Z$ we have: $\sum_j |Z^j| = |D|$.

Proof: The proof is very similar to the one of proposition 1 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 $Prj_f(\hat{m}(p))$. the $\Leftarrow$direction uses the same reasoning for the second condition.

Example 5: *Let* $f = \Lambda_{Ph \in OtherPhilosophers}(Ph \bullet Thinking)$ *and consider* $\hat{m}_0$ *the initial symbolic marking corresponding to node 0 in Figure 2 . In* $\hat{m}_0$*, the marking of place "Thinking" is* $\hat{m}_0(Thinking) = Z_1^1 + Z_2^1$ *such that* $|Z_1^1| = 1$ *and* $|Z_2^1| = 2$ *where* Z_1^1 *belong to FirstPhilosopher and* Z_2^1 *belong to OtherPhilosophers. Hence, the atomic formula f holds in* $\hat{m}_0$ *since a dynamic subclass of OtherPhilosophers exists in this marking and its cardinality is equal to the one of OtherPhilosophers (proposition 1).*

Propositions 1 and 2 can be simply generalized to complex domains as follows:

Proposition 3: Extension to Atomic Formulas on a Complex Color Domain

$SRG_{\mathcal{N}f}, \hat{m} \models \left[f = \theta_{c_{i_1} \in D_1} \cdots \theta_{c_{i_n} \in D_n}\left(\langle c_{i_1}, \ldots, c_{i_n}\rangle \bullet p\right)\right]$ iff the two conditions hold:

(1) for each component $c_{i_q} \in D_q, \exists Z_q, \left(Z_q = \{Z_q^j \subseteq D_q | Z_q^j \in Prj_a(\hat{m}(p))\}\right)$.

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

Proof: The proof is a simple generalization of those of 1 and 2.

In order to deal with CTL* formulas, the former verification process can be generalized by using the rules introduced in [4] and [5]. They have been applied in the general context of state transition graph. They can be applied again due to property 3.1.3 which expresses correspondences of states and paths between the reachability graph and $SRG_{\mathcal{N}f}$. The case of quantified path formulas can be also considered despite the problem presented in section 2.3 for such formulas. Effectively, due to property 2.3.1 the verification of a quantified path formula can be reduced to the verification of the same formula for one arbitrary object of its quantification domain. Let g_1 and g_2 be two path formulas such that $g_1 = \theta_{c_i \in D_i} g_{c_i}$ and $g_2 = \theta_{c_{i_1} \in D_1} \cdots \theta_{c_{i_n} \in D_n}\left(\langle c_{i_1}, \ldots, c_{i_n}\rangle \bullet p\right)$ where D_1.., D_i,..D_n are static subclasses.

Proposition 4: Conjunctive and Disjunctive Path Formulas

(1) $SRG_{\mathcal{N}f}, \hat{\pi} \models g_1$ iff $SRG_{\mathcal{N}f}, \hat{\pi} \models g_{c_a}$ where c_a is the color chosen to be isolated by the rule 4 of rules 3.2.1.

(2) $SRG_{\mathcal{N}f}, \hat{\pi} \models g_2$ iff $SRG_{\mathcal{N}f}, \hat{\pi} \models g_{c_{a_1}, \ldots, c_{a_n}}$ where $\langle c_{a_1}, \ldots, c_{a_n}\rangle$ is the tuple of colors chosen to be isolated by the rule 5 of 3.2.1.

Proof: This proof can be deduced from 2.3.1 for each static subclass separately.

It must be noted that any CTL* formula can be transformed to be expressed by the forms presented in the model checking rules of [4][5]. The transformation is performed using the following general transformations [5]: (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) $Ff \equiv TrueUf$; (4) $Gf \equiv \neg F \neg f \equiv \neg(TrueU \neg f)$.

