Using Maximality-Based Labeled Transition System Model for Concurrency Logic Verification

Djamel Eddine Saidouni and Nabil Belala
Computer Vision Group, LIRE Laboratory, University of Mentouri, Algeria

Abstract: *In this paper, we show the interest of the maximality-based semantics for the check of concurrent system properties. For this purpose, we use the Maximality-based Labeled Transition System (MLTS) as a behavior model. From this point of view, we can omit action temporal and structural atomicity hypotheses; consequently, we can inherit result of combinatorial state space explosion problem solution based on the use of true concurrency semantics. Properties to be verified are expressed using the Computation Tree Logic (CTL). The main contribution of the paper is to show that model checking algorithms proposed in the literature, which are based on interleaving semantics, may be adapted easily to true concurrency semantics for the verification of new properties classes related to simultaneous progress of actions at different states.*

Keywords: *Concurrent system, maximality-based semantics, process algebra, model checking, CTL.*

Received March 18, 2004; accepted July 8, 2004

1. Introduction

Distributed applications, such as communication protocols, are characterized by their big complexity. The development of these applications requires the consideration of inter-processes cooperation taking in account the indeterminism and synchronization induced by their behavior as well as qualitative properties like the absence of deadlock and starvation. Because of their critical character, these applications are often subjected to austere requirements of reliability, aiming “zero error” quality.

In general, users and environment requirements of a system may not be completely formalized because they make part of real world, and bound to customs or opinions that are sometimes badly conceptualized and often subjective (but which must be taken into a count). Even though we suppose that user requirements were analyzed and understood in a satisfactory way, the activity that consists in establishing a specification of the future system keeps an empirical character, it is a question of building a description of certain real world aspects, which necessarily implies simplification [18]. So, it is necessary to have in mind that formal specification may contain mistakes. These mistakes can result from a bad understanding of user requirements or during the formalization of these requirements. So, the need of formal verification approaches.

Formal methods are used for software packages occurring in critical systems for which certain failings can be catastrophic. These methods are based on the use of formal specification models endowed of rigorous semantics.

The formal verification approach concerned by our study is based on models. In this approach, the application to be verified firstly specified by means of the formal description technique LOTOS [5, 15]. This specification will be translated in an operational way towards an underlying model represented by a graph called Maximality-based Labeled Transition System (MLTS) [9, 21]. The expected properties of the system are written in Computation Tree Logic (CTL) and they are verified by means of the model checking approach (see Figure 3).

In spite of temporal logics facilitation of the specification of systems to be verified [16], model checking approach is limited by the state graph combinatorial explosion problem, particularly when the specification model underlying semantics is the interleaving one. Such semantic is characterized, on one hand by the action temporal and structural atomicity hypothesis and on the other hand by the interpretation of parallel execution of two actions as their interleaving executions in time.

To escape the action atomicity hypothesis imposed by interleaving semantics, new semantics, said true concurrency semantics, were defined in the literature [8, 9, 10, 12, 17, 21]. Among these semantics, we can quote a variant of the maximality semantics [9, 21]; its principle consists in using the dependence relations between actions occurrences and by associating to every state of the system the actions, which are potentially in execution. To make an idea, let us consider the example of the behavior expression E = a; stop | | | b; stop. Figure 1-a shows the transition system obtained by the interleaving semantics. However, the application of the maximality-based semantics generates the transition system of Figure 1-b.

A priori this transition relation is more complicated than that of Figure 1-a, because supplementary information is associated to states and to transitions. However, this information can allow more reductions without loss of information. A possible reduction of Figure 1-b consists in eliminating one of the branches of the graph. It is clear that from Figures 1-c and 1-d we can deduct that actions (*a* and *b*) are concurrent, what implies that this reduction did not provoke loss of information of the behavior of E.

To benefit from the expression power of the Maximality-based Labeled Transition System model (MLTS) and model checking approaches, in this paper, we have developed a model checker based on the MLTS model. This study is given concrete by the realization of the tool for Maximality-based Model Checking (MMC). This tool is integrated in v.2 environment for FOrmal COncurrency Verification Environment (FOCOVE) that we developed in our laboratory. Among others, FOCOVE allows the edition of specifications written in CCS or LOTOS, the compilation of these specifications given are results LTS or MLTS according to the semantic choice. Finally, it allows the formal verification of these specifications.

