Introducing Structural Dynamic Changes in Petri Nets: Marked-Controlled Reconfigurable Nets*

Marisa Llorens and Javier Oliver

Departamento de Sistemas Informáticos y Computación,
Universidad Politénica de Valencia,
Camino de Vera s/n, E-46022 Valencia, Spain
{mllorens,fjoliver}@dsic.upv.es

Abstract. The aim of this work is the modeling and verification of concurrent systems that are subject to dynamic changes by using extensions of Petri nets. In previous studies, we have introduced net rewriting systems and a subclass of these called reconfigurable nets. In a 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 reconfigurable net is a 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 depends only on the net topology. Here we introduce marked-controlled net rewriting systems and marked-controlled reconfigurable nets where the enabling of a rule also depends on the net marking. We show an implementation of marked-controlled reconfigurable nets with Petri nets. Even though the expressiveness of both models is the same, with marked-controlled reconfigurable nets, we can easily and directly model systems that change their structure dynamically. It may be more efficient to directly implement the methods of verification of properties of Petri nets on the marked-controlled reconfigurable nets model.

1 Introduction

A Petri net [15, 16] is a formalism used to model, analyze, simulate, control and evaluate the behavior of distributed and concurrent systems. This formalism, however, does not offer a direct way to address modeling issues such as dynamic changes, multiple operating modes of operations, etc. Extensions of Petri nets have been designed to allow for an easy formalization of such features. The benefit in terms of modeling power is usually at the expense of a loss in decidable properties [7]. A trade-off needs to be found between expressiveness and computability. The fundamental goal of this work is the modeling, simulation and

* This work has been partially supported by CICYT TIC 2001-2705-C03-01, by Acción Integrada Hispano-Alemana HA2001-0059 and by Proyecto de Investigación UPV 7176.

verification of concurrent systems that are subject to dynamic changes. For example, to model a system of printers in which we want to choose dynamically between the printing of several copies of the same job on different printers or sequentially printing the copies on the same printer, the solution using *marked-controlled net rewriting systems* is very simple (see Fig. 1 and Fig. 2). It is important that the mechanism for handling dynamic changes in such systems be explicitly represented inside the model so that at each stage of product development, designers can experiment with the effects of structural changes (e.g., by using prototypes). This means that structural changes are taken into account from the very beginning of the design process rather than handled by an external, global system (e.g., by some exception handling mechanism), designed and added to the model describing the system's normal behavior. Thus, we are in favor of an internal and incremental description of changes over an external and uniform one, and a local handling of changes over a global one.

The model of *net rewriting systems* introduced in [3, 11–13] arises from two different lines of research. Both were conducted in the field of the Petri net formalism with the goal of enhancing the expressiveness of the basic model of Petri nets so that it can support the description of concurrent systems that are subject to dynamic changes. The first class of models covers various proposals for merging Petri nets with *graph grammars* [4, 6, 17] while the second class, which is best represented by *Valk's self-modifying nets* [18, 19], considers Petri nets whose flow relations can vary at runtime. Both proposals lead to expressive models that have definite benefits with respect to modeling issues. However, most of the basic decidable properties of Petri nets (place boundedness, reachability, deadlock and liveness) are lost for these extended models. Therefore, no automatic verification tools can be implemented for these models. *Reconfigurable nets* which were introduced in [3, 11–13] as a particular subclass of net rewriting systems, attempt to combine the most relevant aspects of both of these approaches and constitute a class of models for which each of the preceding fundamental properties are decidable. The translation of this model into Petri nets is automatic [11–13]. This equivalence ensures that all the fundamental properties of Petri nets are still decidable for reconfigurable nets, and therefore, this model is amenable to automatic verification tools. In contrast, the class of net rewriting systems is Turing powerful [11–13], and thus automatic verification is no longer possible for this larger class.

In *net rewriting systems*, 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 *reconfigurable net* is a net rewriting system 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. In other words, for a rewriting rule to be enabled, so that a change in configuration can take place, the net topology is the only thing to be taken into account. However, most of the changes in configuration that occur in real systems depend on the state of the system

(represented in a net by its marking). It would therefore be interesting for the enabling of a rewriting rule not only to depend on the net topology but to also depend on the net marking. Basically, the idea is to add a control mechanism to net rewriting systems and to reconfigurable nets so that the net only makes changes in configuration when a certain minimal marking is reached.

