MAXIMALITY SEMANTIC FOR RECURSIVE PETRI NETS

Djamel-Eddine Saïdouni
Departement of computer science
University of Mentouri
25000, Constantine, Algeria
E-mail: saidouni@misc-umc.org

Messaouda Bouneb
Departement of mathematic and computer science
University of El Arbi ben M'hidi
Oum el boighi, Algeria
E-mail: bounebmessaouda@hotmail.com

Jean-Michel Ilié
Departement of computer science
University of Pierre and Marie Curie
75005, Paris, France
Email: jean-michel.ilie@upmc.fr

KEYWORDS
Formal methods, Dynamic systems, Recursive Petri nets, Maximality semantics.

ABSTRACT

This paper is in the framework of the specification and the verification of concurrent dynamic systems. We are interested by recursive Petri net specification model for which we define a maximality semantics. The underlying semantic model is a maximality-based labeled transition system. For this purpose, we propose a maximality operational semantic for recursive Petri nets. As an illustration, a system of filling medical bottles is specified in terms of recursive Petri net and translated to a maximality-based labeled transition system. This later is used to check the system properties. The properties are expressed using the CTL logic and verified by means of the FOCOVE tool.

INTRODUCTION

A Petri net is both a graphical and mathematical representation, used to formally specify the behaviors of concurrent systems. The marking graph associated with a given Petri net is used for checking the expected properties of the system. Indeed this marking graph is seen as a labeled transition system. However labeled transition systems are based on interleaving semantics. This later represents parallel executions as their interleaved sequential executions. To clarify the idea, we consider the example of two Petri nets (Figures 1.(a) and 1.(b)).

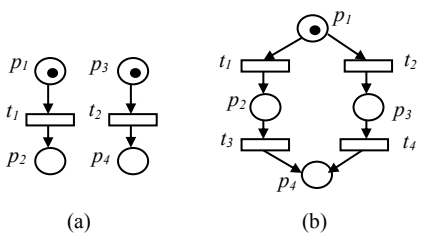


Figure 1: Petri nets.

Figure 1.(a) represents a system which can execute transitions t_1 and t_2 in parallel, whereas Figure 1.(b) represents a system that execute sequentially, either the transitions t_1 and t_3, or the transitions t_2 and t_4.
The marking graphs of the two Petri nets are given respectively by the labeled transition systems (LTS) of Figures 2.(a) and 2.(b). If both transitions t_1 and t_4 are labeled by the action a and t_2 and t_3 by b, then the two marking graphs are isomorphic. Therefore, the concurrent execution of the actions a and b is interpreted as their interleaved execution in time.

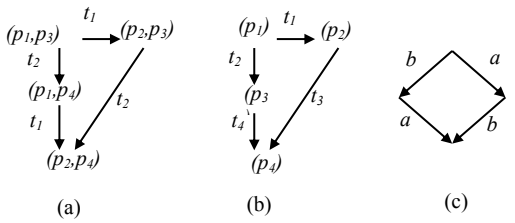


Figure 2: interleaving Semantics

This result is acceptable under the assumption that the firing of each transition corresponds to the execution of an indivisible action with null duration (structural and temporal atomicity of actions). Nevertheless, this assumption is often not realistic in practice.

Taking into account non atomicity of actions in a system has been deeply studied in the literature through the definition of several semantics supporting the concept of action refinement, e.g. L. Aceto and M. Hennessy 1991) (E. Best and al. 1991) (G. Boudol and I. Castellani 1988) (J.P.Courtiat and D.E. Saïdouni 1994) (J.P. Courtiat and D.E. Saïdouni 1995) (P. Darondeau and P. Degano 1989) (P. Darondeau and P. Degano 1991) (P. Darondeau and P. Degano 1993) (P. Degano and R. Gorrieri 1991) (R. Devillers 1992a) (R. Devillers 1992b) (R. Devillers 1993)(E.W. Dijkstra 1971) (W. Janssen and al. 1991) (D.E. Saïdouni 1996) (J.R. van Glabbeek 1990).

As a first advantage, action refinement allows a hierarchical design of systems. A second interest is the ability to semantically characterize concurrent executions of non-instantaneous actions. In this context, the maximality semantic was exploited to specify concurrent systems, through the model of the maximality labeled transition systems. This semantic

Proceedings 27th European Conference on Modelling and Simulation ©ECMS Webjørn Rekdalsbakken, Robin T. Bye, Houxiang Zhang (Editors)
ISBN: 978-0-9564944-6-7 / ISBN: 978-0-9564944-7-4 (CD)

was defined for several models of specifications, including some process algebras and place transition Petri nets (D.E. Saidouni and al. 2008a) (D.E. Saidouni and al. 2008b) (D.E. Saidouni and al. 2009a) (D.E. Saidouni and al. 2009b).