Along this paper, we assume that the reader is familiar with Labeled Transition System (LTS), model checking and the formal description technique LOTOS [1, 5, 7, 13, 14, 15]. In addition, we show that some fairness and liveness properties may be expressed more easily by means of the MLTS model.

2. Maximality-Based Semantics

A detailed presentation of the maximality semantics can be found in [9, 21]. In this section, we content with a reminding of the definition of the MLTS structures and illustrating the concept by simple examples. An MLTS can be defined as follows:

Definition 1: M being a countable set of event names, a maximality- based transition system of support M is a quintuplet $(\Omega, A, \mu, \zeta, \psi)$ with:

- $\Omega = \langle S, T, \alpha, \beta \rangle$ is a transition system such as:
  - *S* is the countable set of states in which the system can be.
  - *T* is the countable set of transitions indicating the change of system states.
  - α, β are two functions from $t \in T$: $\alpha(t)$ is the origin of the transition and $\beta(t)$ is its goal.
- (Ω, A) a transition system labeled by an alphabet *A*.
- ψ.S$\rightarrow 2^{M}_{fn}$ is a function which associates to every state a finite set of maximal event names present at this state.
- μ: T $\rightarrow 2^{M}_{fn}$ is a function which associates to every transition a finite set of event names corresponding to actions that have started their execution so that their terminations allow the start of this transition. This set corresponds to the direct causes of this transition.
- ζ: T $\rightarrow$ M is a function that associates to its transition an event name identifying its occurrence. Such that for any transition $t \in T$,

$$\mu(t) \subseteq \psi(\alpha(t)),\ \xi(t) \notin \psi(\alpha(t)) - \mu(t) \text{ and}$$
$$\mu(\beta(t)) = (\psi(\alpha(t)) - \mu(t)) \cup \{\xi(t)\}. \qquad \square$$

Among others, an MLTS allows to express the behavior of concurrent systems by the determination of the set of actions that are potentially in execution in every state (see Figure 2).

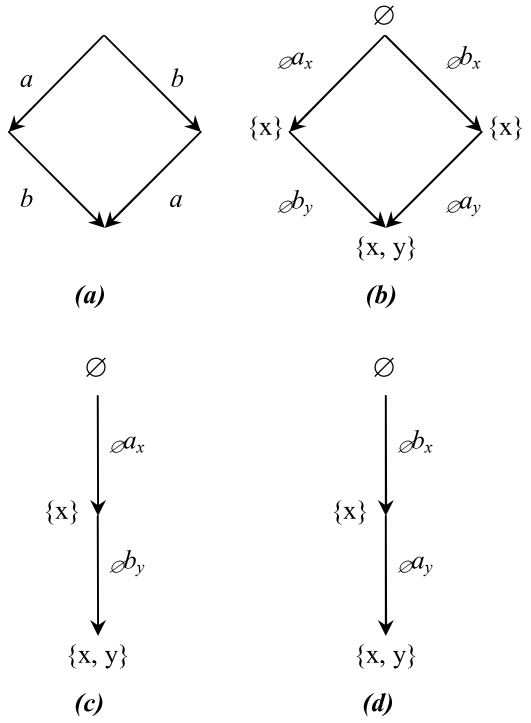

Figure 1. Maximality-based semantics.

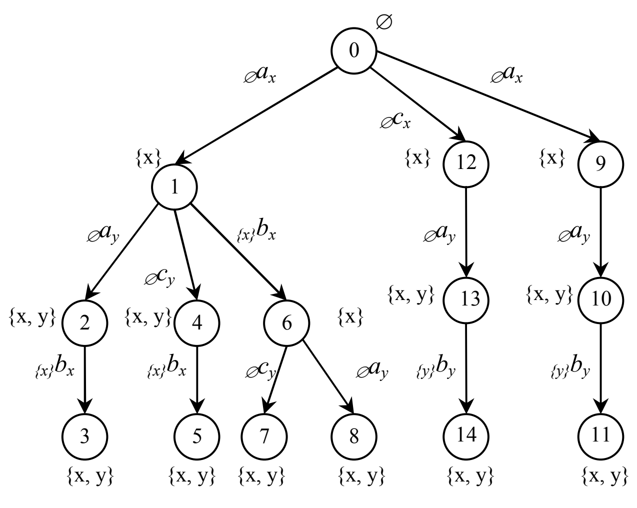

Figure 2. H expression MLTS.

