Computing and Informatics, Vol. 37, 2018, 635–655, doi: 10.4149/cai_2018_3_635

HIERARCHICAL SYSTEM DESIGN USING REFINABLE RECURSIVE PETRI NET

Messaouda Bouneb

Department of Mathematic and Computer Science
El Arbi ben M'hidi University
Oum el boighi, Algeria
e-mail: bounebm.univ@gmail.com

Djamel Eddine Saidouni

Department of Computer Science
Abed Elhamid Mehri Constantine 2 University
Constantine, Algeria
e-mail: saidounid@hotmail.com

Jean Michel Ilie

Department of Computer Science
Pierre and Marie Curie University
Paris, France
e-mail: jean-michel.ilie@lip6.fr

Abstract. This paper is in the framework of the specification and verification of concurrent dynamic systems. For this purpose we propose the model of Refinable Recursive Petri Nets (RRPN) under a maximality semantics. In this model a notion of undefined transitions is considered. The underlying semantics model is the Maximality Abstract Labeled Transition System (AMLTS). Then, the model supports a definition of a hierarchical design methodology. The example of a cutting flame machine is used for illustrating the approach.

Keywords: Recursive Petri nets, hierarchical design, action refinement, maximality labeled transition system

1 INTRODUCTION

Petri nets model is a graphical and mathematical modelling tool which is used to specify, in clear manner, the behaviors of concurrent systems. The marking graph associated with a given Petri net is used for checking the specified 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 by their interleaved sequential executions. To clarify the ideas, we consider the example of two Petri nets (Figures 1 a) and 1 b)). 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 executes sequentially transitions t_1 and t_3 or transitions t_2 and t_4.

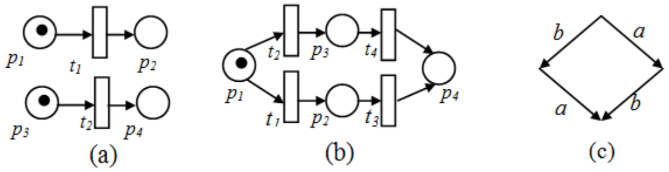

Figure 1. Petri nets

After generating the marking graphs of the two Petri nets, where transitions t_1 and t_4 are labeled by action a and transitions t_2 and t_3 are labeled by action b, the two marking graphs become isomorphic. Therefore the parallel execution of action a and action b is interpreted as their interleaved executions in time. 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, in reality this assumption is not accepted. In order to accept the verification results, the realization constraints should be taken into account at both specification and semantic level. To clarify the idea, let us consider that the transition t_1 (resp. t_4) consists of two sequential transitions t_{1-1} and t_{1-2} (resp. t_{4-1} and t_{4-2}). Transitions t_{1-1} and t_{4-1} are labeled by action a_1 whereas transitions t_{1-2} and t_{4-2} are labeled by action a_2. The refined Petri nets and their labeled transition systems are represented by Figure 2. It is clear that the behaviors of both Petri nets are different.

Indeed, in the first system, the execution of action b can occur between the execution of actions a_1 and a_2; which is not the case in the second system. Taking into account the 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 [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]. Considering such semantics allows a hierarchical design of the systems by refining actions (actions are seen as abstract processes). An other interest of these semantics is the characterization of parallel executions of non instantaneous actions.

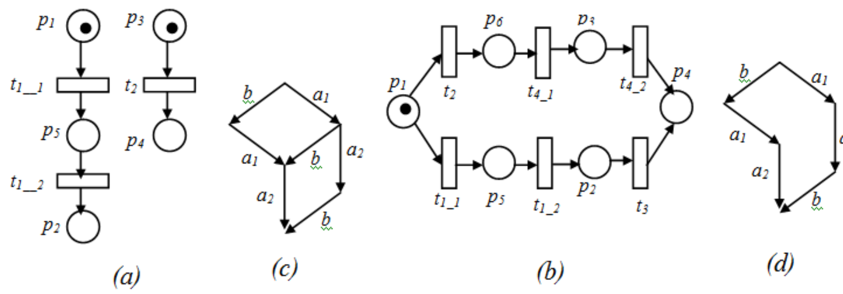

Figure 2. Non structural atomicity of actions

In this context the maximality semantics, through the model of the maximality labeled transition systems, was used for the characterization of concurrent systems. This semantics was defined for process algebras and place transition Petri nets [13, 14, 15, 16, 17]. However, the limits of Petri net model have been highlighted for the specification of systems with dynamic structures such as multi agent systems. For this reason, recursive Petri nets have been defined. Dynamic behaviors are considered through abstract transitions. Since abstract transitions represent activities, the association of true concurrency semantics to the model becomes more appropriate than the use of interleaving semantics. For this purpose, in [18] a maximality operational semantics has been proposed for recursive Petri net model. In order to design concurrent systems, several methodologies have been proposed in the literature, around process algebra and Petri net specification models. As an example we can cite the lotosphere design methodology which is based on the formal description technique Lotos. This design methodology consists in specifying firstly the architecture of the system in terms of its observable behavior, then the specification is refined along the design trajectory until obtaining the more detailed specification. This later takes into account all system functionalities and the environment constraints. As the model is based on the interleaving semantics, the design methodology is not based on action refinement but on transforming a specification, subject of refinement, to an other following some directives. An other design methodology has been defined for hierarchical Petri nets [19]. In [20] a Petri net approach to refining object behavioral specifications has been proposed. As for Lotos, these models are based on the interleaving semantics, too. Consequently design methodologies are not formal.

Since in [18] a true concurrency semantics has been defined for recursive Petri net, it seems interesting to define a design methodology based on refining abstract transitions. Thus, in this paper we extend the model of recursive Petri net by considering refinable transitions. The model will be named Refinable Recursive Petri net. The proposed model is based on the maximality semantics. As for recursive Petri net, dynamic behaviors are considered through abstract transitions. These abstract transitions can be used for a hierarchical system design. Indeed, at a level of abstraction, the details of abstract behavior of a transition may be hidden. They will be exhibited in a further level of abstraction. In the first step, details of abstract transitions behaviors are undefined. These behaviors are gradually introduced along the design trajectory. In this manner, system components are integrated gradually; the initial specification is then the most abstract. This remark leads us to label abstract transitions, the behaviors of which are undefined, by the symbol $\perp$. As an example, let us consider the systems of Figure 1 where transitions t_1 and t_4 are abstract transitions, at this level the difference between the systems may not be seen. The labeled transition systems associated with the two Petri nets are given by Figures 3 a) and 3 b), respectively. To consider the details of refinement of abstract transitions, it is necessary to define relations on behaviors that consider the indefinite character of abstract transitions interstates. It is clear that two undefined transitions may become different after their refinement.

