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)).

\[\text{Figure 1: Petri nets.}\]

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 \(t_2\) and \(t_3\), or the transitions \(t_4\) and \(t_5\).

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 \(t_2\) and \(t_4\) are labeled by the action \(a\) 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.

\[\text{Figure 2: interleaving Semantics}\]

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.


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

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.

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**


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 \( \mathcal{R}=(\mathcal{P},\mathcal{T},\mathcal{I},\mathcal{W},\mathcal{W}^+\mathcal{\Omega},\mathcal{\gamma},\mathcal{K}) \) such that:

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

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

**MAXIMALITY BASED TRANSITIONS SYSTEMS**

Let \( \mathcal{E} \) be a countable set of event names and \( \mathcal{A} \) an alphabet of actions.
Definition 3: A maximality-based labeled transition system defined over $\mathcal{Ev}$ is a 5-uplet $(\Omega, \lambda, \mu, \xi, \psi)$ where $\Omega = \langle S, T, \alpha, \beta, s_0 \rangle$ 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.
- $\alpha$ and $\beta$ 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 $\Omega$.
- $(\Omega, \lambda, \mu, \xi, \psi)$ is a system of transitions wherein each transition is labeled by an action of $A$, an occurrence of which must be started ($\lambda : T \rightarrow A$).
- $\psi : S \rightarrow 2^{\mathcal{Ev}}$ associates with each state, a finite set of maximal event names, related to the actions.
- $\xi : T \rightarrow \mathcal{Ev}$ associates with each transition, the event name, identifying a new occurrence of action to be being started.
- $\mu : T \rightarrow 2^\mathcal{Ev}$ 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 $t$, $\mu(t) \subseteq \psi(\alpha(t))$, $\xi(t) \subseteq \psi(\alpha(t)) \cap \psi(\beta(t)) = \psi(\alpha(t)) \cap \mu(t)$ and $\psi(\beta(t)) = \psi(\alpha(t)) \cap \mu(t) \cup \xi(t)$. For sake of concision, $\mu^x$ is also noted $E_a x$.

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 $\psi$ is used to compute this set:
  $\psi(M) = \bigcup_{p \in P} \psi(p(x_i, \ldots, x_{n-1}))$.
- Assuming that, for all $p$ in $P$, $M(p) = (FT,BT)$ with $BT = \{x_1, \ldots, x_{n-1}\}$.
- Let $E \subseteq Ev$ be a non-empty and finite set of event names, the function $\text{makefree}(E,M)$ is defined to free the bound tokens of a set $E$ from a marking $M$, as follows:
  - $\text{makefree}(\{x_1, \ldots, x_{n-1}\}, M)$ = $\text{makefree}(\{x_{n-1}\}, M)$
  - $\text{makefree}(\{x\}, M)$ = $M$ such that for all $p \in P$, considering $M(p) = (FT,BT)$, then:
    - If there is a bound token $(n,a,x) \in BT$ in $p$ then $M'((p)) = (FT+n, BT-\{(n,a,x)\})$
    - Otherwise, $M'((p)) = M(p)$.
- A transition $t$ of $T$ is enabled in a marking $M$ iff $|M(p)| \geq W(p,t)$ for all $p \in P$. The set of all the transitions enabled in $M$ is denoted $\text{enabled}(M)$.
- The marking $M$ is said to be minimal for the firing of the transition $t$ iff $|M(p)| = W(p,t)$ for all $p \in P$.
- 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 $M_1, M_2 \subseteq M$. If $FT_1 \subseteq FT_2$ and $BT_1 \subseteq BT_2$, then the relation $\leq$ is extended to sets of bound tokens as follows: $\text{BT}_1 \leq \text{BT}_2$ iff $\forall (n,a,x) \in \text{BT}_1$, $\exists (n,a,x) \in \text{BT}_2$ 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 $M_1, M_2 \subseteq M$. The difference $M_1 - M_2$ is a marking $M$ such that for all $p \in P$, if $M_1((p)) = (FT_1,BT_1)$ and $M_2((p)) = (FT_2,BT_2)$ then $M((p)) = (FT_1,BT_1)$ with $FT_3 = FT_1 - FT_2$ and $BT_3 = BT_1 - \{(n,a,x) \in BT_1, (n,a,x) \in BT_2, n_1 \neq n_2\}$.
- The function $\text{get}: 2^{\mathcal{Ev}} \rightarrow Ev$ is a function which satisfies $\text{get}(E) \in E$ for any $E \in 2^{\mathcal{Ev}}$.
- Given a marking $M$, a transition $t$ and an event name $x$ s.t. $x \in \psi(M)$, the function $\text{occurrence}(t,M) = M'$, assuming $M((p)) = (FT,BT)$, and $M'(p) = (FT,BT')$ for all $p \in P$ 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 $\tau_2$ is an abstract transition, the firing of which represents the execution of the action $b$. The firing of $\tau_2$ is a consequence of the end of execution related to the action $a$, attached to the transition $\tau_1$. 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 $p_2$. 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 p5.

\[ \gamma_1 = \{M(p_8) \geq 1\} \]

Figure 5: Recursive Petri net

After the generation of a bound token in the place p5, 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 \( \gamma^t = \{M(p_d) \geq 1\} \). This condition holds when either one of the transition \( t_6 \) or \( t_7 \) produces a token in the right part of the place p8. 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 r of the son thread.

\[ \phi : x \]
\[ \{x\}d : x \]
\[ \{x\}e : x \]
\[ \{x\}f : x \]
\[ \{x\}g : x \]
\[ \{x\}c : x \]
\[ \{x\}f : x \]
\[ \{x\}g : x \]
\[ \{x\}c : x \]

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 \( t_2 \) 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 \( t_1 \).

\[ \gamma_2 = \{M(p_8) \geq 1 \lor M(p_9) \geq 1\} \]

Figure 7: Sequence of activities.