Let H = a; b; stop | | | (c; stop [] a; stop) be a behavior expression. Intuitively, we can notice that during the behavior of such a specification, with the hypothesis that actions are not atomic, in certain states actions (*a* and *c*), (*a* and *a*), (*b* and *c*) as well as *b* and *a* can comply in parallel. It is clear that interleaving semantics does not allow seeing such situations. However, the application of the maximality-based operational semantics of basic LOTOS [9, 21] allows the generation of the MLTS shown in Figure 2. One can notice that states 2, 3, 4, 5, 7, 8, 10, 11, 13, and 14 represent such case of concurrent action executions.

3. MLTS Model Based Logic Verification

CTL is a branching time temporal propositional logic frequently used in the logic verification techniques (model checking) [2, 3, 6, 7, 13]. CTL contains the usual temporal operators: X (the next time), F (possible), G (always), and U (until) whom have to be at once preceded by one of the path quantifiers A (for all paths) and E (there exists a path). For example, AG_p is satisfied in a state if for all paths from this state, *p* is always true.

CTL temporal logic allows expression formulae on states, noted 'φ' and formulae on paths, noted 'ω'. Their syntax is as follows:

$$\varphi ::= p \;|true|\; \neg\varphi \;|\; \varphi \wedge \varphi|\; A\,\omega \;|E\,\omega$$
$$\omega ::= F\,\varphi \;|G\,\varphi \;|X\,\varphi|\;\; \varphi U\,\varphi$$

where $p \in AP$ is an atomic proposition.

We can distinguish eight basic operators in the CTL logic: AX, EX, AG, EG, AF, EF, AU, and EU defined as follows:

- AX*f* is satisfied in a state if the formula *f* is satisfied in all its successors.
- EX*f* is satisfied in a state if at least one of its successors satisfies the formula *f*.
- A state satisfies AF*f* if on every path stemming from this state, there is at least a state that satisfies *f*.
- A state satisfies EF*f* if there is a path stemming from this state containing at least a state that satisfies *f*.
- A state satisfies AG*f* if on all the paths from this state, *f* is always satisfied.
- A state satisfies EG*f* if there is a path stemming from this state where f is always satisfied.
- A[*fUf'*] is satisfied in a given state if on all paths stemming from this state f is always verified until a state which verifies *f'*.
- E[*fUf'*] is satisfied in a given state if there is a path from this state where f is always verified until a state which verifies *f'*.

4. Maximality-Based Semantics and Model Checking

4.1. Kind of Properties to be Verified

In the previous section, it has been showed that properties could be expressed using the CTL temporal logic. Reasoning concerned logical propositions belonging to the system states. The properties subjects of verification are generally divided into two classes, which are the liveness and fairness classes [16]. Obviously, several model checkers, based on interleaving semantics, were developed in the literature. For our part, the use of maximality-based labeled transition system; the information included in the states of the model represents the actions that are potentially in execution. For this fact, one can express belonging properties such as *mutual exclusion* in a more natural way, as well as from new properties that concern actions and their parallel execution. The expression of these properties does not require the use of a new logic or the introduction of new operators since one can use CTL temporal logic and consider actions in states as being atomic formulae. However, what changes is the intuition behind formulae. For example, the formula EF ($p \wedge q$) where *p* and *q* are names of actions means that there is at least a path which leads to a state where parallel execution of *p* and *q* can take place. In a similar way, one can explain intuitively all the formulae of the CTL logic that may be checked using MLTS model as follows:

- $p \wedge q$ in a state *S* means that *p* and *q* can be executed in parallel in the state *S*.
- $\neg p$ in a state S means that the execution of *p* in the state *S* cannot take place.
- EX*p* in a state S_0 means that there is at least a path $(S_0, S_1,...)$ where *p* will be able to comply in the state S_1.
- AX*p* in a state S_0 means that for any path (S_0, S_1, ...) starting from state S_0, *p* may be executed at state S_1.
- E[*p*U*q*] in a state S_0 means that there is a path (S_0, S_1, …, S_k, …) where *q* will be able to comply in the S_k state and *p* will be able to comply in every state of this path that precedes the state S_k.
- A[*p*U*q*] in a state S_0 means that for any path starting from the state S_0, there is a state in this path where *q* may be comply and *p* may be comply in every preceding state.