However, the limits of Petri nets have been highlighted when modeling systems with dynamic structures, such as multi agent systems. Therefore, Petri nets were extended to recursive Petri nets and dynamic behaviors are considered through a new kind of transitions, namely an abstract transition. The firing of such a transition represents the execution of a thread. The behavior of any thread is modeled by the recursive Petri net. As abstract transitions can represent non atomic activities, true concurrency semantics appears to be more appropriate than interleaving one. Abstract transitions can be used to design the dynamic system hierarchically.

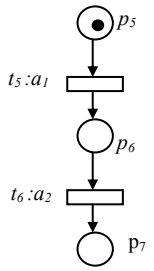


Figure 3: Refinement of abstract transitions.

Consider the two nets of Figure 1 again but assume now that t_1 and t_4 are abstract transitions in order to model the run of complex actions. For sake of simplicity, the behaviors associated with these two transitions accords with the same description, specified by the Petri net of Figure 3, where the transitions t_5 and t_6 are labeled by actions a_1 and a_2 respectively. Any firing of an abstract transition is assumed to create a new thread, whose execution runs from a token in the place p_5 and which can terminate when a token reaches the place p_7. By applying the marking graph generation method from both nets (e.g. in D. Dahmani 2009), we obtain their corresponding labeled systems, highlighted in Figures 4.(a) and 4.(b), respectively.

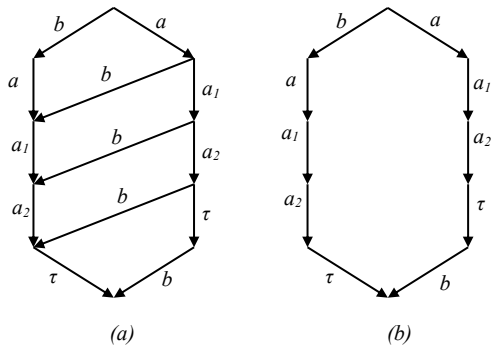


Figure 4: LTS refining the abstract transitions.

At this stage, observe that, the parallelism in the first system is expressed by the interleaved execution of the action *b* and the thread behavioral description related to the action *a*. Due to the differences between the two LTS representations, this example shows clearly that a hierarchical design of systems should be considered under a true concurrency semantic. Actually, two equivalent systems remain equivalent after refining a same action by a same process. Moreover, in the context of recursive Petri net, the interleaving semantic contradicts the fact that abstract transitions model activities (non-atomic actions).

For this purpose we propose a maximality operational semantic for recursive Petri nets. This operational semantic translates any recursive Petri net to a maximality-based labeled transition system. This allows the formal verification of recursive Petri nets. In fact we can use existing approaches and tools which operate on maximality-based labeled transition systems.

RECURSIVE PETRI NETS

Recursive Petri Nets (RPN) has been proposed for the specification and analysis of dynamic systems (S. Haddad and D. Poitrenaud, 2007) (S. Haddad and D. Poitrenaud, 1999) (D. Dahmani and al, 2009) (D. Dahmani and al, 2008).
Formally, recursive Petri nets are defined as follows. The standard definition is extended to associated actions to transitions.

Definition 1 : A recursive Petri net is defined by $R=(P,T,I,W^-,W^+,\Omega,\gamma,K)$ such that:

- P is a finite set of places
- T is a finite set of transitions such that $P \cap T=\emptyset$. It is composed of a disjoin sets of elementary transitions T_{el}, and abstract transitions T_{ab}.
- $I=I_C \cup I_P$ is a finite set of indexes, indicates the cut steps and preemptions.
- $W^-: P\times T\rightarrow \mathbb{N}$ is the precondition matrix.
- W^+: $Px[T_{el} \cup (T_{ab}\times I)\rightarrow \mathbb{N}]$ is the post-condition matrix.
- $\Omega: T_{ab}\rightarrow \mathbb{N}^P$ a function which associates to each abstract transition an ordinary marking (starting marking).
- γ is a family indexed by the set of termination I_C. Each set is specified as an effective representation of semi linear set of final markings.
- $K: T_{el}\times T_{ab}\rightarrow I_P$ a partial function of control which allows the modeling of external preemption.

Definition 2 : A labeled recursive Petri net is a pair $\sum=(R,\lambda)$ where $R=(P,T,I,W^-,W^+,\Omega,\gamma,K)$ is an RPN and $\lambda: T\rightarrow A$ is the action mapping of transitions.

MAXIMALITY BASED TRANSITIONS SYSTEMS

Let Ev be a countable set of event names and A an alphabet of actions.

