Electronic Communications of the EASST
Volume 2 (2006)

Proceedings of the Workshop on Petri Nets and Graph Transformation (PNGT 2006)

A basic tool for the modeling of Marked-Controlled Reconfigurable Petri Nets

Marisa Llorens and Javier Oliver

13 pages

Guest Editors: Paolo Baldan, Hartmut Ehrig, Julia Padberg, Grzegorz Rozenberg
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

A basic tool for the modeling of Marked-Controlled Reconfigurable Petri Nets

Marisa Llorens[1] **and Javier Oliver**[2]

[1] mllorens@dsic.upv.es, [2] fjoliver@dsic.upv.es
Departamento de Sistemas Informáticos y Computación (DSIC)
Universidad Politécnica de Valencia (UPV), Valencia, Spain

Abstract: In previous studies, we have introduced *marked-controlled net rewriting systems* and a subclass of these called *marked-controlled reconfigurable Petri nets*. In a marked-controlled net rewriting system, a system configuration is described as a Petri net, and a change in configuration is described as a graph rewriting rule. A marked-controlled reconfigurable Petri net is a marked-controlled net rewriting system where a change in configuration amounts to a modification in the flow relations of the places in the domain of the involved rule in accordance with this rule, independently of the context in which this rewriting applies. In both models, the enabling of a rule not only depends on the net topology, but also depends on the net marking according to control places. Even though the expressiveness of Petri nets and marked-controlled reconfigurable Petri nets is the same, with marked-controlled reconfigurable Petri nets, we can easily and directly model concurrent and distributed systems that change their structure dynamically. In this article, we present MCReNet, a tool for the modeling and verification of marked-controlled reconfigurable Petri nets.

Keywords: structural dynamic changes, modeling, verification, tools.

1 Introduction

In [BLO03, Llo03, LO04b], we introduced the model of *net rewriting systems* (NRS) and a subclass of this model, *reconfigurable Petri nets* (RN), to analyze, simulate and verify concurrent and distributed systems that are subject to structural dynamic changes. Both models arise from two different lines of research that were conducted in the field of the Petri net formalism [Mur89, Pet81]. The goal of these lines of research is to enhance the expressiveness of the basic model of Petri nets so that it can support the description of structural dynamic changes in concurrent and distributed systems. The first line covers various proposals for merging Petri nets with *graph grammars* [Bal00, Cor95, Sch94], while the second line, which is best represented by *Valk's self-modifying nets* [Val78, Val81], considers Petri nets whose flow relations can vary at runtime. *Reconfigurable Petri nets* attempt to combine the most relevant aspects of both of these approaches and constitute a class of models for which each of the fundamental properties of Petri nets (place boundedness, reachability, deadlock and liveness) are decidable. The translation of this model into Petri nets is automatic [Llo03, LO04b]. This equivalence ensures that the model is amenable to automatic verification tools. In contrast, the class of *net rewriting systems* is Turing powerful [Llo03, LO04b].

In [LO04a], we introduced *marked-controlled net rewriting systems* (MCNRS) and *marked-controlled reconfigurable Petri nets* (MCRN) as an extension of net rewriting systems and reconfigurable Petri nets, respectively, where the enabling of a rewriting rule not only depends on the net topology, but also depends on the net marking according to some net places named *control places*. In MCNRS, a system configuration is described as a Petri net, and a change in configuration is described as a graph rewriting rule, which consists of replacing part of the system (the part that matches the left-hand side of the rewriting rule) with another one (given by the right-hand side of the rewriting rule). A MCRN is a MCNRS where a change in configuration is limited to the modification of the flow relations of the places in the domain of the rewriting rule involved; i.e., the set of places and transitions is left unchanged by rewriting rules.

We are interested in developing a software tool to analyze the structure and dynamic behavior of systems that are modeled using our nets in order to evaluate them and suggest improvements or changes. The goal of our MCReNet tool is to study real systems that are modeled by MCRN.

Our work is closely related to the topic of net transformation systems. There are some works in this area that perform the modification of the net topology and the modification of the system current marking. These works cover the modification of Petri nets in a categorical instead of a set theoretical way [EGP00]. In [LO04a] a detailed related work with respect to our models is presented. In [LKL05], a hybrid process mining approach to dynamic business process is proposed. The approach is based on reconfigurable Petri nets. This paper gives a method of mining dynamic changes in different instances; finally, a Reconfigurable Workflow Net is generated. In [Akr05], a framework for managing changes in service oriented enterprises (SOEs) is presented. The author presents a taxonomy of changes using a combination of several types of Petri nets to model the triggering changes and ensuing reactive changes. These changes are modeled by reconfigurable Petri nets.

The remainder of the paper is organized as follows. Section 2 recalls marked-controlled reconfigurable Petri nets with an illustrative example and presents two algorithms: the first one (previously defined in a formal way in [LO04a]) translates marked-controlled reconfigurable Petri nets into Petri nets and the second one, in the opposite direction, obtains the current state of the marked-controlled reconfigurable Petri net from its equivalent Petri net. In Section 3, we describe the MCReNet tool in detail. Finally, Section 4 presents our conclusions.