Therefore, to express fairness or liveness properties, it is not necessary to use logical formulae indicating the state of evolution of a process, we only would reason directly about actions. By proceeding so, properties will be easier to express and their meaning seems more natural. If one takes as example for the mutual exclusion the classical problem of readers and writers, we suppose that there is one reader and one writer who chair one variable. The accesses to this variable must be exclusive, so instead of expressing mutual exclusion by AG¬ (atC$_1$ ∧ atC$_2$), one can express it simply by AG¬(writer_write ∧ reader_read). Where: writer_write is the action of writing on the variable by the writer process; *reader_read* is the action of reading the variable by the reader process. atC$_1$ is a logical formula that is true if the writer process is in its critical section and atC$_2$ is a logical formula that is true if the reader process is in its critical section.

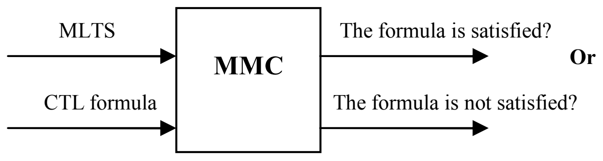

Figure 3. Outside sight of the MMC procedure.

In section 5, more elaborated examples will be presented by a detailed study of philosophers dinner paradigm.

Remark1: It may be seen in the MLTS structure that every action is associated with an event name that allows distinguishing between several parallel executions of the same action at any state (auto-concurrency). Considering this point will allow us to reason about the number of parallel execution of an action at any state, in other words, we may verify the degree of the auto-concurrency in a system. □

For instance *p*: 5 expresses the fact that there is five parallel execution of the action *p*. *p*: 5 will be so considered as being an atomic proposition; what will avoid the introduction of a new operator in the considered logic. One will have then two forms of atomic propositions, the form *p* or some *p: n* where *n* is a positive natural number.

Considering these intuitive aspects and previous notations, one can express new properties such as:

- *Specifying actions incompatibility*: We may express that *a* and *b* are incompatible by AG¬ ($a \wedge b$) which means that they will never be able to be executed concurrently. In a similar way, to verify that actions can be executed concurrently may be expresses by EF¬ ($a \wedge b \wedge \ldots \wedge z$), where *a*, *b*, …, and *z* are action names.
- *Specifying auto-concurrency level*: For instance EF (a: n) is true if there is a state in which *n* actions of name *a* may be in execution simultaneously. It is obvious that such properties cannot be expressed using interleaving models.

4.2. Model Checking Algorithm

Having seen the kind of properties which one could express by means of the MLTS model for possible behavior representation and the temporal logic CTL as specification language of properties, in what follows we illustrate the evaluation method of CTL formulae on the MLTS model through the adaptation of model checking algorthim presented in [7]. The choice of this algorithm is made just as an illustration, it is clear that more impressive model checking algorithms can be adapted to MLTS model in a similar way.

4.2.1. Algorithm Behavior

Let us suppose that one has a finite structure (model) M = (S, R, L) and a CTL formula p_0. The purpose is to determine states s of M where $M, s \models p_0$.
This algorithm is conceived to be executed in different steps:

- First step deals with all p_0 sub-formulae of length 1.
- Second step deals with all p_0 sub-formulae of length 2 and so on.

At the end of the i^{th} step, every state will be labeled by the set of all true sub-formulae of length *i* at this state. To elaborate labeling in step *i*, one needs collected information in previous steps. For example, the state *s* must be labeled with sub-formula ($q \wedge r$) exactly if the state *s* is labeled by *q* and *r*.

For the sub-formula A[qUr], one will need information about successor states of *s* as well as on the state s itself, because A[qUr] = r ∨ (q ∧ AXA[qUr]). Initially, A[qUr] is added to all the already labeled states by r. Then, A[qUr] will be propagated and added to any state labeled by q having all successor labeled by A[qUr].