Definition 3 : A maximality-based labeled transition system defined over Ev is a 5-uplet $(\Omega,\lambda,\mu,\xi,\psi)$ where $\Omega=<S,T,\alpha,\beta,s_0>$ is a system of transitions, such that:

- S is the set of possible states for the system; this set can be finite or infinite.
- T is the set of transitions or changes between states; this set can be finite or infinite.
- α and β are two mappings from T to *S* s.t. for any transition t, we have: $\alpha(t)$ is the source of T and $\beta(t)$ its destination.
- s_0 is the initial state of the transition system Ω.
- (Ω,λ) is a system of transitions wherein each transition is labeled by an action of A, an occurrence of which must be started ($\lambda:T\rightarrow A$).
- $\psi:S\rightarrow 2^{Ev}$ associates with each state, a finite set of maximal event names, related to the actions
- $\xi:T\rightarrow Ev$ associates with each transition, the event name, identifying a new occurrence of action to be being started.
- $\mu:T\rightarrow 2^{Ev}$ associates with each transition, a finite set of event names corresponding to the actions to terminate in order to process the transition.

We have $\psi(s_0)=\emptyset$ and for each transition *t*, $\mu(t)\subseteq\psi(\alpha(t))$, $\xi(t)\notin\psi(\alpha(t))-\mu(t)$ and $\psi(\beta(t))=(\psi(\alpha(t))-\mu(t))\cup\{\xi(t)\}$.

Notation : Let mlts=$(\Omega,\lambda,\mu,\xi,\psi)$ be a maximality-based labeled transition system such that $\Omega=<S,T,\alpha,\beta,s_0>$. Any transition t is denoted $s \xrightarrow{E^{a}_{x}} s'$, t∈T is a transition such that $\alpha(t)=s$, $\beta(t)=s'$, $\lambda(t)=a$, $\mu(t)=E$ and $\xi(t)=x$. For sake of concision, E^{a}_{x} is also noted Ea:x.

MAXIMALITY SEMANTIC FOR PLACE TRANSITIONS PETRI NETS

In this section we recall the maximality semantic of place transition Petri nets, proposed in (D.E. Saidouni and al 2008a). Within the marking graph:

- each place marking in a state is composed of two disjoint parts. The FT part contains free tokens while the BT part contains bound tokens . Therefore each place is marked by a pair (FT, BT).
- each state change (transition) corresponds to the start of execution for an action and is identified by an event name.
- each bound token identifies an action that is eventually being executed (this token corresponds to a maximal event).

Preliminary definitions

Let (P,T,W^-,W^+) be a Petri net and M one of its marking.

- The set of maximal event names in M is the set of all event names that can be used to identify the bound tokens in a marking. Formally, the function ψ is used to compute this set **:**
  $\psi(M)=\cup_{p\in P}\cup_{i=1,\ldots,mp} x_i$,
  assuming that, for all p in P, M(p)=(FT,BT) with $BT = \{(n_1,a_1,x_1),\ldots,(n_{mp},a_{mp},x_{mp})\}$.
- Let E⊂Ev be a non-empty and finite set of event names, the function makefree(E,M) is defined to free the bound tokens of a set E from a marking M, as follows:
  - $makefree(\{x_1,x_2,..,x_n\},M)=makefree(\{x_2,..,x_n\},makefree(\{x_1\},M))$
  - $makefree(\{x\},M)= M'$ such that for all p∈P, considering M(p)=(FT,BT), then:
    - If there is a bound token (n,a,x)∈BT in p then M'(p)= (FT+n, BT-{(n,a,x)})
    - Otherwise, M'(p) = M(p).
- A transition t of T t is enabled in a marking M iff $|M(p)| \geq W^-(p,t)$ for all p∈P. The set of all the transitions enabled in M is denoted *enabled(M)*.
- The marking M is said to be minimal for the firing of the transition t iff $|M(p)|=W^-(p,t)$ for all p∈P.
- Let M_1 and M_2 be two markings of the Petri net (P,T,W^-,W^+) and consider for any p of P that $M_1(p)=(FT_1,BT_1)$ and $M_2(p)=(FT_2,BT_2)$. We have $M_1\leq M_2$ iff $FT_1 \leq FT_2$ and $BT_1\leq BT_2$, such that the relation ≤ is extended to sets of bound tokens as follows: $BT_1\leq BT_2$ iff $\forall(n_1,a,x)\in BT_1$, $\exists(n_2,a,x)\in BT_2$ such that $n_1\leq n_2$.
- Let M_1 and M_2 be two markings of the Petri net (P,T,W^-,W^+) such that $M_1\leq M_2$. The difference M_2-M_1 is a marking M_3 (M_2-M_1=M_3) such that for all p∈P, if $M_1(p)=(FT_1,BT_1)$ and $M_2(p)=(FT_2,BT_2)$ then $M_3(p)= (FT_3,BT_3)$ with $FT_3=FT_2-FT_1$ and $\forall(n_1,a,x)\in BT_1$, $(n_2,a,x)\in BT_2$, if $n_1\neq n_2$ then $(n_2-n_1,a,x)\in BT_3$.
- The function *get*:$2^{Ev}-\{\emptyset\}\rightarrow Ev$ is a function which satisfies get(E)∈E for any $E\in 2^{Ev}-\{\emptyset\}$.
- Given a marking M, a transition t and an event name x s.t. $x\notin\psi(M)$, the function occur(t,x,M)=M', assuming M(p)=(FT,BT), and M'(p)=(FT,BT') for all p∈p, is such that $BT' = BT \cup \{W^+(p,t),\lambda(t),x)\}$ if $W^+(p,t)\neq 0$ and BT'=BT otherwise. Hence, M' augments M with new bound tokens w.r.t. t and x.