Section 2 introduces marked-controlled net rewriting systems. In Section 3, we present the definition of marked-controlled reconfigurable nets and a detailed example. An implementation of marked-controlled reconfigurable nets with Petri nets is shown in Section 4. Finally, we describe some related works and we present our conclusions in Section 5.

2 Marked-Controlled Net Rewriting Systems

This section introduces the model of *marked-controlled net rewriting systems*. First, we establish some notations.

If $R \subseteq X \times Y$ is a binary relation, we let $X'R = \{y \in Y \mid \exists x \in X' \ (x,y) \in R\}$ denote the *image* of $X' \subseteq X$, and $RY' = \{x \in X \mid \exists y \in Y' \ (x,y) \in R\}$ denote the *inverse image* of $Y' \subseteq Y$. The *domain* of R is then $Dom(R) = RY$ and the *codomain* of R is $Cod(R) = XR$.

A *Petri net* [15, 16] is a tuple $\Gamma = (P, T, F)$ where: $P = \{p_1, p_2, \ldots, p_m\}$ is a finite set of places, $T = \{t_1, t_2, \ldots, t_n\}$ is a finite set of transitions ($P \cap T = \emptyset$, $P \cup T \neq \emptyset$) and $F : (P \times T) \cup (T \times P) \to \{0, 1, 2, 3, \ldots\}$ is a set of weighted arcs (flow relation). A *marked Petri net* is a pair (Γ, M_0) where Γ is a Petri net and $M_0 : P \to \{0, 1, 2, 3, \ldots\}$ is the initial marking.

Let $\Gamma = (P, T, F)$ and $\Gamma' = (P', T', F')$ be two Petri nets. We call Γ and Γ' *isomorphic* if there exists a bijection $\varphi : (P \cup T) \to (P' \cup T')$ such that $F(x,y) = F'(\varphi(x), \varphi(y))$ for all $x, y \in P \cup T$.

A *full embedding* of a Petri net $\Gamma = (P, T, F)$ into a Petri net $\Gamma' = (P', T', F')$ is an *injective map* $f : P \cup T \to P' \cup T'$ that maps places to places and transitions to transitions ($f(P) \subseteq P'$ and $f(T) \subseteq T'$) such that for any pair of elements $x, y \in P \cup T$, $F(x,y) = F'(f(x), f(y))$. The image of Γ by f is then called a *full subnet* of Γ'.

Definition 1. *A* marked-controlled net rewriting system *is a structure* $N = (\mathcal{R}, (\Gamma_0, M_0))$ *where* $\mathcal{R} = \{r_1, \ldots, r_h\}$ *is a finite set of rewriting rules and* (Γ_0, M_0) *is a marked Petri net.*

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

1. $L = (P_L, T_L, F_L)$ *and* $R = (P_R, T_R, F_R)$ *are Petri nets called the* left-hand side *and the* right-hand side *of* r*, respectively;*
2. $\tau \subseteq (P_L \times P_R) \cup (T_L \times T_R)$*, called the* transfer relation *of* r*, is a binary relation relating places of* L *to places of* R *and transitions of* L *to transitions of* R*:* $P_L\tau \subseteq P_R$, $\tau P_R \subseteq P_L$, $T_L\tau \subseteq T_R$ *and* $\tau T_R \subseteq T_L$*;*
3. ${}^\bullet\tau \subseteq \tau$*, and* $\tau^\bullet \subseteq \tau$ *are sub-relations of the transfer relation called the* input interface *relation and the* output interface *relation, respectively;*

4. *C is a finite set of places, called* control places*, which is a subset of places of L, $C \subseteq P_L$;*
5. *$\mathbb{M}$ is the minimum marking of places of C required so that the rule is enabled.*

A configuration *of a marked-controlled net rewriting system N is a Petri net $\Gamma = (P, T, F)$.*

A state *of a marked-controlled net rewriting system N is a marked Petri net (Γ, M). The pair (Γ_0, M_0) is called the* initial state *of the marked-controlled net rewriting system.*

An event *of a marked-controlled net rewriting system is either a transition or a rewriting rule: $E = T \cup R$.*

Definition 2. *A* net rewriting system *[3, 11–13] is a marked-controlled net rewriting system where the set of control places C is empty ($C = \emptyset$) for all rewriting rules, that is, there is no restriction on the net marking.*