In the same way one may argues for E[qUr].
It may be noted that the other model operators are implicit and defined as much as the following abbreviations:

$$q \vee r \equiv \neg(\neg q \wedge \neg r)$$
$$q \Rightarrow r \equiv \neg q \vee r$$
$$q \Leftrightarrow r \equiv (q \Rightarrow r) \wedge (r \Rightarrow q)$$
$$\mathrm{AX}q \equiv \neg\mathrm{EX}\neg q$$
$$\mathrm{EF}q \equiv \mathrm{E}[\mathrm{true}\,\mathrm{U}q]$$
$$\mathrm{AG}q \equiv \neg\mathrm{EF}\neg q$$
$$\mathrm{AF}q \equiv \mathrm{A}[\mathrm{true}\,\mathrm{U}q]$$
$$\mathrm{EG}q \equiv \neg\mathrm{AF}\neg q$$

Algorithm

Input: A temporal structure M = (S, R, L) as semantic model and a formula p_0 written in CTL.
Output: Set of states of M satisfying p_0 Formula.

```
begin
for i = 1 to length ($p_0$) do
for all sub-formula p of $p_0$ of length i do
case form of p do
  p = P: atomic proposition:
    /* nothing to do */
  p = $q \wedge r$:
    for all s $\in$ S do
      if q $\in$ L (s) and r $\in$ L (s) then
        add ($q \wedge r$) to L (s);
        end if
  p = ¬q:
    for all s ∈ S do
      if q ∉ L (s) then
        add ¬q to L (s);
      end if
  p = EXq:
    for all s ∈ S do
      if ∃ successor s' of s / q ∈ L (s') then
        add EXq to L (s);
      end if
  p = A[qUr]:
    for all s ∈ S do
      if r ∈ L (s) then
        add A[qUr] to L (s);
      end if
    end for
    for j = 1 to Card (S) do
      for all s ∈ S do
        if q ∈ L (s) and if ∀ successor s' of s /
          A[qUr] ∈ L (s') then
          add A[qUr] to L (s);
        end if
      end for
    end for
  p = E[qUr]:
    for all s ∈ S do
      if r ∈ L (s) then
        add E[qUr] to L (s);
      end if
    end for
    for j = 1 to Card (S) do
      for all s ∈ S do
        if q ∈ L (s) and if ∃ successor s' of s /
          E[qUr] ∈ L (s') then
          add E[qUr] to L (s);
        end if
      end for
    end for
end case
end for
end for
end
```

This algorithm version has a linear temporal complexity according to the length of formula to be verified and quadratic complexity according to the size of *M* structure [13].

Choosing CTL logic as properties specification language, the fact that the interesting information in MLTS structure is encapsulated into state leads us to adapt this algorithm to our study for properties verification on MLTS model by considering actions at MLTS states as atomic propositions and taking into account the case where *p* has the form *q*: *n* as follows:

```
case p = q:n,
for all s ∈ S do
  if ∃ $x_1, x_2, ..., x_n$ / $x_1, x_2, ..., x_n$ ∈ L (s)
    and act ($x_1$, s) = q, act ($x_2$,s) = q, ..., act ($x_n$, s) = q
    and card ({$x_1, x_2, ..., x_n$}) = n then
      add q:n to L (s);
  end if
end for
```

Where:

x_1, x_2, x_n: are event names.
q: is an action.
act: is a function that has in input an event name and a state, it returns the name of action associated to the event given as parameter in this state.

5. Examples

In this section, we present the study of philosophers' dinner problem. The formal LOTOS specification of this problem can be found in [22].

5.1. Mutual Exclusive

Mutual exclusion concerns the use of every fork (not shared resource). For two philosophers, property is expressed by:

AG (not (*Philo1TakeF1* and *Philo2TakeF1*) and not (*Philo1TakeF2* and *Philo2TakeF2*)).

We have verified the case of two, three and four philosophers by using our tool, and we had as result the property satisfaction on all case studies.

5.2. Deadlock Free

Deadlock situation in our example occurs when each philosopher (a process) have a fork and asks for another fork already allocated to another philosopher. This situation leads to a circular waiting situation. Deadlock free situation means that the system may always progress in the future at each state. This property is expressed by: AG ((EX true) or *delta*).