MAXIMALITY SEMANTIC FOR RECURSIVE PETRI NETS

Let us first explain the proposed approach through simple examples.

A- Start and end of abstract transition firings

Consider the recursive Petri net of Figure 5 where t_2 is an abstract transition, the firing of which represents the execution of the action *b*. The firing of t_2 is a consequence of the end of execution related to the action *a,* attached to the transition t_1. The firing of this abstract transition starts the execution of a "son" thread, in addition to the initial thread. Both act concurrently on recursive Petri net, but with a distinct marking. The ordinary marking attached to the abstract transition is used to initialize the marking of the son Petri net. This is interpreted by the production of a token in the place p_5. The creation of a son Petri net from a thread is

represented by the firing of a virtual transition called *admitted*, here interpreted as the start of the action *b* attached to the abstract transition at the father thread level. The start of execution of the action *admitted(b)* is identified by the event *x*. This firing is similar to the firing of an elementary transition; it is followed by the production of a bound token related to this action in the BT part of the place p_5.

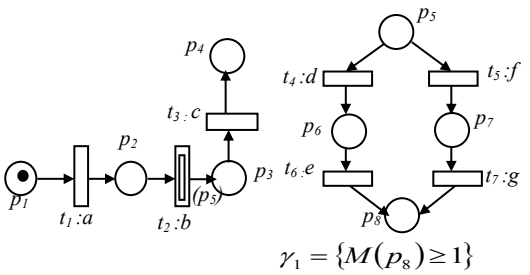


Figure 5: Recursive Petri net

After the generation of a bound token in the place p_5, any transition that can be fired from this thread will be immediately executed. But it is necessary to take into account the satisfaction of the termination predicate $\gamma=\{|M(p_8)|\geq 1\}$. This condition holds when either one of the transition t_6 or t_7 produces a token in the right part of the place p_8. When this predicate becomes true, a transition called *finished* can fire, which makes the return to the father thread, indeed this transition represents the cut step τ of the son thread.

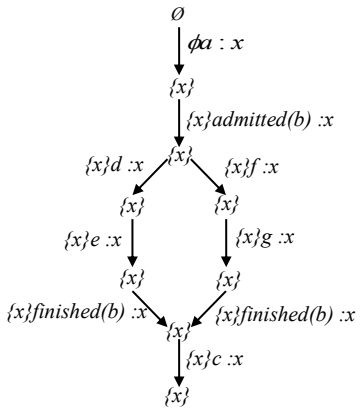


Figure 6: Start and end of an abstract transition

Generally, a transition *finished* is viewed as an elementary transition. Its firing causes the production of tokens in the BT part of all the places which belong to the post set of the abstract transition. Just after the end of the execution of abstract transition, the firing of the transition t_3 may happen. Figure 6 represents, the maximality labeled transitions system generated from this Petri net. Note that the event *x* identifies the action *admitted(b)* as well as the start of execution of the thread itself, thus it can be re-used within this thread. Once the son thread is finished, this event name can be re-used in the father thread.

Semantic of sequencing

Abstract transitions extend the relation of causality to the refinement of threads. Actually, the terminations of a thread can condition the start of another thread. For example in the Petri net of Figure 7, the activities of t_2 causally depends on the end of the activity of t_1.

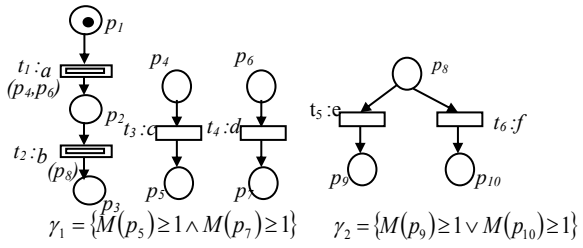


Figure 7: Sequence of activities.

Figure 8 represents the maximality labeled transitions system obtained by applying the maximality approach.