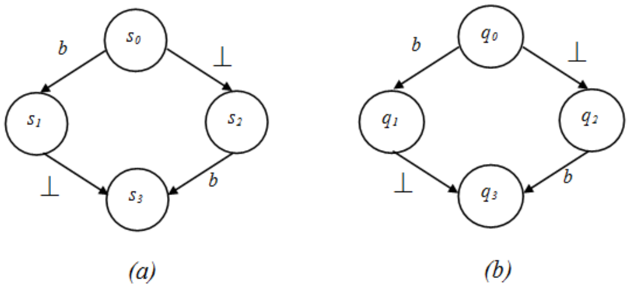

Figure 3. Interleaving approach for recursive Petri nets

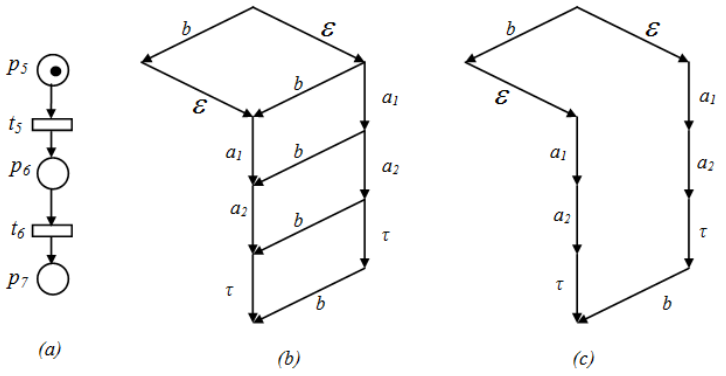

Figure 4. Refinement of abstract transitions

2 REFINABLE RECURSIVE PETRI NETS

A refinable recursive Petri net is a recursive Petri net on which all transitions are labeled by actions and abstract transitions may be labeled by $\perp$ (the abstract transitions the behaviors of which are undefined).

Formal Definitions

Definition 1. A refinable recursive Petri net is 9-uple $R = (P, T, I, W^-, W^+, \Omega, \gamma, K, \lambda)$ such that:

- P is a finite set of places.
- T is a finite set of transitions such that: $T = T_{el} \cup T_{ab}$ and $T_{el} \cap T_{ab} = \emptyset$. T_{el} denotes elementary transitions and T_{ab} denotes the abstract transitions, knowing that: $T_{ab} = T_{abd} \cup T_{abi}$ and $T_{abd} \cap T_{abi} = \emptyset$. The set of abstract transitions where behavior is defined are noted T_{abd} and the set of abstract transitions where behavior is indefinite are noted T_{abi}.
- $I = I_c \cup I_p$ is a finite set of indexes, indicates the cut steps and preemptions. $I \subset \mathbb{N}$.
- $W^- : P \times T \longrightarrow N$ is the matrix of precondition.
- $W^+ : P \times [T_{el} \cup (T_{ab} \times I) \longrightarrow N]$ is the matrix of post-condition.
- $\Omega : T_{ab} \longrightarrow \mathbb{N}^P$ is 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 (markings of termination on which the standard operations like union, intersection, projection and complementation, member test are applicable).
- $K : T_{el} \times T_{ab} \longrightarrow I_p$ is a partial function of control preemption.
- $\lambda : T \longrightarrow \mathbb{L} \cup \{\perp\}$ such that $\perp$ is the undefined label. λ is the labeling function which associates to each transition an action name. $\mathbb{L}$ ranged over by a, b, ... In practice the transition label is the name of an action.

  – $\forall t \in T_{abd}, \lambda(t) \in \mathbb{L}$.
  – $\forall t \in T_{abi}, \lambda(t) = \perp$

Definition 2. Let $R_1 = \left(P_1, T_1, I_1, W_1^-, W_1^+, \Omega_1, \gamma_1, K_1, \lambda_1\right)$ and $R_2 = (P_2, T_2, I_2, W_2^-, W_2^+, \Omega_2, \gamma_2, K_2, \lambda_2)$ be two refinable recursive Petri nets such that:

- $t \in T_{abd1}$ is an abstract transition, knowing that $T_{abd1} \subset T_{ab1} \subset T_1$.
- I_{ct} is the indexes set of the cuts of transition t.
- I_{pt} is the indexes set of preemption of transition t.
- $v \in \mathbb{N}^P$ is the start marking of transition t.
- $\gamma_t = \{\gamma_i / i \in I_{ct}\}$ is the termination set of transition t.

Then $\rho(t, R_1, R_2, I_{ct}, I_{pt}, v, \gamma_t) = \left(P_3, T_3, I_3, W_3^-, W_3^+, \Omega_3, \gamma_3, K_3, \lambda_3\right)$ is the refinable recursive Petri net obtained after the refinement of t in R_1 by R_2, such that:

- $P_3 = P_1 \cup P_2$.

- $T_3 = T_{el3} \cup T_{ab3}$ such that:
  - $T_{el3} = T_{el1} \cup T_{el2}$.
  - $T_{ab3} = T_{abd3} \cup T_{abi3}$ with:
    - $T_{abd3} = T'_{abd1} \cup T_{abd2}$ with $T'_{abd1} = T_{abd1} \cup \{t\}$.
    - $T_{abi3} = T'_{abi1} \cup T_{abi2}$ with $T'_{abi1} = T_{abi1} - \{t\}$.
- $I_3 = I_{c3} \cup I_{p3}$ such that:
  - $I_{c3} = I'_{c1} \cup I_{c2}$ with $I'_{c1} = I_{c1} \cup I_{ct}$.
  - $I_{p3} = I'_{p1} \cup I_{pt}$.
- $W_3^- : P_3 \times T_3 \longrightarrow \mathbb{N}$ such that $t' \in T_3$:

$$W_3^-(p', t') = \begin{cases} W_1^-(p', t'), & \text{if } p' \in P_1, t' \in T_1, \\ W_2^-(p', t'), & \text{if } p' \in P_2, t' \in T_2. \end{cases}$$

- $W_3^+ : P_3 \times [T_{el3} \cup (T_{abd3} \times I_3) \cup T_{abi3}] \longrightarrow \mathbb{N}$ such that: $\forall t' \in T_3$:

$$W_3^+(p', t', i') = \begin{cases} W_1^+(p', t', i'), & \text{if } p' \in P_1, t' \in T_1, \\ W_2^+(p', t', i'), & \text{if } p' \in P_2, t' \in T_2. \end{cases}$$

- $\Omega_3 : T_{abd3} \longrightarrow \mathbb{N}^P$ such that $\forall t' \in T_{abd3}$:

$$\Omega_3(t') = \begin{cases} \Omega_1(t'), & \text{if } t' \in T_{abd1}, \\ \Omega_2(t'), & \text{if } t' \in T_{abd2}, \\ v, & \text{if } t' = t. \end{cases}$$

- $\gamma_3 = \gamma'_1 \cup \gamma_2$ such that: $\gamma'_1 = \gamma_1 \cup \gamma_t$.
- $K_3 : T_{el3} \times T_{abd3} \longrightarrow I_{p3}$ such that: $\forall t_1 \in T_{el3}, t_2 \in T_{abd3}$:

$$K_3(t_1, t_2) = \begin{cases} K_1(t_1, t_2), & \text{if } t_1 \in T_{el1} \text{ and } t_2 \in T_{abd1}, \\ K_2(t_1, t_2) & \text{if } t_1 \in T_{el2} \text{ and } t_2 \in T_{abd2}, \\ i, & \text{such that } i \in (\mathbb{N} - (I_1 \cup I_2)). \end{cases}$$

- $\lambda_3 : T_3 \longrightarrow \mathbb{L} \cup \{\perp\}$: such that $\forall t' \in T_3$:

$$\lambda_3(t') = \begin{cases} \lambda_1(t'), & \text{if } t' \in T'_{abi1} \cup T'_{abd1}, \\ \lambda_2(t'), & \text{if } t' \in T_2, \\ a, & \text{otherwise with } a \in \mathbb{L}. \end{cases}$$

3 MAXIMALITY-BASED LABELED TRANSITION SYSTEMS

A maximality-based labeled transitions system is a graph labeled on both states and transitions. Each state is labeled by a set of event names. Each event name identifies the start of execution of an action which occured before this state. This action is said to be potentially under execution in this state. A transition between two states s_i and s_j is labeled by a 3-uple (G, a, x) (noted ${}_G a_x$) where x is the event name identifying the start of execution of the action a and G identifies the set of event names representing the causes of the action a. Elements of G belong to state s_i. Occurence of this transition terminates actions identified by G, thus, the set of event names corresponding to state s_j is that of s_i from which the set G is substructed and the event name x is added. The formal definition of a maximality-based labeled transition system is given in Definition 3.