2 Marked-Controlled Reconfigurable Petri nets

This section recalls the model of *marked-controlled reconfigurable Petri nets* introduced in [LO04a].

2.1 Syntax and semantics

Definition 1 (Marked-Controlled Reconfigurable Petri net) A *marked-controlled reconfigurable Petri net (MCRN)* [LO04a] is a structure $N = (P, T, \mathcal{R}, \gamma_0)$, where $P = \{p_1, \ldots, p_n\}$ is a non-empty and finite set of places; $T = \{t_1, \ldots, t_m\}$ is a non-empty and finite set of transitions that is disjoint from P ($P \cap T = \emptyset$); $\mathcal{R} = \{r_1, \ldots, r_h\}$ is a finite set of rewriting rules; and γ_0 is a marked Petri net.

A *rewriting rule* $r \in \mathcal{R}$ is a structure $r = (D, {}^\bullet r, r^\bullet, C, \mathbb{M})$, where:

- $D \subseteq P$ is the *domain* of r;
- ${}^\bullet r : (D \times T) \cup (T \times D) \to \mathbb{N}$ and $r^\bullet : (D \times T) \cup (T \times D) \to \mathbb{N}$ are the *preconditions* and *postconditions* of r, (i.e. they are the flow relations of the domain places before and after the change in configuration due to rule r);
- C is a subset of places of D, $C \subseteq D$, called *control places*, and
- $\mathbb{M}$ is the required minimum marking of places of C so that the rule can be enabled.

A *configuration* of a MCRN is a Petri net $\Gamma = (P, T, F)$.
A *state* γ of a MCRN is a marked Petri net $\gamma = (\Gamma, M)$. The state γ_0 is called the *initial state* of the MCRN.
The *events* of a MCRN are the transitions and the rewriting rules: $E = T \cup \mathcal{R}$.

We represent a rewriting rule using formal sums notation as

$$r = \sum_{p \in D} p(\sum_{t \in T} {}^\bullet r(p,t) \cdot t - \sum_{t \in T} {}^\bullet r(t,p) \cdot t) \rhd \sum_{p \in D} p(\sum_{t \in T} r^\bullet(p,t) \cdot t - \sum_{t \in T} r^\bullet(t,p) \cdot t)$$

Definition 2 (Configuration Graph of a MCRN) The *configuration graph* $G(N)$ of a MCRN $N = (P, T, \mathcal{R}, \gamma_0)$ is the labeled directed graph whose nodes are the configurations, such that there is an arc from configuration Γ to configuration Γ' labeled with rule $r = (D,\ {}^\bullet r,\ r^\bullet,\ C,\ \mathbb{M}) \in \mathcal{R}$, which we denote $\Gamma[r\rangle\Gamma'$, if and only if the following holds:

$$\forall p \in C :\ M(p) \geq \mathbb{M}(p)$$

$$\forall p \in D :\ \begin{cases} F(p,t) = {}^\bullet r(p,t) \text{ and } F(t,p) = {}^\bullet r(t,p) \\ F'(p,t) = r^\bullet(p,t) \text{ and } F'(t,p) = r^\bullet(t,p) \end{cases}$$

$$\forall p \notin D :\ F(p,t) = F'(p,t) \text{ and } F(t,p) = F'(t,p)$$

Notice that we require the control places to be marked with at least marking $\mathbb{M}$. The transition relation must contain arcs of the exact multiplicity appearing in the left-hand side of the rewriting rule, and we do not allow rewriting if arcs of a greater multiplicity are present.
The dynamic evolution of a MCRN is then given by its state graph.

Definition 3 (State Graph of a MCRN) The *state graph* of a MCRN $N = (P, T, \mathcal{R}, \gamma_0)$ is the labeled directed graph whose nodes are states of N and whose arcs (labeled with events) are of two kinds:

1. *firing of a transition:* arc from state (Γ, M) to (Γ, M') that is labeled with transition t when t can fire in the net Γ at marking M and leads to M', and
2. *change in configuration:* arc from state (Γ, M) to state (Γ', M) that is labeled with rule $r \in \mathcal{R}$ if $\Gamma[r\rangle\Gamma'$ is a transition of the configuration graph of N.