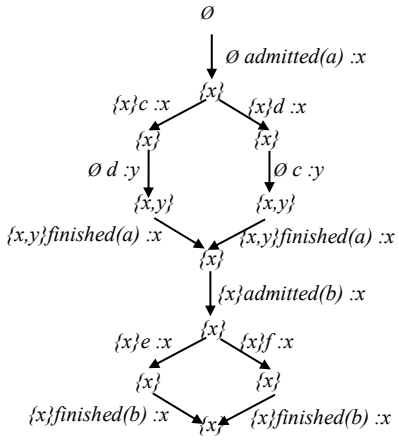


Figure 8: Sequencing and re-use of event names.

Semantic of parallelism

Consider now the recursive Petri net of Figure 9 where t_1 is an abstract transition. The behaviors associated with the two firing occurrences of this transition can be executed concurrently in parallel.

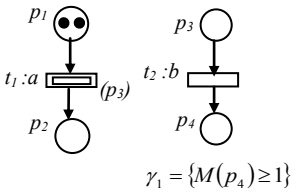


Figure 9: Parallelism of the abstract transitions.

The desired semantic must transform this relation of independence between two firing occurrences of the abstract transition into a parallel execution of two thread activities. The obtained maximality labeled transitions system is represented by Figure 10. The two firing occurrences of the thread, corresponding to the transition t_1, are identified by the event names *x* and *y*.

The set *{x,y}* means that the two corresponding activities implied by t_1 and t_2 may be in parallel execution, unless to be finished.

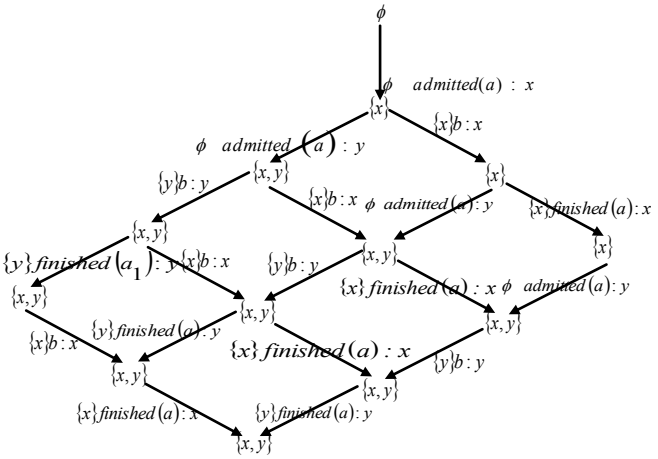
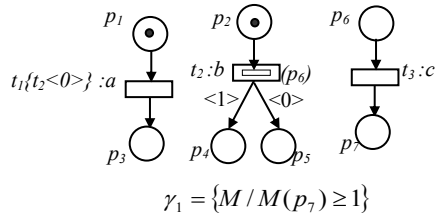


Figure 10 : Semantic of parallelism between threads.

Semantic of preemption

Consider the recursive Petri net of Figure 11. In the initial state of the system, observe that the free token in p_1 enables the firing of the elementary transitions t_1. In the same way the free token in p_2 enables the firing of the abstract transition t_2. The mentioned notation $t_1\{t_2<0>\}$ specifies that each firing of the elementary transition t_1 ends all the thread activities caused by some firings of t_2 (preemption concept).



$$\gamma_1 = \{M / M(p_7) \geq 1\}$$

Figure 11 : Modeling of preemption.

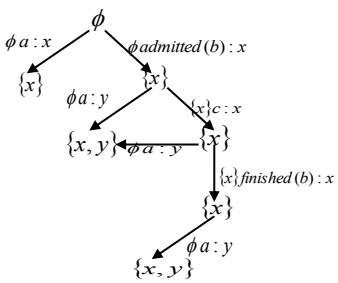


Figure 12 : Semantic scheme of a preemption.

From the initial state, three scenarios are possible, which are summarized in the maximality labeled transitions system of Figure 12. Observe that in all the cases, the firing of t_1 ends the execution of the thread corresponding to the transition t_2. In the first state, labeled by the set *{x,y}*, the event *y* identifies the start of execution of the transition t_1 however the event *x* represents the termination of the abstract transition t_2.

Operational semantic for labeled recursive Petri nets

In this section, we consider a labeled recursive Petri net $R=(S,T,I, W^-, W^+,\Omega,\gamma,K,\lambda)$ and different markings M, M_i, … of R.

Preliminary definitions.

1. We call thread configuration any pair of the form $(TH,(M)^e_{Ta})$ such that :

– M : is a marking of R
– e is an event name
– Ta is an abstract transition
– *TH* is a set of thread configurations.

Please, note that a thread configuration $(TH,(M)^e_{Ta})$ generally defines a tree of threads linked by a childhood relationship, the root of which is marked by M and is created under *e* by firing Ta. Let *THs* be the set of all the possible thread configurations for R.