In order to apply a rewriting rule r to a configuration Γ, one must first identify a full subnet Γ' of Γ that is isomorphic to the left-hand side of the rule; that is, there exists a bijection $\varphi : (P_L \cup T_L) \to (P' \cup T')$ such that $F_L(x, y) = F'(\varphi(x), \varphi(y))$ for all $x, y \in P_L \cup T_L$. The elements of Γ (places or transitions) that do not belong to Γ' constitute the *context* of the rule. It is also required that $\forall p \in C, M(\varphi(p)) \geq \mathbb{M}(p)$. In order for the rule to be enabled, an element x' of Γ' must also have an element x of its preset that belongs to the context only if x' belongs to the input interface of the rule. In addition, an element x' of Γ' must also have an element x of its postset that belongs to the context only if x' belongs to the output interface of the rule. When these conditions are met, the rewriting can take place and it proceeds by replacing the subnet Γ' with the right-side R of the rule and by fixing the connections between the elements of R and those in the context according to the interface relation. The transfer relation is not only used to rewrite the left-side of the rule with the right-side but it is also used to transfer the tokens in Γ' to R. Notice that, since the transfer relation can be any type of relation, tokens may be duplicated or may disappear.

The dynamic evolution of a marked-controlled net rewriting system is given by its state graph.

Definition 3. *The* state graph of a marked-controlled net rewriting system $N = (\mathcal{R}, (\Gamma_0, M_0))$ *is the labeled directed graph whose nodes are the states of N (i.e., marked Petri nets) and whose arcs (labeled with events of N) are of two kinds described below:*

- firing of a transition: *arcs from state (Γ, M) to state (Γ', M') that are labeled with transition t when transition t can fire in the net Γ at marking M and leads to marking M':* $(\Gamma, M) \xrightarrow{t} (\Gamma', M') \iff (\Gamma = \Gamma' \quad and \quad M[t\rangle M' \text{ in } \Gamma)$.
- change in configuration: *arcs from state (Γ, M) to state (Γ', M') that are labeled with rule $r = (L, R, \tau, {}^\bullet\tau, \tau^\bullet, C, \mathbb{M}) \in \mathcal{R}$, when there exists a full embedding $f : L \to \Gamma$ such that $\forall\ x \notin f(L)$ and $y \in P_L \cup T_L$:*

$$x \in {}^\bullet f(y) \Rightarrow y \in Dom({}^\bullet\tau) \text{ and } x \in f(y)^\bullet \Rightarrow y \in Dom(\tau^\bullet)$$

and $\forall\ p \in C,\ M(f(p)) \geq \mathbb{M}(p)$ *and the following holds where* $\Gamma = (P, T, F)$ *and* $\Gamma' = (P', T', F')$*:*

$$P' = P - f(P_L) + P_R \text{ such that } P_L\tau \subseteq P_R$$
$$T' = T - f(T_L) + T_R \text{ such that } T_L\tau \subseteq T_R$$

where the meaning of + (-) is adding (removing) places/transitions to (from) Γ*. The name of places* P_R *(transitions* T_R*) added to* Γ *must be new in order to avoid clashes.*

$$F'(x,y) = \begin{cases} F(x,y) & if\ x,y \notin P_R \cup T_R \\ F_R(x,y) & if\ x,y \in P_R \cup T_R \\ \sum_{y_i \in {}^\bullet\tau y} F(x, f(y_i)) & if\ x \notin P_R \cup T_R\ \wedge\ y \in P_R \cup T_R \\ \sum_{x_i \in \tau^\bullet x} F(f(x_i), y) & if\ x \in P_R \cup T_R\ \wedge\ y \notin P_R \cup T_R \end{cases}$$

$$M'(p) = \begin{cases} M(p) & if\ p \notin P_R \\ \sum_{p' \in {}^\tau p} M(f(p')) & if\ p \in P_R \end{cases}$$

The following example illustrates the definition of a marked-controlled net rewriting system and the meaning of a change in configuration.