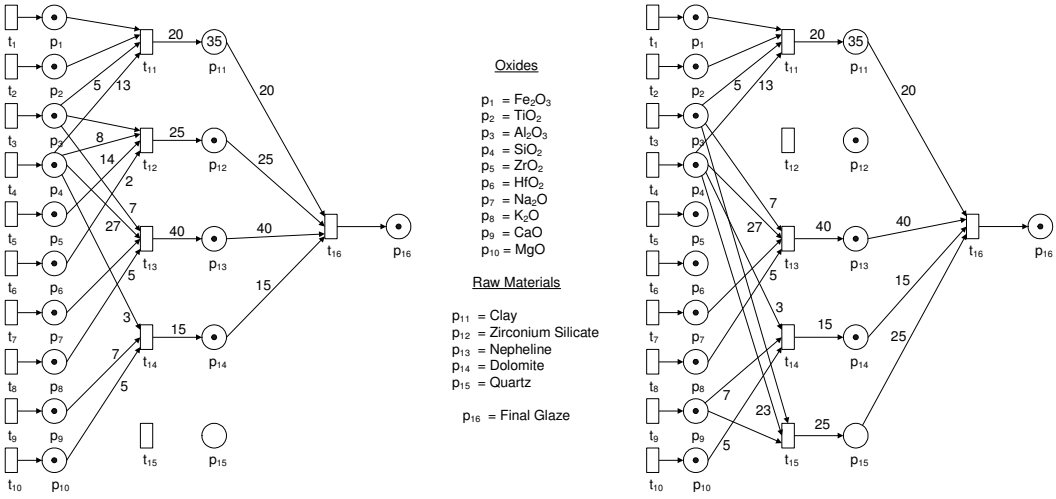
In other words, the set of labeled arcs of the state graph of N is given by

$$\{(\Gamma, M) \xrightarrow{t} (\Gamma, M') | M[t\rangle M' \text{ in } \Gamma\} \cup \{(\Gamma, M) \xrightarrow{r} (\Gamma', M) | \Gamma[r\rangle\Gamma' \text{ in } G(N)\}.$$

2.2 Example

The following example shows a system modeled by a MCRN, and a change in configuration that depends on both the net topology and the net marking. We have chosen an example that is related to manufacturing processes since, in these types of problems, changes in the structure are very frequent and they usually depend on the state of the manufacturing processes.

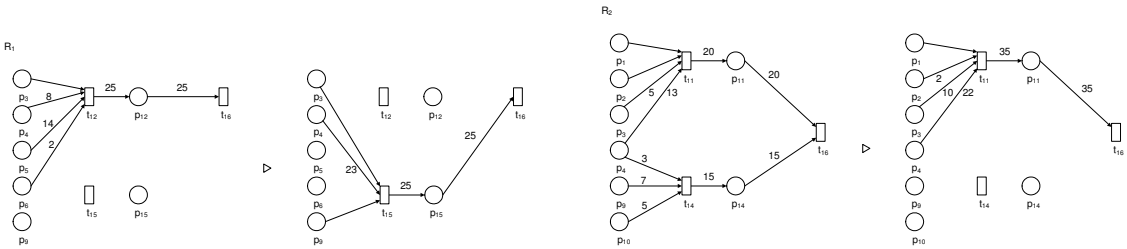
Example 1 *(Glaze Manufacture)* *In the manufacturing process of a glaze, different glazes are made by using different raw materials that, after firing, have certain characteristics.*



(a) The initial state

(b) The new state reached after applying rule R_1 to the state in Fig. 1(a)

Figure 1: Two states of the MCRN of Example 1



(a) Rule R_1

(b) Rule R_2

Figure 2: Rewriting rules R_1 and R_2 of the MCRN of Example 1

Figure 1(a) is the initial state $\gamma_0 = (\Gamma_0, M_0)$ *of a MCRN that represents a glaze with the following composition: 20% clay, 25% zirconium silicate, 40% nepheline, and 15% dolomite. Because of its oxide composition, this glaze is bell-glazed and opaque, water does not mark it, and it has a low lineal dilatation coefficient (LDC). If we substitute the zirconium silicate by quartz, the*

opaque glaze turns into a semi-opaque glaze with a high LDC. If we remove all of the dolomite and increase the clay, we obtain a glaze that water marks and that is disk-glazed. Changing the composition of the glaze in our model means changing the net topology. To make these changes possible, we define two rewriting rules using the formal sums notation previously introduced. We show them (R_1 and R_2) graphically in Fig. 2(a) and Fig. 2(b), respectively. The subset of control places C is empty for rule R_1, whereas, for rule R_2, this subset is $C = \{p_{11}\}$ and $\mathbb{M}(p_{11}) = 35$ (i.e., to remove all of the dolomite and increase the clay, at least 35% clay is needed in place p_{11}). Therefore, the MCRN consists of 16 places, 16 transitions, and 2 rewriting rules.

$$R_1: p_3(-t_{12})+p_4(-8t_{12})+p_5(-14t_{12})+p_6(-2t_{12})+p_9(\emptyset)+p_{12}(25t_{12}-25t_{16})+p_{15}(\emptyset)$$
$$\triangleright$$
$$p_3(-t_{15})+p_4(-23t_{15})+p_5(\emptyset)+p_6(\emptyset)+p_9(-t_{15})+p_{12}(\emptyset)+p_{15}(25t_{15}-25t_{16})$$

$$R_2: p_1(-t_{11})+p_2(-t_{11})+p_3(-5t_{11})+p_4(-13t_{11}-3t_{14})+p_9(-7t_{14})+p_{10}(-5t_{14})+p_{11}(20t_{11}-20t_{16})+p_{14}(15t_{14}-15t_{16})$$
$$\triangleright$$
$$p_1(-t_{11})+p_2(-2t_{11})+p_3(-10t_{11})+p_4(-22t_{11})+p_9(\emptyset)+p_{10}(\emptyset)+p_{11}(35t_{11}-35t_{16})+p_{14}(\emptyset)$$

Figure 1(b) shows the state that is reached when rule R_1 (in Fig. 2(a)) is applied to the initial state represented in Fig. 1(a). In this new state, all the zirconium silicate has been substituted by quartz.