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

Semantic of parallelism

Consider now the recursive Petri net of Figure 9 where \( t_2 \) is an abstract transition. The behaviors associated with the two firing occurrences of this transition can be executed concurrently in parallel.

\[ \gamma_3 = \{M(p_3) \geq 1\} \]

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 \( t_2 \), 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.

**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 \( M, M_0, \ldots \) of \( R \).

**Preliminary definitions.**

1. We call thread configuration any pair of the form \((TH, (M)^{ST}_T)\) 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 \((TH, (M)^{ST}_T)\) 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 \((TH, (M_0)^{ST}_T)\). For sake of homogeneity, an extra event, namely \( e_0 \not \in \text{Ev} \), is introduced to virtually launch the initial thread.

Moreover, according to any thread configuration \((TH, (M)^{ST}_T)\), let us introduce the following mappings:

- \( \psi: THs \to \text{Ev} \) is the mapping which yields all the event names referred in a thread configuration and its descents. It is recursively defined by:
  - \( \psi(\emptyset, (M)^{ST}_T) = \psi(M) \)
  - \( \psi(TH, (M)^{ST}_T) = (\bigcup_{t \in TH} \psi(th)) \cup \psi(M) \)

- \( \text{makefree}: 2^{\text{Ev}} \times \text{THs} \to \text{THs} \) is used to free bound token within a thread configuration. It is recursively defined by:
  - \( \text{makefree}((E, (\emptyset, (M)^{ST}_T))) = (\emptyset, \text{makefree}(E, (M)^{ST}_T))^e \)
  - \( \text{makefree}((E, (TH, (M)^{ST}_T))) = \left( \bigcup_{t \in TH} \text{makefree}(E, th), \text{makefree}(E, (M)^{ST}_T) \right)^e \)

- The enabling test of transitions is standard. The set \( \text{min}(M, t) \) denotes the set of all possible minimal markings built from \( M \) that enable the transition \( t \). To deal with cut step, let the mapping \( \text{cutstep} \) be s.t.
  - \( \text{cutstep}((TH, (M)^{ST}_T), \gamma) \) is true iff \( \forall p \in P \; [M(p) \geq \gamma(p)] \)

- The production of tokens specified by the mapping \( \text{occur} \) 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. \[ \frac{M \in \text{enabled}(M) \forall T \in T_d}{\text{TH}(M)^e_{T_d}} \] such that:
   - \( \forall M' \in \min(M, t), E = \psi(M') \)
   - \( M' = \text{makefree}(E, M - M') \)

2. \[ \frac{M_T \in \text{enabled}(M) \forall T \in T_{ab}}{\text{TH}(M)^e_{T_{ab}}} \] such that:
   - \( \forall M' \in \min(M, T), E = \psi(M') \)
   - \( M' = \text{makefree}(E, M - M') \)
   - \( \forall p \in P, M'(p) = M^*(p) \)
   - \( \forall T' \in \text{TH} \bigcup \{ \{0, (M_0)^T_{i} \} \} \) such that:
     - \( \forall p \in P, M_0(p) = \left\{ \begin{array}{l}
     \{0, (\Omega(T_i)(p), \text{admitted}(\lambda(T_i)), x) \} \text{ if } \Omega(T_i)(p) \neq 0 \\
     \{0, \emptyset \} \text{ otherwise }
     \end{array} \right. \)
     - \( x = \text{get} \left( M - (\psi(M) - E) \cup \psi(\text{TH}) \right) \)

3. \[ \frac{\text{th}_i \in \text{cutstep}(\text{th}_i)}{\text{TH}(M)^e_{T_d}} \] such that:
   - \( \forall \text{th}_i \in \{ \text{TH}_{i}(M)^e_{T_d} \} \)
   - \( x = e_{th_i}, \text{TH} = \text{TH} - \{ \text{th}_i \} \)
   - \( M' = \text{occur}(T_i, x, M) \) such that:
     - \( \forall p \in P, M'(p) = (\text{FT}', \text{BT}') \)
     - \( M(p) = (\text{FT}, \text{BT}) \) then
     - \( \text{FT}' = \text{FT} \)
     - \( \text{BT}' = \left\{ \begin{array}{l}
     \{ \{W^+(p, t, j), \text{finished}(\lambda(T_i)), x) \} \text{ if } (W^+(p, t, j) \neq 0) \\
     \text{BT otherwise}
     \end{array} \right. \)

4. \[ \frac{M \in \text{enabled}(T) \forall T \in T_{rp}}{\text{TH}(M)^e_{T_{rp}}} \] such that:
   - \( M, M' \in \text{enabled}(T) \forall T \in T_{rp} \)

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 formal way, we can directly reason 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:

\[ \text{AG} \text{ signal} \Rightarrow \text{EX} \text{ admitted(treat)} \]

CONCLUSION

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


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.

Messaaouda 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 Ilji 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.