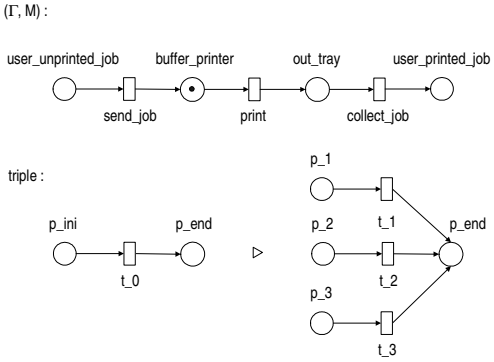

Fig. 1. Controlled Net Rewriting System modeling a system of printers

Example 1. The marked-controlled net rewriting system in Fig. 1 models the printing of several copies of the same job using different printers. The token in the place *buffer_printer* represents one copy of the job to print. In this state (Γ, M), to obtain three copies, the job must be sent to print three times and these three copies can only be printed sequentially (one after the other). The rewriting rule *triple* offers the possibility of printing each one of them using

a different printer (assuming there is some job to print). The subset of control places is $C = \{p\_ini\}$ and $\mathbb{M}(p\_ini) = 1$. The transfer relation τ is given by $\tau = \{(\{p\_ini\}, \{p\_1, p\_2, p\_3\}), (\{t\_0\}, \{t\_1, t\_2, t\_3\}), (\{p\_end\}, \{p\_end\})\}$ and the input and output interface relations are ${}^\bullet\tau = \{(\{p\_ini\}, \{p\_1, p\_2, p\_3\})\}$ and $\tau^\bullet = \{(\{p\_end\}, \{p\_end\})\}$, respectively. Figure 2 shows the new state (Γ', M') due to the change in configuration caused by the rewriting rule.

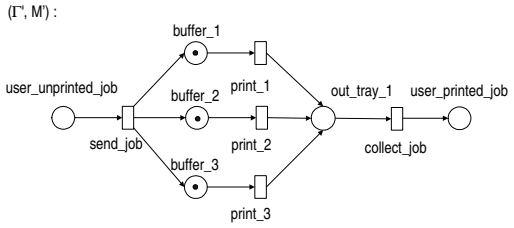

Fig. 2. Change in configuration due to the rewriting rule *triple*

Proposition 1. *Marked-controlled net rewriting systems are Turing powerful.*

Proof. It is straightforward from [11–13].

3 Marked-Controlled Reconfigurable Nets

Reconfigurable nets are a subclass of net rewriting systems where the transfer relation τ is a bijection [3, 11–13]. In other words, the set of places and transitions is left unchanged by rewriting rules in reconfigurable nets. Such rules are enabled if and only if the net has a certain topology. Moreover, such rules only change the flow relations of the places in their domains. We introduce marked-controlled reconfigurable nets as a subclass of marked-controlled net rewriting systems where the transfer relation τ is a bijection. We introduce marked-controlled reconfigurable nets as an extension of reconfigurable nets where the enabling of a rewriting rule also depends on the net marking.

Definition 4. *A* marked-controlled reconfigurable net *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 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 the initial state.*

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 marked-controlled reconfigurable net is a Petri net* $\Gamma = (P, T, F)$.

A state γ *of a marked-controlled reconfigurable net is a marked Petri net* $\gamma = (\Gamma, M)$.

The events *of a marked-controlled reconfigurable net are its transitions together with its rewriting rules:* $E = T \cup \mathcal{R}$.

Definition 5. *A* reconfigurable net *[3, 11–13] is a marked-controlled reconfigurable net where the set of control places* C *is empty* ($C = \emptyset$) *for all rewriting rules; that is, there is no restriction on the net marking.*

We represent a rewriting rule using formal sums notation as

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

Definition 6. *The* configuration graph $G(N)$ *of a marked-controlled reconfigurable net* $N = (P, T, \mathcal{R}, \gamma_0)$ *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 the 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 marked-controlled reconfigurable net is then given by its state graph.

Definition 7. *The* state graph *of a marked-controlled reconfigurable net* $N = (P, T, \mathcal{R}, \gamma_0)$ *is the labeled directed graph whose nodes are states of* N *and whose arcs (labeled with events) are of two kinds:*

- firing of a transition: *arcs from state* (Γ, M) *to* (Γ, M') *that are labeled with transition* t *when transition* t *can fire in the net* Γ *at marking* M *and leads to marking* M',
- change in configuration: *arcs from state* (Γ, M) *to state* (Γ', M) *that are 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*

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

The following example shows a system that is modeled by a marked-controlled reconfigurable net and a change in configuration that depends on the net topology and that also depends on the net marking.