There are only four possible configurations. The initial configuration Γ_0 and the configuration Γ_1 are the configurations of the states represented in Fig. 1(a) and Fig. 1(b), respectively.

2.3 Relationship between Petri nets and MCRNs

In [LO04a], we prove that MCRNs are equivalent to Petri nets, but they provide somewhat more compact representations of concurrent and distributed systems whose structure evolves at run-time. The substantial increase in the number of transitions with respect to the MCRN implies that the size of the Petri net grows considerably. This can be observed in Fig. 4 that shows the Petri net (20 places, 72 transitions) equivalent to the MCRN of Example 1. Watching this figure, we can imagine how big the entire Petri net is and how we can best model the system with a MCRN.

The Petri net $\widetilde{N} = (\widetilde{P}, \widetilde{T}, \widetilde{F}, \widetilde{M})$ equivalent to a MCRN $N = (P, T, \Re, \gamma_0)$ is obtained following Algorithm 1, that is completely defined in a formal way in [LO04a]. The set of places $\widetilde{P}$ is $P \cup \{q_0, \ldots, q_k\}$, where P is the set of places of the original MCRN and each place q_i is one specific place attached to each possible configuration $\Gamma_i \in G(N)$. The marking $\widetilde{M}$ for places of the subset P is the marking that they have in the initial state γ_0 of the MCRN N and for the added places $\{q_0, \ldots, q_k\}$ is 0 except for place q_0 whose marking is 1. This marking shows that Γ_0 is the current configuration.

Now, we have developed an algorithm that acts in the other direction. That is, it is possible to extract the state $\gamma_i = (\Gamma_i, M_i)$ of a MCRN from the corresponding marked Petri net $\widetilde{N} = (\widetilde{P}, \widetilde{T}, \widetilde{F}, \widetilde{M})$, at a given moment, following Algorithm 2. This algorithm starts from a marked Petri net, whose topology is the result of Algorithm 1 and whose marking is the result of some transition firing sequence applied to the Petri net obtained from Algorithm 1. Taking these facts as a

Algorithm 1 Petri Net equivalent to a Marked-Controlled Reconfigurable Petri net

1: We start from the initial configuration Γ_0. For Γ_0, we obtain all the possible configurations due to the firings of all enabled rewriting rules, that is, we obtain all the nodes of the first level of the configuration graph $G(N)$. If we imagine places and transitions of a configuration as if they were located in two different parallel planes, i.e., a plane with the set of places and a plane with the set of transitions connected by the flow relations of the represented configuration, we will have only one plane of places, and as many planes of transitions as there are different configurations.
2: For each plane of transitions we add a place q_i, connected by an input arc and an output arc with all transitions of the plane.
3: We also add a transition r_{0i} between place q_0 of the initial configuration and each plane q_i of each reachable configuration from q_0; this represents the change in configuration due to rule r, $\Gamma_0[r\rangle\Gamma_i$.
4: If there are control places involved in the change in configuration $\Gamma_0[r\rangle\Gamma_i$, we connect transition r_{0i} with each control place of rule r (in the plane of places) using an input arc and an output arc whose weight is the required minimum marking of the control place for the rule to be enabled.
5: We repeat this process for each configuration that is distinct from the initial configuration, that is, for each plane of transitions obtained from the initial plane.

starting point, it is not very difficult to distinguish the subnet that represents the current state of the initial MCRN (the input of Algorithm 1).

Theorem 1 *It is possible to extract the state $\gamma_i = (\Gamma_i, M_i)$ of a MCRN from the corresponding marked Petri net $\widetilde{N} = (\widetilde{P}, \widetilde{T}, \widetilde{F}, \widetilde{M})$, at a given moment.*

Proof. The corresponding marked Petri net $\widetilde{N}$ has been obtained from Algorithm 1. It is straightforward to extract the state $\gamma_i = (\Gamma_i, M_i)$ of a MCRN following Algorithm 2. □

3 The MCReNet Software Tool

To study real systems that are modeled by MCRN, it would be useful to have a software tool to analyze the structure and dynamic behavior of the modeled system in order to evaluate it and suggest improvements or changes. The MCRN model is equivalent to the Petri net model and, therefore, the decidable properties of Petri nets are still decidable for our model. It is possible to obtain an automatic verification tool for MCRN. Our objective is to develop a tool to design and verify systems modeled by MCRN. This tool has a graphical editor, a simulator and an analyzer for several common properties.