This CTL expression means that every state of the system will have at least a successor state. Should the opposite occur, it is necessary that the last action which is potentially in execution in this state, according to the maximality-based semantics, is the action 'δ' (delta) which means successful process ending. Obtained results are:

- At first, we wrote specifications satisfying the hypothesis that forks are taken in the same order (elimination of one of deadlock necessary conditions). This result is confirmed with the tool by the satisfaction verification of the previous formula.
- Secondly, we specified the resources request in any order. In that case, the formula is not verified, and deadlock states were detected [22].

5.3. Starvation Free

Starvation free is expressed by the fact that at each time when a philosopher wants to eat then he will be able to do it in the future. For a philosopher *i*, starvation free is expressed by a CTL formula AG (*Philo_i_WantEating* $\Rightarrow$ AF *Philo_i_Eat*).

6. Conclusion

In this paper, we discussed the interest of maximality-based labeled transition system model for the verification of concurrency properties. This contribution was mainly led by the presence of maximality information in states and transitions. In particular, we showed how this information makes easy writing atomic propositions expressing properties to be verified. A particular attention was payed to the natural and intuitive reading of these properties. As an example, we underlined the verification of the concurrency level in system behavior. With the aim of giving concrete expression to this study, we showed how a classic model checking algorithm [7, 13], can be adapted easily to the maximality-based labeled transition system model. Concerning this study, we adopted a global evaluation approach.

To clarify ideas, we applied our results on the philosophers' dinner problem. Outside this work, it is to note that we applied our approach for the study of concrete applications of communication protocols domain, as example we resumed the study of the transport service presented in [5]. The validation of results was realized by the use of the FOrmal COncurrency Verification Environment (FOCOVE).

As perspective of this work, we can intend to spread our approach to local evaluation methods (one-the-fly) and the complexity study of model checking algorithms being able to be developed. Because the maximality-based labeled transition system model is a true concurrency model, we can inherit results of works aiming to resolve the state-graph combinatorial explosion problem by the use of such semantics [4, 11, 25, 26, 19, 20]. Finally, it is to note that interesting results have been developed for the definition of symbolic verification methods based on the MLTS model [24] that we are hoped to be applicable on real-time models [23].

References

[1] Arnold A., *Systèmes de Transitions Finis et Sémantique des Processus Communicants*, Masson, Paris, 1992.

[2] Ben-Ari M., Manna Z., and Pnueli A., "The Temporal Logic of Branching Time," *Acta Informatica*, vol. 20, pp. 207-226, 1983.

[3] Bernard B., Schnoebelen Ph., Bidoit M., Laroussinie F., and Petit A., *Vérification de Logiciels: Techniques et Outils du Model Checking*, Vuibert Editions, Paris, 1999.

[4] Berthomieu B. and Vernadat F., "State Class Constructions for Branching Analysis of Time Petri Nets," *Technical Report 02130*, LAAS-CNRS, 7 Avenue du colonel Roche, 31077, Toulouse Cedex, France, 2002.

[5] Bolognesi T. and Brinksma E., "Introduction to the ISO Specification Language LOTOS," *Computer Networks and ISDN Systems*, vol. 14, pp. 25-59, 1987.

[6] Clarke E. M. and Emerson E. A., "Design and Synthesis of Synchronization Skeletons Using Branching Time Temporal Logic," *in Proceedings of IBM Workshop on Logics of Programs*, vol. 131, pp. 52 -71, Springer-Verlag, 1981.

[7] Clarke E. M., Emerson E. A., and Sistla A. P., "Automatic Verification of Finite State Concurrent Systems Using Temporal Logic Specifications," *ACM Transactions on Programming Languages and Systems*, vol. 8, no. 2, pp. 244-263, April 1986.

[8] Coelho da Costa R. J. and Courtiat J. P., "A True Concurrency Semantics for LOTOS," *in Proceedings of Formal Description Techniques (FORTE'92)*, North-Holland, 1993.

[9] Courtiat J. P. and Saidouni D. E., "Relating Maximality-Based Semantics to Action Refinement in Process Algebras," *in Proceedings of the 7th International Conference on Formal Description Techniques (FORTE'94)*, in Hogrefe D. and Leue S. (Eds), pp. 293-308, Chapman & Hall, 1995.

[10] Darondeau P. and Degano P., "Causal Trees," *in Proceedings of ICALPS'89*, vol. 372, pp. 234-248, Springer-Verlag, 1989.