Formal Definitions

Definition 3. Let $\mathcal{H}$ be a countable set of event names. $2^{\mathcal{H}}$ denotes the set of part-set of $\mathcal{H}$.

A maximality-based labeled transitions system of support $\mathcal{H}$ is a fivefold $(\eta, \varphi, \mu, \xi, \theta)$ with: $\eta = \langle S, TR, \alpha, \beta, S_0 \rangle$ is a system of transitions such that:

- S is the set of states in which the system may be found, this set can be finite or infinite.
- TR is the set of transitions indicating the change of states which the system can do; this set can be finite or infinite.
- α and β are two applications of TR in S such that for any transition $tr \in TR$ we have: $\alpha(tr)$ is the origin of the transition tr and $\beta(tr)$ its goal.
- S_0 is the initial state of the transition system η.
- (η, φ) is a system of transitions labeled by the function φ on an alphabet $\mathbb{L}$, called support of (η, φ). $(\varphi : TR \longrightarrow \mathbb{L})$ such that $\mathbb{L}$ ranged over by a, b, ... In practice a transition label is a name of an action.
- $\theta : S \longrightarrow 2^{\mathcal{H}}$ is a function which associates to each state a finite set of maximal event names. With the assumption that $\theta(S_0) = \emptyset$.
- $\mu : TR \longrightarrow 2^{\mathcal{H}}$ is a function which associates to each transition a finite set of event names corresponding to the actions which began their execution and their terminations cause the execution of this transition.
- $\xi : TR \longrightarrow \mathcal{H}$ is a function which associates to each transition the event name identifying its occurrence.

With the condition that for each transition $tr \in TR$ $\mu(tr) \subseteq \theta(\alpha(tr))$, $\xi(tr) \notin \theta(\alpha(tr)) - \mu(tr)$ and $\theta(\beta(tr)) = (\theta(\alpha(tr)) - \mu(tr)) \cup \{\xi(tr)\}$.

Notation 1. Let $mlts = (\eta, \varphi, \mu, \xi, \theta)$ be a maximality-based labeled transitions system such that $\eta = \langle S, TR, \alpha, \beta, S_0 \rangle$. $tr \in TR$ is a transition such that $\alpha(tr) = s$, $\beta(tr) = s'$, $\varphi(tr) = a$, $\mu(tr) = E$ and $\xi(tr) = x$. The transition tr will be noted $s \xrightarrow{E a_x} s'$.

Definition 4. Let $mlts_1 = (\eta_1, \varphi_1, \mu_1, \xi_1, \theta_1)$ and $mlts_2 = (\eta_2, \varphi_2, \mu_2, \xi_2, \theta_2)$ be two maximality labeled transition systems with: $\eta_1 = (S_1, TR_1, \alpha_1, \beta_1, s_{01})$ and $\eta_2 = (S_2, TR_2, \alpha_2, \beta_2, s_{02})$. $mlts_1$ and $mlts_2$ are isomorphic if there exists a bijection $h : S_1 \longrightarrow S_2$ such that: $\forall s, s' \in S_1 / s = \alpha_1(tr)$ and $s' = \beta_1(tr)$ then: $tr \in TR_1 \Longleftrightarrow tr' \in TR_2$ with:

- $\alpha_2(tr') = h(s)$ and $\beta_2(t) = h(s')$.
- $\theta_1(s) = \theta_2(h(s))$ and $\theta_1(s') = \theta_2(h(s'))$.
- $\varphi_1(tr) = \varphi_2(tr')$.
- $\mu_1(tr) = \mu_2(tr')$.
- $\xi_1(tr) = \xi_2(tr')$.

Definition 5. An abstract maximality labeled transition system $amlts = (\eta, \varphi, \mu, \xi, \theta)$ is a MLTS labeled by function φ on the alphabet $\mathbb{L} \cup \{\perp\}$. $\varphi : TR \longrightarrow \mathbb{L} \cup \{\perp\} / \perp$ denotes an undefined labeling.

Definition 6. An abstract maximality labeled transition system $amlts = \eta, \varphi, \mu, \xi, \theta)$ is said $\perp$-free if and only if $\forall tr \in TR : \varphi(tr) \neq \perp$. A $\perp$-free abstract maximality labeled transition system is a maximality labeled transition system.

Definition 7. The isomorphism on maximality labeled transition systems is extended to abstract maximality labeled transition system by extending the function φ to $\mathbb{L} \cup \{\perp\}$.

4 MAXIMALITY SEMANTICS FOR PETRI NET

In this section we recall the maximality approach of place transition Petri nets, proposed in [14, 16]. We introduce through simple example useful notations and functions for the definition of marking graph associated to a Petri net in a maximality-based approach.

Consider the example of the marked Petri net of Figure 5. With the launch of the transition t_1, it is clear that the firings of transitions t_2 and t_3 are conditioned by the end of the action related to t_1. To capture this causal dependence between firings of transitions, we consider that tokens produced by the firing of the transition t_1 are bound to this transition, namely the token in place p_2 and the token in place p_3 (Figure 6 b)). We can see that, in the initial state, the token in p_1 is not bound to any transition; this token is called free in this state, then the marked Petri net of Figure 6 a). In the case when t_2 would be fired, it could be argued that the action associated with the firing of t_1 has finished its execution. As a result, the token in p_3

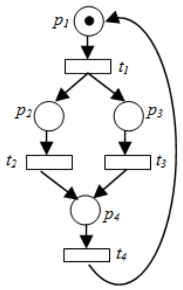

Figure 5. Marked Petri net

will become free. Resulting marking after the firing of the transition t_2 is given in Figure 6 c).