Since there are numerous automatic design and verification tools based on Petri nets for both business and academic use [Webb], and the fact that Petri nets and MCRN are equivalent, we have developed a software tool that translates a MCRN into its equivalent Petri net. We start from the following premises:

Algorithm 2 State of a Marked-Controlled Reconfigurable Petri Net from the corresponding Petri Net

1: We start from the marked Petri net $\widetilde{N} = (\widetilde{P}, \widetilde{T}, \widetilde{F}, \widetilde{M})$ where $\widetilde{P}$ is the set of places $\widetilde{P} = P \cup \{q_0, \ldots, q_k\}$, $\widetilde{T}$ is the set of transitions $\widetilde{T} = (\{q_0, \ldots, q_k\} \times T) \cup \widetilde{R}$, $\widetilde{F}$ is the flow relation and $\widetilde{M}$ is the marking of the net.

2: At marking $\widetilde{M}$, $\sum_{j=0}^{k} \widetilde{M}(q_j) = 1$, that is, only one place q_i of the subset of places $\{q_0, \ldots, q_k\}$ is marked with one token. From the subset of transitions $\{q_0, \ldots, q_k\} \times T \in \widetilde{T}$ only the transitions in $\{q_i\} \times T$ are enabled.

3: The current state $\gamma_i = (\Gamma_i, M_i)$ of MCRN is composed by:

- the current configuration $\Gamma_i = (P_i, T_i, F_i)$ where the set of places is $P_i = P$, the set of transitions is $T_i = \{q_i\} \times T$, and the flow relation F_i is given by

$$\forall p \in P_i, \forall t \in T_i : \begin{cases} F_i(p,t) = \widetilde{F}(p,(q_i,t)) \\ F_i(t,p) = \widetilde{F}((q_i,t),p) \end{cases}$$

- the current marking M_i is $M_i(p) = \widetilde{M}_i(p) \quad \forall p \in P$

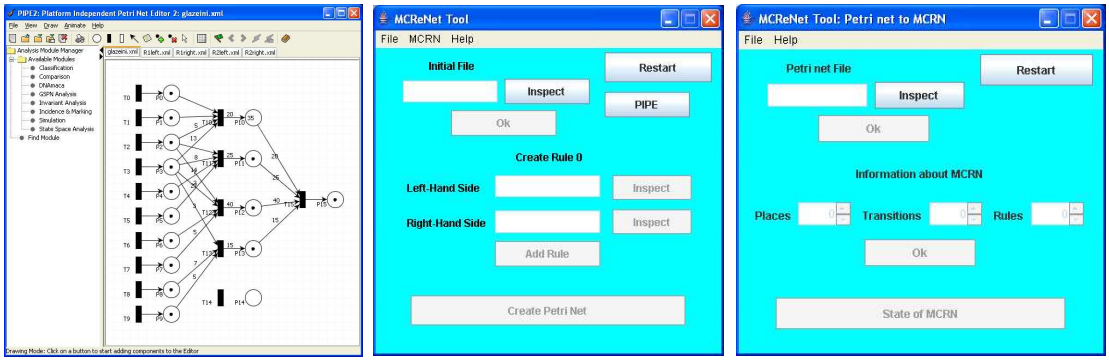
- both the initial state and the rewriting rules of a MCRN are Petri nets;
- a MCRN is formally equivalent to a Petri net;
- an algorithm that obtains an equivalent Petri net from a MCRN is available (Algorithm 1).

The idea is to use an already existing graphical editor of Petri nets to draw the initial MCRN, i.e., a Petri net. Our tool will use this net:

1. To make a syntactical analysis of the files generated by the editor and to store the relevant information in the suitable data structures.
2. To implement the algorithm to translate a MCRN into an equivalent Petri net.
3. To create a file (from the resulting Petri net) that can be viewed from the editor.

We have chosen PIPE (*Platform Independent Petri net Editor*) [Webc, BCC+04, Akh05], a free software for non business use that allows the design, analysis, and simulation of Petri nets. This editor allows us to create, save and load Petri nets according to the last standard XML for Petri nets, PNML (*Petri Net Markup Language* [Webd]).

The user draws the initial state of the MCRN, and the several rewriting rules using PIPE. Figure 3(a) shows the user interface of PIPE, which basically consists of a pane for the analysis modules and a graphical editing pane. The active window shows the initial state of the MCRN



(a) Initial state of the MCRN for Example 1 in PIPE (b) MCReNet Tool (c) MCReNet Tool 2

Figure 3: Main screens of PIPE and MCReNet Tool

for Example 1. The background windows contain the left and right-hand sides of the rewriting rules.