– The initial thread configuration is built from the initial marking of R, e.g. M_0. It is denoted $(TH,(M_0)^{e_0}_{Ta})$. For sake of homogeneity, an extra event, namely $e_0 \notin$ Ev, is introduced to virtually launch the initial thread.

Moreover, according to any thread configuration $(TH,(M)^e_{Ta})$, let us introduce the following mappings:

- ψ:THs→2^{Ev} is the mapping which yields all the event names referred in a thread configuration and its descents. It is recursively defined by:
  - ✓ $\psi(\emptyset,(M)^e_{Ta})=\psi(M)$
  - ✓ $\psi(TH,(M)^e_{Ta})=\left(\cup_{th\in TH}\psi(th)\right)\cup\psi(M)$
- makefree: 2^{Ev} x THs → THs is used to free bound token within a thread configuration. It is recursively defined by:
  - ✓ $\text{makefree}\left(E,(\emptyset,(M)^e_{Ta})\right)=\left(\emptyset,\left(makefree(E,M)\right)^e_{Ta}\right)$
  - ✓ $\text{makefree}\left(E,(TH,(M)^e)\right)=$
$$\left(\bigcup_{th\in TH} makefree\,(E,th), makefree(E,M)\right)^e_{Ta}$$
- The enabling test of transitions is standard. The set *min(M,t)* denotes the set of all possible minimal markings built from M that enable the transition t. To deal with cut steps, let the mapping *cutstep* be s.t.
  - ✓ $\text{cutstep}\left((TH,(M)^e_{Ta}),\gamma_j\right)$ is true iff
$$\forall p\in P,|M(p)|\geq\gamma_j(p)$$
- The production of tokens specified by the mapping *occur* must be adapted to deal with different cases of firings, elementary and abstract transitions (see below in the semantic rules).

Semantic rules : The following four semantic rules allow one to create the maximality labeled transitions system of any labeled recursive Petri net, automatically.

1. $\dfrac{M,t\in enabled(M)\wedge t\in T_{el}}{(TH,(M)^{e}_{Ta})\xrightarrow{E^{\lambda(t)}x}(TH,(M')^{e}_{Ta})}$ such that:

$$\begin{cases} * \ \forall M'' \in \min(M,t), E=\psi(M''), \\ \qquad M''' = makefree(E, M-M'') \\ * \ M' = occur(t,x,M''') \text{ such that:} \\ \quad \forall p\in P, \text{if } M'''(p)=(FT''',BT''') \text{ and} \\ \qquad M'(p)=(FT',BT') \text{ then} \\ FT' = FT''' \text{ and} \\ BT' = \begin{cases} BT'''\cup\{(W^{+}(p,t),\lambda(t),x)\} \text{if } W^{+}(p,t)\neq 0 \\ \text{and} \\ BT''' \text{ otherwise} \end{cases} \\ * \ x = get\left(M-\left((\psi(M)-E)\cup\psi(TH)\right)\right) \end{cases}$$

2. $\dfrac{M,T_i\in enabled(M)\wedge T_i\in T_{ab}}{(TH,(M)^{e}_{Ta})\xrightarrow{E^{admitted(\lambda(T_i))}x}(TH',(M')^{e}_{Ta})}$ such that

$$\begin{cases} * \ \forall M'' \in \min(M,T), E=\psi(M''), \\ \qquad M''' = makefree(E, M-M'') \\ * \ \forall p\in P, M'(p)=M'''(p) \\ * \ TH' = TH\cup\{(\emptyset,(M_0)^{x}_{T_i})\} \\ \quad \text{such that} \\ \quad \forall p\in P, \ M_0(p)= \\ \begin{cases} (0,\{(\Omega(T_i)(p),admitted(\lambda(T_i)),x)\}) \text{ if } \Omega(T_i)(p)\neq 0 \\ \\ (0,\emptyset) \text{ otherwise} \end{cases} \\ * \ x = get\left(M-\left((\psi(M)-E)\cup\psi(TH)\right)\right) \end{cases}$$

3. $\dfrac{th_i,\frac{\exists\gamma_j\in\gamma}{cutstep(th_i,\gamma_j)}}{(TH,(M)^{e}_{Ta})\xrightarrow{\{x\}^{finished(\lambda(T_i))}x}(TH',(M')^{e}_{Ta})}$ such that

$$\begin{cases} * \ \forall th_i = \left(TH_i,(M_i)^{e_i}_{T_i}\right)\in TH, \\ \qquad x=e_i, \ TH' = TH-\{th_i\} \\ * \ M' = occur(T_i,x,M) \text{ such that:} \\ \quad \forall p\in P, \text{ if } M'(p)=(FT',BT') \text{ and} \\ \qquad M(p)=(FT,BT) \text{ then} \\ FT' = FT \\ BT' = \begin{cases} BT\cup\{(W^{+}(p,t,j),finished(\lambda(T_i)),x)\} \\ \qquad if \ (W^{+}(p,t,j)\neq 0) \\ \\ BT \text{ otherwise} \end{cases} \end{cases}$$