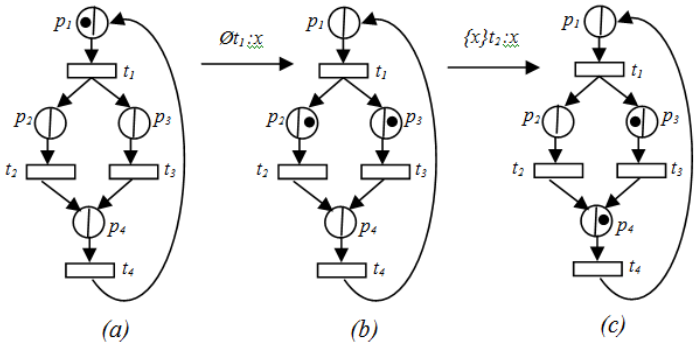

Figure 6. Free and bound tokens in a marking

To distinguish between free and bound tokens in a place, we can imagine that a place is composed of two separated parts. The left part contains free tokens while the right one will contain bound tokens. In a place, the number of free tokens will be denoted by $\mathcal{FT}$, while bound tokens set will be noted $\mathcal{BT}$. So each place is marked by $(\mathcal{FT}, \mathcal{BT})$. Hence, we obtain the succession of markings of Figure 6. Each bound token identifies an action that is eventually being executed (this token corresponds to a maximal event). Also each transition of marking graph corresponds to the start of execution of an action which is identified by an event name. Since a weight of an edge linking a transition to a place may be grater than one, a firing transition may produce more than one bound token, the bound token is identified by a tuple (n, t, x) where n is a number of instance of a bound token, t is a firing transition producing this bound token and x is an event name identifying the transition firing in time. Note that the firing condition of a transition is only conditioned by the number of free and bound tokens in places.

Preliminary Definitions

A Petri net is a tuple (P, T, W) where:

- P : is a finite set of places.
- T : is a finite set of transition such that $P \cap T = \emptyset$.
- $W : (P \times T) \cup (T \times P) \longrightarrow \mathbb{N}$ is the weight function.

Let (P, T, W) be a Petri net with a marking M:

- The set of maximal event names in M is the set of all event names identifying bound tokens in the marking M. Formally, the function δ will be used to calculate this set, it can be defined as: $\delta : M \longrightarrow PR(\mathcal{H})$. $\delta(M) = \cup_{p \in P} \{x_1, x_2, .., x_m\}$ such that $M(p) = (\mathcal{FT}, \mathcal{BT})$ with: $\mathcal{BT} = \{(n_1, t_1, x_1), \ldots, (n_m, t_m, x_m)\}$.
- Let $X \subset \mathcal{H}$ be a finite set of event names. The operation of transforming bound tokens defined by X to free tokens in the marking M is defined by the inductive function makefree as follows:
  - $\mathit{makefree}(\{x_1, x_2, \ldots, x_n\}, M) = \mathit{makefree}(\{x_2, \ldots, x_n\}, \mathit{makefree}(\{x_1\}, M))$
  - $\mathit{makefree}(\{x\}, M) = M'$ such that for all $p \in P$, if $M(p) = (\mathcal{FT}, \mathcal{BT})$ then:
    - ∗ If there is $(n, t, x) \in \mathcal{BT}$ then $M'(p) = (\mathcal{FT} + n, \mathcal{BT} - \{(n, t, x)\})$ (conversion of n bound tokens identified by the event name x to free tokens).
    - ∗ Otherwise, $M'(p) = M(p)$.
- $\mid M(p) \mid = \mathcal{FT} + \sum_{i=1}^{m} n_i$ such that $M(p) = (\mathcal{FT}, \mathcal{BT})$ with $\mathcal{BT} = \{(n_1, t_1, x_1), \ldots, (n_m, t_m, x_m)\}$.
- Let t be a transition of T; t is said to be enabled by the marking M iff $\mid M(p) \mid \geq W(p, t)$ for all $p \in P$. The set of all transitions enabled by the marking M will be noted *enabled*(M).
- The marking M is said minimal for the firing of the transition t iff $\mid M(p) \mid = W(p, t)$ for all $p \in P$.
- Let M_1 and M_2 be two markings of the Petri net (P, T, W). $M_1 \subseteq M_2$ iff $\forall p \in P$, if $M_1(p) = (\mathcal{FT}_1, \mathcal{BT}_1)$ and $M_2(p) = (\mathcal{FT}_2, \mathcal{BT}_2)$ then $\mathcal{FT}_1 \geq \mathcal{FT}_2$ and $\mathcal{BT}_1 \subseteq \mathcal{BT}_2$ such that the relation $\subseteq$ is extended to bound tokens sets as follows: $\mathcal{BT}_1 \subseteq \mathcal{BT}_2$ iff $\forall (n_1, t, x) \in \mathcal{BT}_1, \exists (n_2, t, x) \in \mathcal{BT}_2$ such that $n_1 \leq n_2$.
- Let M_1 and M_2 be two markings of the Petri net (P, T, W) such that $M_1 \subseteq M_2$. The difference $M_2 - M_1$ is a marking M_3 $(M_2 - M_1 = M_3)$ such that for all $p \in P$, if $M_1(p) = (\mathcal{FT}_1, \mathcal{BT}_1)$ and $M_2(p) = (\mathcal{FT}_2, \mathcal{BT}_2)$ then $M_3(p) = (\mathcal{FT}_3, \mathcal{BT}_3)$ with $\mathcal{FT}_3 = \mathcal{FT}_2 - \mathcal{FT}_1$ and $\forall (n_1, t, x) \in \mathcal{BT}_1$, $(n_2, t, x) \in \mathcal{BT}_2$, if $n_1 \neq n_2$ then $(n_2 - n_1, t, x) \in \mathcal{BT}_3$.
- $\mathit{get} : 2^{\mathcal{H}} \longrightarrow \mathcal{H}/$ for any $\mathit{elt} \in 2^{\mathcal{H}}$, $\mathit{get}(\mathit{elt}) \in \mathit{elt}$ is the function which associates to any transition an events name.

- Given a marking M, a transition t and an event name $x \notin \delta(M)$, *occur* $(t, x, M) = M'$ such that for all $p \in P$, if $M(p) = (\mathcal{FT}, \mathcal{BT})$ then $M'(p) = (\mathcal{FT}, \mathcal{BT}')$ with $\mathcal{BT}' = \mathcal{BT} \cup \{W(t,p), t, x\}$ if $W(t,p) \neq 0$ and $\mathcal{BT}' = \mathcal{BT}$, otherwise. Hence, M' is the resulting marking obtained by the addition of bound tokens related to the firing of transition t to the marking M.
- $\lambda : T \longrightarrow \mathbb{L}$ is a function which associates to any transition an action name, such that $\mathbb{L}$ ranged over by a, b, $\ldots$. In practice a transition label is a name of an action.

5 MAXIMALITY SEMANTICS FOR REFINABLE RECURSIVE PETRI NETS

All definitions remain valid for refinable recursive Petri net. Other notations and functions will be given for the definition of the operational maximality semantics. The proposed approach and the interest of hierarchical design are illustrated through simple examples. Since abstract transitions behaviors are introduced gradually, conflict, sequencing and parallelism relations linking an abstract transition to other transitions are extended to transitions of refinement Petri nets.

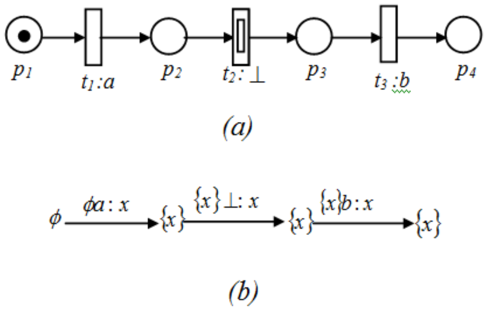

Figure 7. Start and end of undefined abstract transition

Consider the refinable recursive Petri net of Figure 7 a) in which t_2 is an abstract transition with an undefined behavior. The firing of this transition is caused by the end of the execution of the action associated to the transition t_1. Since the behavior of this transition is undefined, it will be fired as an elementary transition labeled by $\perp$. For this fact the generation of the marking graph for this net consists in the generation of the marking graph for classical Petri nets. Note that the event x identifies the beginning of the execution of an undefined process $\perp$.