Various XML files are obtained (one for the initial state, and two for each one of the rewriting rules). These XML files have a well-defined structure where the division between places, transitions, and arcs is clearly visible. Any file of this type has a headline that shows what language is used, here PNML [Webd]. Next, the name and the type of the represented net appear.

```
<?xml version="1.0" encoding="iso-8859-1" ?> <pnml>
    <net id="Net-One" type="P/T net">
```

For every place appears an identifier, the graphic coordinates, a label, and the initial marking. For every transition appears an identifier, the graphic coordinates, a label, the orientation, a rate, and if it is timed or not.

```
<place id="P0">
    <graphics>
        <position x="90.0" y="60.0" />
    </graphics>
    <name>
        <value>P0</value>
        <graphics />
    </name>
    <initialMarking>
        <value>1</value>
        <graphics>
            <offset x="0.0" y="0.0" />
        </graphics>
    </initialMarking>
</place>
```

```
<transition id="T0">
    <graphics>
        <position x="30.0" y="60.0" />
    </graphics>
    <name>
        <value>T0</value>
        <graphics />
    </name>
    <orientation>
        <value>0</value>
    </orientation>
    <rate>
        <value>1.0</value>
    </rate>
    <timed>
        <value>false</value>
    </timed>
</transition>
```

For every arc it is indicated if it is an arc from a transition to a place or from a place to a transition (source and target), the arc weight, the begin and end coordinates, and if it is a curved arc or not.

```
<arc id="T0 to P0" source="T0" target="P0">
    <graphics />
    <inscription>
        <value>2</value>
        <graphics />
    </inscription>
    <arcpath id="000" x="45" y="71"
     curvePoint="false" />
    <arcpath id="001" x="86" y="71"
     curvePoint="false" />
</arc>
```

```
<arc id="P0 to T0" source="P0" target="T0">
    <graphics />
    <inscription>
        <value>5</value>
        <graphics />
    </inscription>
    <arcpath id="000" x="115" y="70"
     curvePoint="false" />
    <arcpath id="001" x="156" y="70"
     curvePoint="false" />
</arc>
```

After this information, the headline labels are closed.

```
    </net>
</pnml>
```

Figure 3(b) shows the initial screen of the MCReNet Tool. User introduces the XML files that represent the initial state and the left-hand and right-hand side of all rules from this screen.

A syntactical analysis of these files is made in order to extract the information from them. The name and its marking are obtained for each place. The name is obtained for each transition. The weight and the places and transitions that the arc connects are obtained for each arc (the incidence matrix of the net [Mur89]). A consistency check is made, that is, it is checked whether the initial net together with pairs of nets representing rules create a MCRN (whether they differ only in the flow relation). Since both the left-hand side and the right-hand side of the rules are Petri nets with identical places and identical transitions, besides knowing its names, we are interested in knowing the flow relation between them in each part of the rule (the incidence matrix of the left-hand side and the incidence matrix of the right-hand side of the rewriting rule).

The process of translation begins when user presses the **Create Petri net** button. The translator that obtains the Petri net that is equivalent to the initial MCRN is implemented in Java [Weba] following Algorithm 1. The first step of this algorithm is to obtain the configuration graph $G(N)$ of the initial MCRN. We obtain a procedure that (from the initial state and the rewriting rules) checks which rules are enabled and it obtains the resulting configurations of the firings of these rules. The process is repeated for each one of the obtained configurations, as long as new configurations are obtained and until no more rules are firable. All the possible configurations of the MCRN must be stored indicating which configurations are immediately reachable from a given configuration, due to the firing of which rules.

Once $G(N)$ is obtained, the tool is ready to translate this net into its equivalent Petri net. According to Algorithm 1, the translation process consists in adding new places, new transitions, and their corresponding flow relations to the places and transitions of the initial configuration. Since the number of places and transitions of the equivalent Petri net is known [LO04a], the dimensions of the incidence matrices can be known. The previous and posterior incidence matrices are filled according to the flow relations of the configurations of $G(N)$. A file with the same format as the PIPE files is obtained from the resulting Petri net. Therefore, this Petri net can be displayed on the PIPE editor (see Fig. 4).

After obtaining the equivalent Petri net, the process of analysis and simulation can begin. The PIPE editor itself offers the possibility of doing simulation and analysis of properties. Figures 5(a), 5(b) and 5(c) show some of these analyses applied to the equivalent Petri net in Fig. 4; specifically: (a) the Petri net classification results based on the connectivity between places and