Example 6: *Let us perform the model checking through SRG depicted in Figure 2 and built with respect to* $f = \Lambda_{Ph \in PH} EG[Ph \bullet WaitingFork \rightarrow F(Ph \bullet Thinking)]$ *.This formula is transformed using the four transformation rules presented in [5] and reported in this section previously. The transformed formula:*
$f = \Lambda_{Ph} E \neg \{TrueU \neg [(\neg Ph \bullet WaitingFork) \vee (TrueU(Ph \bullet Thinking))]\}$ *is verified using the propositions of section 4: from proposition 4, we can reduce the model checking of f to the model checking of* f_l *expressed by* Ph_l *selected by using rule 5 of rules 3.2.1:* $f_l = E \neg \{TrueU[(Ph_l \bullet WaitingFork) \wedge \neg(TrueU(Ph_l \bullet Thinking))]\}$, *then formula* f_l *is checked recursively using the rules presented in [4] and correspond to the temporal and boolean operators expressed in the formula. Finally, Proposition 1 is applied in order to check the atomic subformulas,* $f_{l,1} = Ph_l \bullet WaitingFork$ *and* $f_{l,2} = Ph_l \bullet Thinking$, *at the end of the recursion loop. In consequence, by scanning* $SRG_{\mathcal{N}f}$ *of figure 2 we can find a path* $\hat{\pi} = \hat{m}_8, \hat{m}_7, \hat{m}_6, \hat{m}_0$ *through which f holds (* $\hat{m}_i$ *is the symbolic marking corresponding to node i).*

5. The Model Checking Equivalence

We prove that our verification method through $SRG_{\mathcal{N}f}$ is equivalent to the one performed through the reachability graph of $\mathcal{N}$ noted $RG_{\mathcal{N}}$.

Theorem:

(i) Model checking equivalence for state formulas:
$RG_{\mathcal{N}}, m' \models f \Leftrightarrow SRG_{\mathcal{N}f}, \hat{m} \models f$, $\forall m' = s(\hat{m})$ where $s \in AS(\mathcal{N}f)$.

(ii) Model checking equivalence for path formulas:

(a) From M, if $\pi = m_0, \ldots, m_n$ is a path where $M, \pi \models f$ then there is $\hat{\pi} = \hat{m}_0, \ldots, \hat{m}_n$, a corresponding representatives, such that $SRG_{\mathcal{N}f}, \hat{\pi} \models f$.

(b) From $SRG_{\mathcal{N}f}$, if $\hat{\pi} = \hat{m}_0, \ldots, \hat{m}_n$ is a path of symbolic markings for which $SRG_{\mathcal{N}f}, \hat{\pi} \models f$ then $\forall m_0' = s(\hat{m}_0)$ where $s \in AS(\mathcal{N}f)$, and $\forall \pi = m'_0, \ldots, m'_n$ where $m_i' = s(\hat{m}_i)$ we have $RG_{\mathcal{N}}, \pi \models f$.

Proof: For (i), the equivalence is proved by the following reasoning: assume that s is a symmetry of $AS(\mathcal{N}f)$ such that $s(m') = \hat{m}$. In consequence $RG_{\mathcal{N}}, s(m') \models s(f)$ Since $s(f) = f$ by definition of $AS(\mathcal{N}f)$, we have $SRG_{\mathcal{N}f}, \hat{m} \models f$. For (ii), direction (a) is immediate since for any firing sequence there is a symbolic firing sequence in the associated SRG[3]. Direction (b) is proved by induction on the number of path markings. Assume that $\hat{\pi} = \hat{m}_0, \hat{m}_1$ and consider an arbitrary marking $m_0' = s(\hat{m}_0)$ where $s \in AS(\mathcal{N}f)$. Assume that there is a marking $m_1' = s(\hat{m}_1)$ for which the formula does not hold on path $\pi = m'_0, m'_1$. In consequence, two possible cases can appear: (1) both source and destination markings of the formula do not hold in the associated source and destination markings in $SRG_{\mathcal{N}f}$ (2) one of them does not hold in the associated source or destination markings in $SRG_{\mathcal{N}f}$. In fact, this is not possible because of (i). If we consider now that (b) holds for a path which contains n markings we can simply deduce that it holds for a path of (n+1) markings by reasoning similarly for each firing of the path.

6. Conclusion

The proposed model checking technique, of CTL* formulas through symbolic reachability graph, 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][4][5], a graph is built for a class of properties specified by a class of formulas which correspond to the same automorphism group. 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 [6]. 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.

7. References

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

[2] 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.

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

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

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

[6] 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.

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