Consider now the behavior of the process associated to the transition t_2 is modeled by the Petri net of Figure 8 a). The firing of this abstract transition starts the execution of its associated thread. The ordinary marking defined by the semilinear set will be prolonged to the marking of the instance of the Petri net son, this is interpreted by the addition of a token in the place p_5. The passing from the Petri

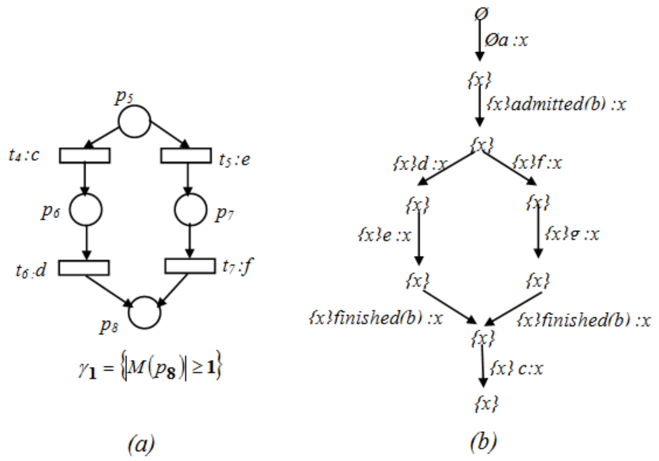

Figure 8. Refinement of the abstract transition t_2

net father to the Petri net son is made through the firing of a virtual transition called *admitted*. The firing of the transition *admitted* is causally dependent on the action a, 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 deposition of a token related to this action in the right part of the place p_5.

After the generation of a linked 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 predicate of termination$\gamma_1 = \{\mid M\left(p_8\right) \mid \geq 1\}$. This condition will be satisfied, when the transition t_6 or exclusively the transition t_7 deposits at least a token in the right part of the place p_8. When this predicate becomes true, a transition called *finished* will be fired, it makes the return to the father thread, indeed this transition represents the cut step of the son thread τ. Generally, a transition *finished* is regarded as an elementary transition. Its firing causes the emersion of the tokens defined by the post condition of the abstract transition in the right part of all places which belong to the post set of this one. Just after the end of the execution of abstract transition, the firing of the transition t_3 can happen. Figure 8 b) 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 the execution of the thread itself, thus it can be re-used within this thread. Once the thread is finished, this event name can be re-used in the father thread.

5.1 Comparison of Abstract MLTS

Definition 8. Let sys_1 and sys_2 be two systems such that: $[\mid sys_1 \mid]_{\{mlts\}}$ is not $\perp$-free and $[\mid sys_2 \mid]_{\{mlts\}}$ is not $\perp$-free. If $[\mid sys_1 \mid]_{\{mlts\}}$ and $[\mid sys_2 \mid]_{\{mlts\}}$ are abstractly isomorphs $\nRightarrow$ $[\mid \rho\left(sys_1\right) \mid]_{\{mlts\}}$ and $[\mid \rho\left(sys_2\right) \mid]_{\{mlts\}}$ are abstractly isomorphs such as: ρ is the process of refinement of an abstract transition $[||]_{\{mlts\}}$ is the process which interprets a Petri net to an abstract maximality labeled transition system.

A proposition of equality is said decidable if we can demonstrate this proposition or prove its negation. By definition two systems are equal if they have the same semantics representation. In the example of Figure 9, the two Petri nets seem that describe the same system, their amlts are abstractly isomorph (Figure 9 c)). But this does not mean that their refined systems stay equal.

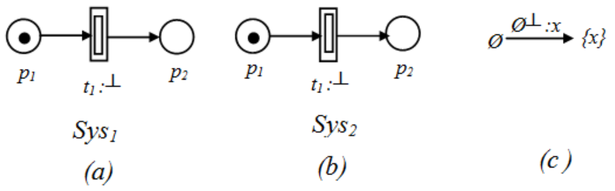

Figure 9. Comparison of two recursive Petri nets

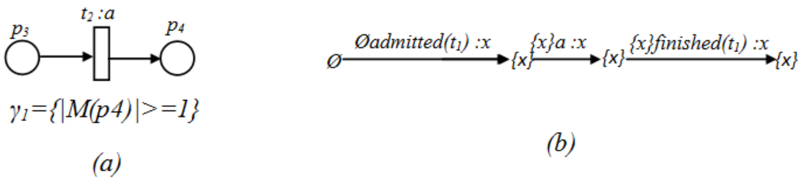

Figure 10. Case of equality after refinement

When we refine the two occurrences of the abstract transitions t_1 in sys_1, t_1 in sys_2 by the Petri net of Figure 10 a), we get two abstract isomorph maximality labeled transition systems given by the Figure 10 b). In this case, both systems are equal. However, if we refine the transition t_1 in sys_1 by the Petri net of Figure 11 a) and we refine the transition t_1 in sys_2 by the Petri net of Figure 11 b), we get two maximality labeled transition systems, which are not isomorphs at this level.

5.2 Maximality Operational Semantics for Refinable Recursive Petri Nets

Preliminary Definitions

- $\mathcal{THR}$: is the set of all threads.
- We call thread any configuration of the form: $\left(TH_i, (M_i)^{N_i}_{ref(T_i)}\right)$ such that:
  - M_i is the father marking.
  - TH_i is the set of the son threads where each thread is identified by a event name.
  - *ref* (T_i) is the Petri net corresponding to this thread, it describes the behavior of abstract transition T_i.

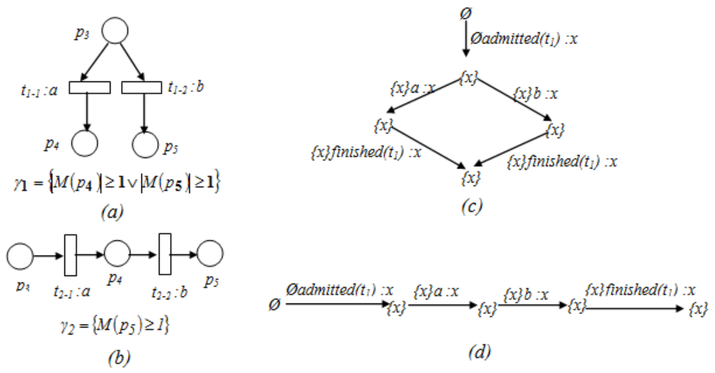

Figure 11. Case of inequality after refinement

- N_i is the event's name which identifies this thread.
- The initial configuration noted $(\emptyset,(M_0)_R^{x_0})$ is built from an initial marking M_0 of the principal Petri net R.