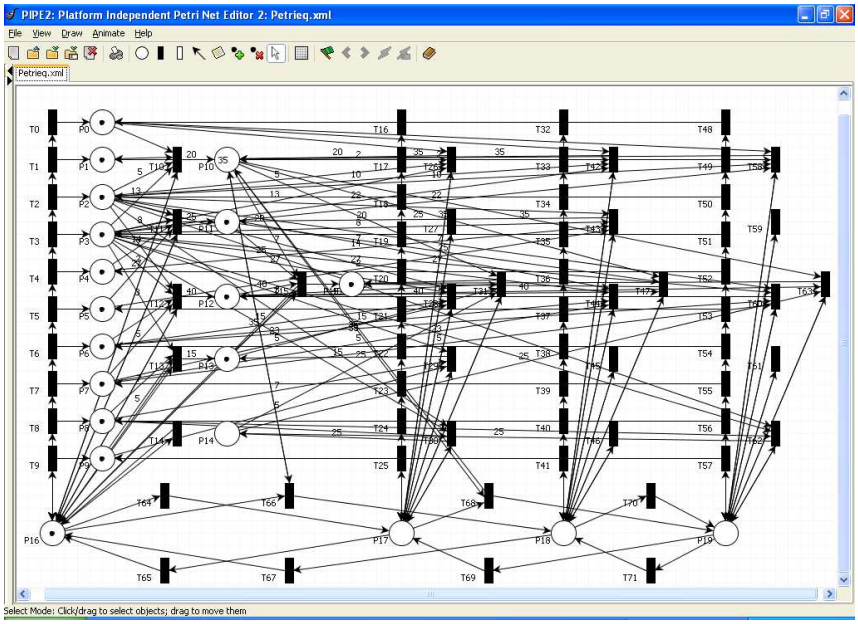
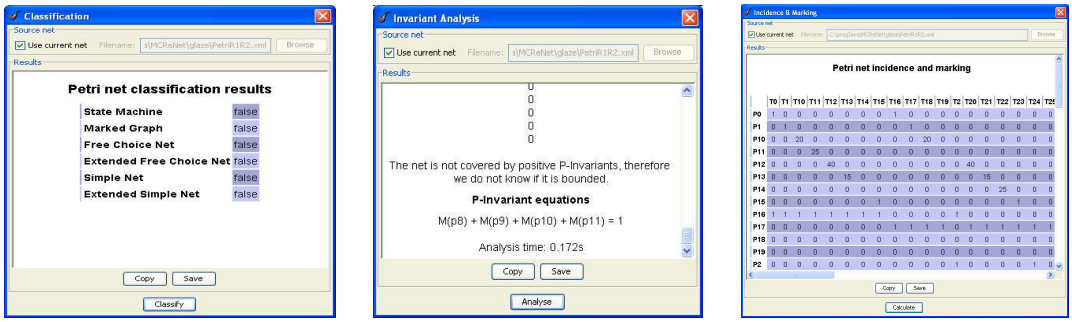


Figure 4: Petri net equivalent to the MCRN of Example 1 in PIPE

transitions, (b) the invariant analysis and (c) the incidence and marking matrices and the enabled transitions set.



(a) The classification results (b) The invariant analysis (c) The incidence matrices

Figure 5: Some analyses of the Petri net of Fig. 4

We have also implemented the Algorithm 2 that obtains the state of a MCRN from the corresponding Petri net in a given state. The idea is that users can animate the Petri net (firings of enabled transitions) and, at any time, they can find and see the equivalent state of the equivalent MCRN. The data capture for Algorithm 2 is performed from the screen of the MCReNet tool showed in Fig. 3(c). After introducing the Petri net file and checking that the number of places and transitions of the MCRN are correct, when user presses the **State of MCRN** button, the current state of MCRN is obtained, as a PIPE file, following Algorithm 2.

4 Conclusions and further work

We have developed a tool to analyze, simulate, and verify systems that are modeled with MCRN. On the basis of the equivalence between Petri nets and MCRN, we have initially opted to develop a tool that exploits the capabilities of the Petri net tools that are currently available. The main goal of our tool is the implementation of a translation algorithm of a MCRN into an equivalent Petri net (Algorithm 1). We have chosen PIPE as the graphical editor, simulator and analyzer. MCReNet integrates PIPE with the translator. Therefore, user can draw the initial state and rewriting rules of the MCRN, which is then translated into its equivalent Petri net. In addition, a simulation and an analysis of properties of this net can be performed. All these processes are integrated in the MCReNet tool. The other goal of our tool is the implementation of Algorithm 2 to obtain the current state of a MCRN from its equivalent Petri net at a given moment.

As further work, we will develop a tool that is completely autonomous. Our new tool will be able to directly carry out the editing, simulation and analysis of net properties on MCRN. We are currently working on the theoretical aspects of the analysis methods for Petri nets in order to apply them directly to MCRN. Also we are studying the relationships between our approaches and other existing proposals using rewriting techniques for providing a reconfiguration mechanism for Petri nets (like, e.g., open Petri nets [BCE+06] and high-level replacement systems applied to Petri nets [PER95, HEM05]).

Acknowledgements: This work has been partially supported by the EU (FEDER) and the Spanish MEC under grant TIN2005-09207-C03-02 and by the ICT for EU-India Cross-Cultural Dissemination Project ALA/95/23/2003/077-054.

Bibliography

[Akh05] N. Akharware. PIPE2: Platform Independent Petri Net Editor. Msc. project report, Department of Computing, Imperial College, London, 2005.

[Akr05] M. S. Akram. Managing Changes to Service Oriented Enterprises. Master's thesis, Virginia Polytechnic Institute and State University, USA, 2005.

[Bal00] P. Baldan. *Modelling Concurrent Computations: From Contextual Petri Nets to Graph Grammars*. Phd thesis, Computer Science Department, University of Pisa, Italy, 2000.