4. $\dfrac{M,M\in enabled(t),t\in T_{el}\wedge K(t,T_i)\in I_p}{(TH,(M)^{e}_{Ta})\xrightarrow{E^{\lambda(t)}x}(TH',(M')^{e}_{Ta})}$

CASE STUDY

In order to illustrate the interest of the proposed approach, let us consider a fault tolerant system, which consists of a "machine PKB of the society SAIDAL in Algeria". Later, we will use a logical approach of checking. A machine PKB is composed of a turn table which rotates the bottles, a dynamic arm which moves the bottles sequentially in a rectilinear way, a filler which fills the bottles by the medicine, and a stopper closing. The speed of a turn table can cause that some of the bottles can fall. When a bottle falls, a signal crosses the photo cell. This causes the task of shifting bottles to be preempted. The machine enters a state where the problem must be recovered: first raise the bottle, then charge it. The machine PKB is modeled in Figure 13.

The maximality labeled transitions system is brought out by Figure 14. The properties to be checked are expressed in CTL. In a natural way, we can directly reasons about the actions. It should also be noted that all the properties were checked by the tool FOCOVE (Formal Concurrency Verification Environment). For example, we can verify that the execution of the action "signal" directly causes that the thread "treat" responsible for recovering errors, is launched:

AG signal => EX admitted(treat)

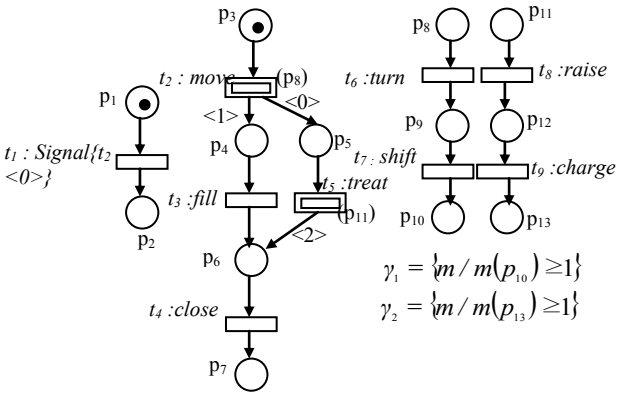


Figure 13: Modeling of a machine PKB

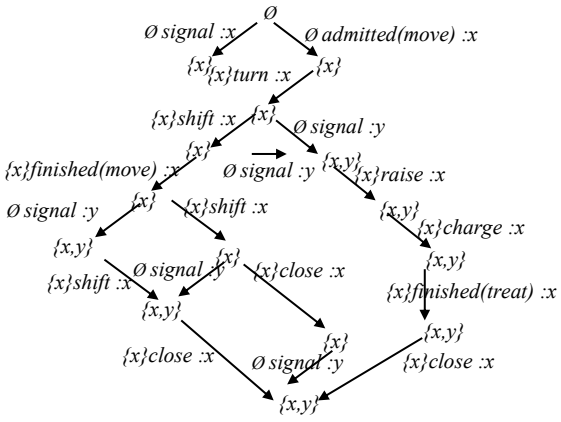


Figure 14: Maximality LTS of the PKB machine.

CONCLUSION

We proposed an operational method for building a maximality labeled transitions system from a labeled recursive Petri nets. This makes possible to take

advantadge of the verification techniques developed for the maximality labeled transitions systems. Here, the properties relating to the good performance of a system specified by a Petri net can be checked on it corresponding maximality labeled transitions system. It worth noting that the structure of the maximality labeled transitions systems represents the parallel execution of actions, as well as the parallel execution of threads.

REFERENCES

Aceto. L and Hennessy. M. 1991. "Adding action refinement to finite process algebra", in J. L. Albert, B. Monien and M. R. Artalejo, eds, ICALP'91, lncs, 510:506-519.

Andrews. D, Groote . J and Middelburg .C, eds, Workshop on Semantics of Specification Languages (SoSL'93), Workshops in Computing, Springer-Verlag, Utrecht, 289-303.

Best E, Devillers R, Kiehn A and Pomello L, 1991, "Concurrent bisimulations in Petri nets", Acta Informatica 28:231-264.

Boudol. G, and Castellani. I. 1988, Concurrency and atomicity, TCS 59:1-60.

Courtiat. J.P and Saïdouni. D.E. 1994, "Action refinement in LOTOS", in A. Danthine, G. Leduc and P. Wolpe, eds, Protocol Specification, Testing and Verification (PSTV'93), North-Holland, 341-354.

Courtiat. J.P and Saïdouni. D.E. 1995, "Relating maximality-based semantics to action refinement in process algebras".