• Let the labeled refinable recursive Petri net $R=(P,T,I,W^-,W^+,\Omega,\gamma,K,\lambda)$ provided with a marking M:

- $\psi:\mathcal{THR}\to 2^{\mathcal{H}}$. The function which determines the events names in a thread is recursively defined by:
  * $\psi\left(\emptyset,(M)_R^N\right)=\delta(M)$.
  * $\psi\left(TH,(M)_R^N\right)=\left(\cup_{i=1}^n\psi(th_i)\right)\cup\delta(M)$ with $TH=\{th_1,th_2,..,th_n\}$.
- $\forall\left(TH,(M)_R^N\right)\in\mathcal{THR}$. $X\subset\mathcal{H}$ is a finished set of events names. $clean\left(X,\left(TH,(M)_R^N\right)\right)$ is recursively defined by:
  * $clean\left(X,\left(\emptyset,(M)_R^N\right)\right)=\left(\emptyset,(makefree(X,M))_R^N\right)$.
  * $clean\left(X,\left(TH,(M)_R^N\right)\right)=\left(\cup_{i=1}^n clean(X,th_i),(makefree(X,M))_R^N\right)$ with $TH=\{th_1,th_2,\dots,th_n\}$.
- $\forall\left(TH,(M)_R^N\right)\in\mathcal{THR}$, $t\in T$ is sensitized by this thread if and only if:
  * $\forall p\in P:|M(p)|\geq w(p,t)$ or
  * $\exists th_i=\left(TH_i,(M_i)_{ref(T_i)}^{N_i}\right)\in TH$ such that: $|M_i(p)|\geq w(p,t)$ for all $p\in ref(T_i)(p)$.
- The function *cutstep* : $\mathcal{THR}\times\gamma\to$ *boolean* is defined as follows:

$$\forall th_i=\left(TH_i,(M_i)_{ref(T_i)}^{N_i}\right)\in\mathcal{THR},\forall\gamma_i\in\gamma$$

then

$$\begin{cases} \mathit{true}, & \text{if } \forall p \in \mathit{ref}\,(T_i)\,(p)\,, M_i\,(p) \geq n \text{ with } \gamma_i = \{\mid M\,(p) \mid \geq n/n \in \mathbb{N}\}\,, \\ \mathit{false}, & \text{otherwise.} \end{cases}$$

- $TH \mid t > TH'$ means that the execution of the transition t from TH leads to TH'.
- A sequence $TH_0 t_1 TH_1 t_2 \ldots$ is an occurrence sequence iff $TH_{i-1} \mid t_i > TH_i$ for $i \geq 1$. A sequence $\sigma = t_1 t_2 \ldots$ is a transition sequence starting with TH_0 iff there is an occurrence sequence $TH_0 t_1 TH_1 t_2 \ldots$. If a finite sequence $t_1 t_2 \ldots t_n$ leads from TH to TH', we write $TH \mid t_1 t_2 \ldots t_n > TH'$.

Semantics Rules

The operational semantics of labeled refinable recursive Petri net allowing the generation of a maximality labeled transitions system is defined by the following rules:

1. $\dfrac{\text{For } M_1 \text{ a marking}, t \in \mathit{enabled}\,(M_1) \text{ with } t \in T_{el} \cup T_{abi}}{\left(TH_1, (M_1)_R^N\right) \xrightarrow{{}_E\lambda(t)_x} \left(TH_1, (M_2)_R^N\right)}$ such that:
   $\forall M_3 \in \min\,(M_1, t)$:
   - $E = \delta\,(M_3)\,, M_4 = \mathit{makefree}\,(E, M_1 - M_3)$
   - $M_2 = \mathit{occur}\,(t, x, M_4)$ such that:
     - $\forall p \in P$ if $M_4\,(p) = (\mathcal{FT}_4, \mathcal{BT}_4)$ then: $M_2\,(p) = (\mathcal{FT}_4, \mathcal{BT}_2)$ with

$$\mathcal{BT}_2 = \begin{cases} \mathcal{BT}_4 \cup \{(w\,(t,p,i)\,, \lambda\,(t)\,, x)\}\,, & \text{if } w\,(t,p,i) \neq 0, \\ \mathcal{BT}_4, & \text{otherwise.} \end{cases}$$

   - $x = \mathit{get}\left(\mathcal{H} - \left(\psi\left(\mathit{clean}\left(E, \left(TH_1, (M_1)_R^N\right)\right)\right)\right)\right)$.

2. $\dfrac{\text{For } M_1 \text{ a marking }, T_i \in \mathit{enabled}\,(M_1) \wedge T_i \in T_{abd}}{\left(TH_1, (M_1)_R^N\right) \xrightarrow{{}_E\mathit{admitted}(\lambda(T_i))_x} \left(TH_2, (M_2)_R^N\right)}$ such that:
   $\forall M_3 \in \mathit{min}\,(M_1, T_i)$:
   - $E = \delta\,(M_3)\,, M_4 = \mathit{makefree}\,(E, M_1 - M_3)$.
   - $\forall p \in P : M_2\,(p) = M_4\,(p)$.
   - $TH_2 = TH_1 \cup \left\{\left(\emptyset, (M_0)_{\mathit{ref}(T_i)}^{\{x\}}\right)\right\}$ such that:

$$\left((M_0)_{\mathit{ref}(T_i)}^{\{x\}}\right)(p) = \begin{cases} (0, \{(\Omega\,(T_i)\,(p)\,, \mathit{admitted}\,(\lambda\,(T_i))\,, x)\})\,, & \text{if } \Omega\,(T_i)\,(p) \neq 0 \\ (0, \emptyset)\,, & \text{otherwise.} \end{cases}$$

   - $x = \mathit{get}\left(\mathcal{H} - \psi\left(\mathit{clean}\left(E, \left(TH_1, (M_1)_R^N\right)\right)\right)\right)$.

3. $\dfrac{th_i, \exists \gamma_i \in \gamma / cutstep\left(th_i, \gamma_i\right)}{\left(TH_1, (M_1)_R^N\right) \overset{\{x\} finished(\lambda(T_i))_x}{\longrightarrow} \left(TH_2, (M_2)_R^N\right)}$ such that:
$\forall th_i \in TH_1 / th_i = \left(TH_i, (M_i)_{ref(T_i)}^{N_i}\right), N_i = \{x\}$:

- $M_2 = occur\left(finished\left(T_i\right), \{x\}, M_1\right)$.
- $\forall p \in P$: if $M_2(p) = \left(\mathcal{FT}_2, \mathcal{BT}_2\right)$ then

$$\mathcal{BT}_2 = \begin{cases} \mathcal{BT}_1 \cup \left\{\left(w(t,p,i), finished\left(\lambda\left(T_i\right)\right), x\right)\right\}, & \text{if } w(t,p,i) \neq 0, \\ \mathcal{BT}_1, & \text{otherwise.} \end{cases}$$

- $TH_2 = TH_1 - \{th_i\}$.