[11] De Souza M. L., "*Application des Méthodes d'ordre Partiel aux équivalences Comportementales des Systèmes Concurrents*," *PhD Thesis*, INRIA Sophia-Antipolis, France, December 1995.

[12] Devillers R., "Maximality Preservation and ST-idea for Action Refinement," in Rozenberg G. (Ed), *Advances in Petri Nets*, vol. 609, pp. 108-151, Springer-Verlag, 1992.

[13] Emerson E. A., *Temporal and Modal Logic*, in van Leeuwen J. (Ed), Handbook of Theoretical Computer Science: Formal Models and Semantics, Ch. 16, vol. B, pp. 995-1072, Elsevier, 1990.

[14] Emerson E. A. and Lei C. L., "Efficient model Checking in Fragments of the Propositional Mu-Calculus," *in proceedings of the 1st LICS*, pp. 267-278, 1986.

[15] ISO/IEC, *LOTOS*, *A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour*, *International Standard*

8807, International Organization of Standardization, Information Processing Systems, Open Systems Interconnection, Geneva, September 1988.

[16] Lamport L., "What Good in Temporal Logic?," *Information Processing Letters*, vol. 83, pp. 657-668, 1983.

[17] Langerak R., "Bundle Event Structures: A Non-Interleaving Semantics for LOTOS," in Diaz M. and Groz R. (Eds), *in proceedings of FORTE'92*, pp. 331-346, North-Holland, 1993.

[18] McDermid J., "Formal Methods: Use and Relevance for the Development of Safety Critical Systems," in Bernet P. (Ed), *Safety Aspects of Computer Control*, Butterworth/Heineman, 1991.

[19] Peled D., "Combining Partial Order Reductions with on-The-Fly Model Checking," *in Proceedings of CAV'92*, vol. 818, Springer-Verlag, 1994.

[20] Ribet P., Vernadat F., and Berthomieu B., "On Combining the Persistent Sets Method with the Covering Steps Graph Method," *Technical Report 02388*, LAAS-CNRS, 7 Avenue du colonel Roche, 31077, Toulouse Cedex, France, 2002.

[21] Saïdouni D. E., "*Sémantique de maximalité: Application au Raffinement d'actions en LOTOS*," *PhD Thesis*, LAAS-CNRS, 7 Avenue du colonel Roche, 31077, Toulouse Cedex, France, 1996.

[22] Saïdouni D. E. and Belala N., "Vérification de Propriétés Exprimées en CTL sur le Modèle des Systèmes de Transitions étiquetées Maximales," *Research Report*, Laboratoire LIRE, Université Mentouri, 25000 Constantine, Algérie, 2003.

[23] Saïdouni D. E. and Courtiat J. P., "Prise en Compte des Durées d'action Dans les Algèbres de Processus par l'utilisation de la Sémantique de Maximalité," *in Proceeding of Colloque Francophone sur l'Ingénierie des Protocoles (CFIP'2003)*, Hermes, Paris, 2003.

[24] Saïdouni D. E. and Labbani O., "Maximality-Based Symbolic Model Checking," *in Proceedings of ACS/IEEE International Conference on Computer Systems and Applications*, Tunisia, July 2003.

[25] Valmari A., "A Stubborn Attack on State Explosion," *in Proceedings of CAV'90*, DIMACS, ACM, 1990.

[26] Wolper P. and Godefroid M. P., "Partial-order Methods for Temporal Verification," *in Proceedings of CONCUR'93*, vol. 715, Springer-Verlag, 1993.

Djamel Eddine Saidouni obtained his BEng degree from University of Mentouri Constantine, Algeria, in 1990. After that, he joined the LAAS/CNRS Laboratory, Toulouse, France where he prepared his DEA in communicating systems. He obtained his PhD in theoretical computer science and concurrency from the University of Paul Sabatier, Toulouse, France in 1996. Currently, he is a permanent researcher in computer science at the LIRE Laboratory. His main research interest is formal specification and verification of complex distributed and real time systems.

Nabil Belala obtained his BEng degree from University of Mentouri Constantine, Algeria, in June 2002. Currently, he prepares his MSc degree in computer science at the LIRE Laboratory of the University of Mentouri Constantine. His research interest is formal specification and verification of real-time systems.