Dahmani D., Ilié J-M., Boukala M. 2009, "Reachability analysis for Recursive Petri Nets with shared places". Abstractions for Petri Nets and Other Models of Concurrency (APNOC'09) - Petri Nets 2009, 65-79.

Dahmani. D. "Extensions of recursive Petri nets for temporal analysis of systems with dynamic structure". Doctoral thesis (2009). University of Sc. Techno. Houari Boumediene algeria.

Dahmani D., Ilié J-M., Boukala M. 2008, "Time Recursive Petri Nets". Transactions on Petri Nets and Other Models of Concurrency I (TopNoc) - Petri nets and system engineering, lncs, vol. 5100-2008, 104-118.

Darondeau. P and Degano. P. 1989, "Causal trees", in ICALP'89, LNCS, Springer-Verlag, 372:234-248.

Darondeau. P and Degano. P. 1991, "About semantic action refinement", Fundamenta Informaticae 14:221-234.

Darondeau. P and Degano. P. 1993, "Refinement of actions in event structures and causal trees", TCS 118:21-48.

Degano. P. and Gorrieri. R. 1991, "Atomic refinement in process description languages", in A. Tarlecki, ed., Mathematical Foundations of Computer Science, LNCS, Springer-Verlag, 520: 121-130.

Devillers. R. 1992a, "Maximality preservation and the ST-idea for action refinement", in G. Rozenberg, ed., Advances in Petri Nets, LNCS Springer-Verlag, 609:108-151.

Devillers. R. 1992b, "Maximality preserving bisimulation", TCS 102:165-183.

Devillers. R. 1993, "Construction of S-invariants and S-components for refined Petri boxes", in M. A. Marsan, ed., ATPN'93, LNCS, Springer-Verlag, 691:242-261.

Dijkstra. E.W 1971, "Hierarchical ordering of sequential processes", Acta Informatica 1(2):115-138.

Hogrefe. D and Leue. S, eds, IFIP TC6/WG6.1, 7th Int. Conf. on Formal Description Techniques (FORTE'94), Chapman & Hall, 293-308.

Janssen. W., Poel. M. and Zwiers. J. 1991, "Action systems and action refinement in the development of parallel systems", CONCUR'91, LNCS, 527:298-316.

Saïdouni D.E. and Courtiat. J.P. 1994, "Syntactic action refinement in presence of multiway synchronization". Semantics of Specification Languages 1993: 289-303

Saïdouni. D.E. 1996, "Maximality semantic: Application to actions refinement in LOTOS", PhD thesis., LAAS-CNRS, 7 av. du Colonel Roche, 31077 Toulouse Cedex France.

Saidouni. D.E., Belala. N and Bouneb. M. "Using maximality- based labeled transitions as model for Petri nets". The International Arab Conference on Information Technology (ACIT'2008).

Saidouni. D. E, Belala. N and Bouneb. M. "Aggregation of transitions in marking graph generation based on maximality semantics for Petri nets". Verification and Evalution of Computer and Communication systems (VECOS'2008).

Saidouni. D. E., Belala. N and Bouneb. M. "Using maximality- based labeled transitions as model for Petri nets". The International Arab Journal of Information Technology (IAJIT'2009a)

Saidouni. D. E, Belala. N and Bouneb. M. "Maximality-Based Structural Operational Semantics for Petri Nets". 2nd Mediterranean Conference on Intelligent Systems and Automation (CISA'2009b).

Van Glabbeek. R. J. 1990, "The refinement theorem for ST-bisimulation semantics", in IFIP Working Conference on Programming Concepts and Methods, North-Holland.

AUTHOR BIOGRAPHIES

Djamel E. Saïdouni was born in Algeria, he obtained his BEng degree from University of Mentouri Constantine, Algeria (1990). He obtained his PhD in theoretical computer science and concurrency from the University of Paul Sabatier, Toulouse, France (1996). Actually he is a professor at the University of Constantine 2 in Algeria and a permanent researcher of MISC laboratory where he is a head of FDCS team. His research domain research concerns formal specification and verification of complex distributed and real time systems.

.

Messaouda Bouneb was born in Algeria, she obtained her BEng degree from University of Mentouri Constantine, Algeria (2005). In February 2009, she obtained her M.Sc. degree in computer science at the University of El Arbi Ben-M'hidi Oum El-Bouaghi, Algeria. Her research domain is formal specification and verification of concurrent systems by the use of Petri nets.

Jean-Michel Ilié was born in France, he obtained several degrees in electronics and informatics among with its PhD thesis from the UPMC University of Paris (1990). Currently, member of the Paris Descartes University in its conference master higher grade (2009), he is also a permanent researcher of the LIP6 laboratory - UPMC. The fields of his research concern the formal validation of complex embedded distributed systems.