4. $\dfrac{M_1, M_1 \in enabled(t), t \in T_{el} \wedge K\left(t, T_i\right) \in I_p}{\left(TH_1, (M_1)_R^N\right) \overset{E\lambda(t)_x}{\longrightarrow} \left(TH_2, (M_2)_R^N\right)}$ such that:
$\forall th_i \in TH_1 / th_i = \left(TH_i, (M_i)_{ref(T_i)}^{N_i}\right)$, $\forall M_3 \in \min\left(M_1, t\right)$:

- $E = \delta\left(M_3\right), M_4 = makefree\left(E, M_1 - M_3\right)$.
- $M_2 = occur\left(occur\left(t, E, M_3\right), finished\left(\lambda\left(T_i\right)\right), N_i\right)$.
- $\forall p \in P$: $if M_2(p) = \left(\mathcal{FT}_2, \mathcal{BT}_1\right)$ then, $M_2(p) = \left(\mathcal{FT}_4, \mathcal{BT}_2\right)$ with:

$$\mathcal{BT}_2 = \begin{cases} \mathcal{BT}_4 \cup \left\{\left(w(t,p,i), t, x\right)\right\}, & \text{if } w(t,p,i) \neq 0, \\ \mathcal{BT}_4 \cup \left\{\left(w(T_i,p,i), finished\left(\lambda\left(T_i\right)\right), N_i\right)\right\}, & \text{if } w(t,p,i) \neq 0, \\ \mathcal{BT}_4, & \text{otherwise.} \end{cases}$$

- $TH_2 = TH_1 - \{th_i\}$.
- $x = get\left(\mathcal{H} - \left(\psi\left(clean\left(E, \left(TH_1, (M_1)_R^N\right)\right)\right) - \psi\left(th_i\right)\right)\right)$.

6 CASE STUDY

As an example, we consider a flame cutting machine used to fabricate pieces for vehicles: this machine consists of: the reading head, the blowtorches and the template which is a diagram dimensioning pieces. It has a movable table on which the template is located, and then transmits it as a dimensional information. The read information is transmitted to blowtorches controller, as a result the blowtorches cut the piece from metal sheet according to that as defined on the template. Because some external events may cause errors (modification of table position) during the cutting process, in such a case the following actions are immediately produced:

- Displaying output indicating the occurrence of a problem using a lamp.
- The blowtorches will be stopped.
- The table is reported in its initial position.

The assessment of this system is done by verifying some properties, expressed in CTL logic, using the formal verification environment FOCOVE (Formal Concurrent Verification Environment).

Step 01: In this first step we abstractly model the system tasks. Consequently tasks considering behaviors are not yet known. In the specification of Figure 12 a), transitions t_2 and t_3 are associated respectively to fabricate and maintain processes. These transitions are labeled by "bottom" since their behaviors are unknown.

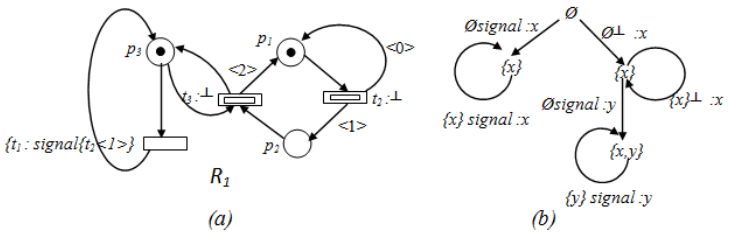

Figure 12. Modelling of flame cutting machine in step 01

The abstract maximality labeled transition system corresponding to this refinable recursive Petri net is shown by Figure 12 b).

Verification

Using the CTL logic in the context of the maximality semantics where actions represent activities has been studied in [16]. Actions are considered as atomic propositions. Then an action name a in a formula associated to a given state means that action a may be in execution at this state. At this level of abstraction we can, for example, verify that always after each execution of the undefined process related to the transition *fabricate* we can launch this process again. This property is expressed in CTL logic as follows: $AG(\perp => \perp)$.

Following the semantics of CTL formula in the context of the maximality semantics, the formula $AG(\perp => \perp)$ means that a not well known activity at a given state leads to the execution of this activity again. Following the refinement process, this activity will be specified in the following refinement steps.

Step 02: In this step we label the abstract transition t_2 by the action *fabricate*, so now we will give details of this abstract transition. This is done by refining the transition t_2 in Petri net R_1 by the process described in Petri net R_2. $\rho\left(t_2, R_1, R_2, \{0\}, \{1\}, \langle p_4\rangle, \{\gamma_0 = \{M/ \mid M(p_6) \mid\geqslant 1\}\}\right) = R_3$

The abstract maximality labeled transition system of this refinable recursive Petri net is shown by Figure 14.

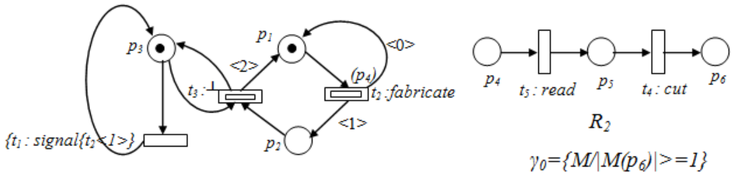

Figure 13. Modelling of flame cutting machine in step 02

Verification

- After each execution of process "fabricate" we can again lunch it.

$$AG\,(\mathit{finished}\,(\mathit{fabricate}) \Longrightarrow \mathit{admitted}\,(\mathit{fabricate}))\,.$$

- When a problem of cutting appears, the undefined process $\perp$ corresponding to the transition maintain will be automatically launched.

$$AF\,(\mathit{signaler} \Longrightarrow EX\perp)\,.$$

- When a problem of cutting appears, the process fabricate will be preempted.

$$AG\,(\mathit{signaler} \Longrightarrow \mathit{finished}\,(\mathit{fabricate}))\,.$$

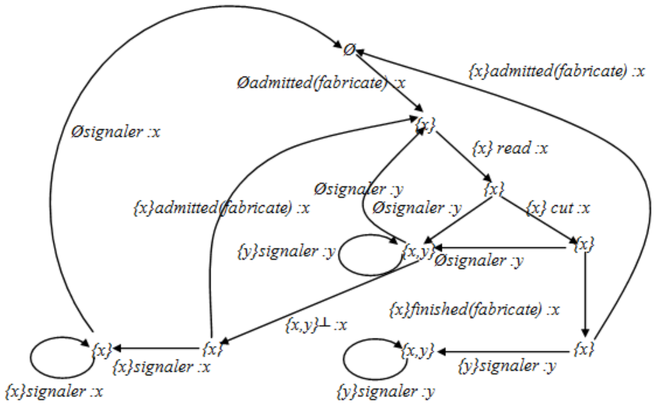

Figure 14. Abstract maximality labeled transition system in step 2

Step 03: Now we give details of the process corresponding to the abstract transition t_3, it will be labeled by the action *maintain*.

$$\rho\,(t_3, R_3, R_4, \{2\}\,, \emptyset, \langle p_7\rangle\,, \{\gamma_2 = \{M/\mid M\,(p_9)\mid \geqslant 1\}\}) = R_5.$$

The maximality labeled transition system obtained by applying the proposed approach consists of 12 states and 19 transitions. Due to its size it cannot be depicted in this paper.