[BCC+04] T. Barnwell, M. Camacho, M. Cook, M. Gready, P. Kyme, M. Tsouchlaris. PIPE. Platform Independent Petri-net Editor. Final report, Petri Net Analyser - Group 4. Department of Computing, Imperial College, London, 2004.

[BCE+06] P. Baldan, A. Corradini, H. Ehrig, R. Heckel, B. Knig. Bisimilarity and Behaviour-Preserving Reconfigurations of Open Petri Nets. Technical report CS-2006-09, Dip. di Informatica, Universit Ca' Foscari Di Venezia, November 2006.

[BLO03] E. Badouel, M. Llorens, J. Oliver. Modelling Concurrent Systems: Reconfigurable Nets. In Arabnia and Mun (eds.), *Int. Conf. on Parallel and Distributed Processing Techniques and Applications (PDPTA'03)*. Volume IV, pp. 1568–1574. CSREA Press, Las Vegas, Nevada (USA), June 23-26 2003.

[Cor95] A. Corradini. Concurrent Computing: From Petri Nets to Graph Grammars. *Electronic Notes in Theoretical Computer Science* 2, 1995. Invited talk at the *Joint COMPUGRAPH/SEMAGRAPH Workshop on Graph Rewriting and Computation*. http://www.elsevier.nl/locate/entcs/volume2.html

[EGP00] H. Ehrig, M. Gajewsky, F. Parisi-Presicce. High-Level Replacement Systems Applied to Algebraic Specifications and Petri Nets. In *Handbook of Graph Grammars and Computing by Graph Transformation, Vol. III: Concurrency, Parallelism and Distribution*. Pp. 341–400. World Scientific, 2000.

[HEM05] K. Hoffmann, H. Ehrig, T. Mossakowski. High-Level Nets with Nets and Rules as Tokens. In Ciardo and Darondeau (eds.), *Proc. 26th Int. Conf. on Applications and Theory of Petri Nets (ICATPN'05)*. Lecture Notes in Computer Science 3536, pp. 268–288. Springer, Miami, USA, June 20-25 2005.

[LKL05] N. Li, J. Kang, W. Lv. A Hybrid Approach for Dynamic Business Process Mining Based on Reconfigurable Nets and Event Types. In *Proc. IEEE Int. Conf. on e-Businees Engineering (ICEBE'05)*. Pp. 289–294. IEEE Computer Society, Beijing, Peoples Rep. China, 2005.

[Llo03] M. Llorens. *Redes Reconfigurables. Modelización y Verificación*. Phd thesis (in spanish), Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, Spain, 2003.

[LO04a] M. Llorens, J. Oliver. Introducing Structural Dynamic Changes in Petri Nets: Marked-Controlled Reconfigurable Nets. In *Proc. Int. Conf. on Automated Technology for Verification and Analysis (ATVA'04)*. Lecture Notes in Computer Science 3299, pp. 310–323. Springer Verlag, Taipei, Taiwan, 2004.

[LO04b] M. Llorens, J. Oliver. Structural and Dynamic Changes in Concurrent Systems: Reconfigurable Petri Nets. *IEEE Transactions on Computers* 53(9):1147–1158, September 2004.

[Mur89] T. Murata. Petri Nets: Properties, Analysis and Applications. In *Proc. of the IEEE*. Volume 77(4), pp. 541–580. April 1989.

[PER95] J. Padberg, H. Ehrig, L. Ribeiro. Algebraic High-Level Net Transformation Systems. *Mathematical Structures in Computer Science* 5(2):217–256, 1995.

[Pet81] J. L. Peterson. *Petri Net Theory and the Modeling of Systems*. Englewood Cliffs, NJ: Prentice-Hall, USA, 1981.

[Sch94] H. Schneider. Graph Grammars as a Tool to Define the Behavior of Processes Systems: From Petri Nets to Linda. In *Fifth International Conference on Graph Grammars and their Application to Computer Science*. Pp. 7–12. Williamsburg, USA, 1994.

[Val78] R. Valk. Self-Modifying Nets, a Natural Extension of Petri Nets. In Ausiello and Bhm (eds.), *Proc. of the 5th Int. Coll. Automata, Languages and Programming (ICALP'78)*. Lecture Notes In Computer Science 62, pp. 464–476. Springer-Verlag, Udine, Italy, 1978.

[Val81] R. Valk. Generalizations of Petri Nets. In Gruska and Chytil (eds.), *Proc. of 10th. Symposium on Mathematical Foundations of Computer Science (MFCS'81)*. Lecture Notes in Computer Science 118, pp. 140–155. Springer-Verlag, Strbske Pleso, Czechoslovakia, 1981.

[Weba] Sun's Home for Java: http://java.sun.com/.

[Webb] The Home Page of Petri Net Tools on the Web: http://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/.

[Webc] The Home Page of PIPE: http://pipe2.sourceforge.net/.

[Webd] The Home Page of PNML: http://www.informatik.hu-berlin.de/top/pnml/.