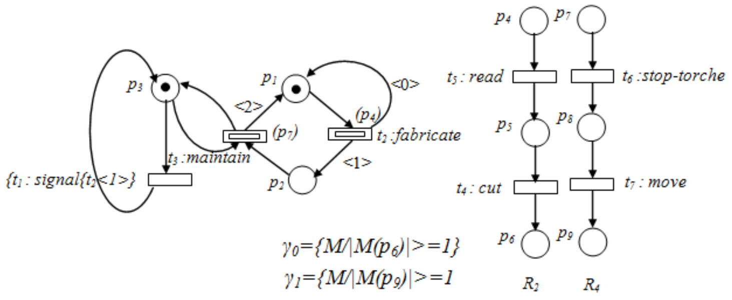

Figure 15. Modelling of flame cutting machine in step 03

Verification

All proprieties which are verified in step 2 are still to be verified in this step 3. In addition we can verify other properties like:

- When an error will occur while the process *fabricate* is running the flame cutting will be stopped.

$$AG\,((signaler\ \ and\ \ finished\,(fabricate)) \Longrightarrow EG\ \ stop\text{-}torche)$$

7 CONCLUSION

In this paper, we have proposed a new approach to modelling concurrent systems by defining a new model named refinable recursive Petri nets, which permits a hierarchical design, so the functionalities and features of systems can be added gradually. Also we have proposed an operational method for generating a maximality labeled transition system associated to the refinable recursive Petri nets. This will make it possible to benefit from the developed results of verification around the model of maximality labeled transition systems. For this fact, the properties related to the good performance of a system specified by a refinable recursive Petri net can be checked on its corresponding maximality labeled transition system. It should be noted that the structure of the maximality labeled transition system represents, in a natural way, the parallel execution of actions, as well as the parallel execution of threads.

REFERENCES

[1] Aceto, L.—Hennessy, M.: Adding Action Refinement to Finite Process Algebra. In: Albert, J. L., Monien, B., Artalejo, M. R. (Eds.): Automata, Languages and Programming (ICALP '91). Springer, Lecture Notes in Computer Science, Vol. 510, 1991, pp. 506–519.

[2] Andrews, D.—Groote, J.—Middelburg, C. (Eds.): Semantics of Specification Languages (SoSL). Springer, London, Workshops in Computing, 1993.

[3] Best, E.—Devillers, R.—Kiehn, A.—Pomello, L.: Concurrent Bisimulations in Petri Nets. Acta Informatica, Vol. 28, 1991, pp. 231–264, doi: 10.1007/BF01178506.

[4] Boudol, G.—Castellani, I.: Concurrency and Atomicity. Theoretical Computer Science, Vol. 59, 1988, No. 1-2, pp. 25–84, doi: 10.1016/0304-3975(88)90096-5.

[5] Courtiat, J. P.—Saidouni, D. E.: Action Refinement in LOTOS. In: Danthine, A., Leduc, G., Wolpe, P. (Eds.): Protocol Specification, Testing and Verification (PSTV '93). North-Holland, 1994, pp. 341–354.

[6] Darondeau, P.—Degano, P.: Causal Trees. In: Ausiello, G., Dezani-Ciancaglini, M., Della Rocca, S. R. (Eds.): Automata, Languages and Programming (ICALP '89). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 372, 1989, pp. 234–248.

[7] Degano, P.—Gorrieri, R.: Atomic Refinement in Process Description Languages. In: Tarlecki, A. (Ed.): Mathematical Foundations of Computer Science (MFCS 1991). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 520, 1991, pp. 121–130.

[8] Devillers, R.: Maximality Preservation and the ST-Idea for Action Refinement. In: Rozenberg, G. (Ed.): Advances in Petri Nets. Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 609, 1992, pp. 108–151, doi: 10.1007/3-540-55610-9_170.

[9] Courtiat, J.-P.—Saïdouni, J.-E.: Relating Maximality-Based Semantics to Action Refinement in Process Algebras. In: Hogrefe, D., Leue, S. (Eds.): IFIP TC6/WG6.1, 7th International Conference on Formal Description Techniques (FORTE '94), Chapman, Hall, IFIP Conference Proceedings 6, 1994, pp. 293–308.

[10] Janssen, W.—Poel, M.—Zwiers, J.: Action Systems and Action Refinement in the Development of Parallel Systems. In: Baeten, J. C. M., Groote, J. F. (Eds.): CONCUR '91. Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 527, 1991, pp. 298–316.

[11] Saidouni, D. E.—Courtiat, J. P.: Syntactic Action Refinement in Presence of Multiway Synchronization. Semantics of Specification Languages (SoSL), 1994, pp. 289–330, doi: 10.1007/978-1-4471-3229-5_16.

[12] Van Glabbeek, R. J.: The Refinement Theorem for ST-Bisimulation Semantics. IFIP Working Conference on Programming Concepts and Methods, North-Holland, 1990.

[13] Saidouni, D. E.: Maximality Semantic: Application to Actions Refinement in LOTOS. Ph.D. thesis, LAAS-CNRS, 7 av. du Colonel Roche, 31077 Toulouse Cedex France, 1996 (in French).

[14] Saidouni, D. E.—Belala, N.—Bouneb, M.—Boudjadar, A.—Ouchène, B.: Using Maximality-Based Labeled Transitions as Model for Petri Nets. The International Arab Conference on Information Technology (ACIT '08), 2008.

[15] Saidouni, D. E.—Belala, N.—Bouneb, M.: Aggregation of Transitions in Marking Graph Generation Based on Maximality Semantics for Petri Nets. Proceedings

of the Second International Conference on Verification and Evaluation of Computer and Communication Systems (VECoS '08), 2008, pp. 6–16.

[16] Saidouni, D. E.—Belala, N.—Bouneb, M.: Using Maximality-Based Labelled Transitions as Model for Petri Nets. The International Arab Journal of Information Technology (IAJIT), Vol. 6, 2009, No. 5, pp. 440–446.

[17] Saidouni, D. E.—Belala, N.—Bouneb, M.: Maximality-Based Structural Operational Semantics for Petri Nets. 2nd Mediterranean Conference on Intelligent Systems and Automation (CISA), 2009.

[18] Saidouni, D. E.—Bouneb, M.—Ilie, J. M.: Maximality Semantic for Recursive Petri Net. Proceedings of 27th Europeen Conference on Modelling and Simulation (ECMS '13). 2013, pp. 544–550. ISBN 978-0-9564944-7-4, doi: 10.7148/2013-0544.

[19] Buchholz, P.: Hierarchical High Level Petri Nets for Complex System Analysis. Computer Science IV, Dortmund university, D-44221, Dortmund Germany, 1994, doi: 10.1007/3-540-58152-9_8.

[20] Cheung, K.-S.—Chow, P. K.-O.: A Petri-Net Approach to Refining Object Behavioural Specifications. Informatica, Vol. 33, 2009, No. 2, pp. 221–232.

Messaouda Bouneb received her B.Eng. degree from the University of Mentouri Constantine, Algeria (2005). In February 2009, she received her M.Sc. degree in computer science from the University of El Arbi Ben-M'hidi Oum El-Bouaghi, Algeria. Her research domain is formal specification and verification of real-time systems using Petri nets.

Djamel Eddine Saidouni received his B.Eng. degree from the University of Mentouri Constantine, Algeria (1990). He received his Ph.D. in theoretical computer science from the University of Paul Sabatier, Toulouse, France (1996). His domain research is the formal specification and verification of complex distributed and real time systems.

Jean Michel Ilie received several degrees in electronics and informatics along with his Ph.D. thesis from the UPMC University of Paris (1990). Currently, he is a member of the Paris Descartes University in its conference on a master higher grade (2009), he is also Permanent Researcher of the LIP6 laboratory, UPMC. The fields of his research concern the formal validation of complex embedded systems.