Example 2 (Transmission Net). Figure 3 is the initial state $\gamma_0 = (\Gamma_0, M_0)$ of a marked-controlled reconfigurable net that represents a transmission net that is receiving blocks of information from two machines (represented by transitions t_1 and t_8) to be carried to a main building (represented by place p_7). The initial marking is $M_0 = (1, 2, 0, 1, 0, 0, 1)$.

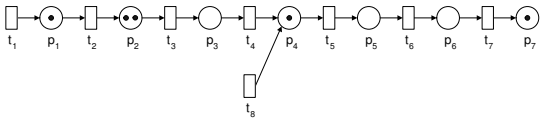

Fig. 3. The initial state of transmission net

This net has two different parts that are delimited by places p_1, p_3 and p_7. In each part, some changes in configuration can take place and they are represented by rewriting rules. In the first part, the packages can be sent in ones, in twos, in threes or in fours, depending on the weight of the arc from place p_2 to transition t_3. In the second part, the net offers three possibilities for the forwarding of data in the net: packages follow the normal path $t_4p_4t_5p_5t_6p_6t_7p_7$; packages avoid transition t_5 and follow the path $t_4p_4t_6p_6t_7p_7$; and finally, packages can go directly through the path $t_4p_4t_7p_7$, avoiding transitions t_5 and t_6. We define then eight rewriting rules using the formal sums notation previously introduced. We show the last one (R_8) graphically in Fig. 4.

$$\mathsf{R}_1\colon \mathsf{p}_2(\mathsf{t}_2\text{-}\mathsf{t}_3) \rhd \mathsf{p}_2(\mathsf{t}_2\text{-}2\mathsf{t}_3) \qquad \mathsf{R}_2\colon \mathsf{p}_2(\mathsf{t}_2\text{-}\mathsf{t}_3) \rhd \mathsf{p}_2(\mathsf{t}_2\text{-}3\mathsf{t}_3)$$

$$\mathsf{R}_3\colon \mathsf{p}_2(\mathsf{t}_2\text{-}\mathsf{t}_3) \rhd \mathsf{p}_2(\mathsf{t}_2\text{-}4\mathsf{t}_3) \qquad \mathsf{R}_4\colon \mathsf{p}_2(\mathsf{t}_2\text{-}2\mathsf{t}_3) \rhd \mathsf{p}_2(\mathsf{t}_2\text{-}3\mathsf{t}_3)$$

$$\mathsf{R}_5\colon \mathsf{p}_2(\mathsf{t}_2\text{-}2\mathsf{t}_3) \rhd \mathsf{p}_2(\mathsf{t}_2\text{-}4\mathsf{t}_3) \qquad \mathsf{R}_6\colon \mathsf{p}_2(\mathsf{t}_2\text{-}3\mathsf{t}_3) \rhd \mathsf{p}_2(\mathsf{t}_2\text{-}4\mathsf{t}_3)$$

$$\mathsf{R}_7\colon \mathsf{p}_4(\mathsf{t}_4{+}\mathsf{t}_8\text{-}\mathsf{t}_5){+}\mathsf{p}_5(\mathsf{t}_5\text{-}\mathsf{t}_6) \rhd \mathsf{p}_4(\mathsf{t}_4{+}\mathsf{t}_8\text{-}\mathsf{t}_6){+}\mathsf{p}_5(\emptyset)$$

$$\mathsf{R}_8\colon \mathsf{p}_4(\mathsf{t}_4{+}\mathsf{t}_8\text{-}\mathsf{t}_5){+}\mathsf{p}_5(\mathsf{t}_5\text{-}\mathsf{t}_6){+}\mathsf{p}_6(\mathsf{t}_6\text{-}\mathsf{t}_7) \rhd \mathsf{p}_4(\mathsf{t}_4{+}\mathsf{t}_8\text{-}\mathsf{t}_7){+}\mathsf{p}_5(\emptyset){+}\mathsf{p}_6(\emptyset)$$

The subset of control places C is empty for all rewriting rules except for rules R_7 and R_8 where $C = \{p_4\}$ and $\mathbb{M}(p_4) = 1$ (i.e., to change the path in the second part of the net, at least one package must be present in place p_4).

Therefore, the marked-controlled reconfigurable net consists of 7 places, 8 transitions and 8 rewriting rules. Figure 5 shows the state reached when rule R_8 is applied to the initial state represented in Fig. 3. In this new state, packages are sent in ones and follow the path $t_4p_4t_7p_7$, avoiding transitions t_5 and t_6.

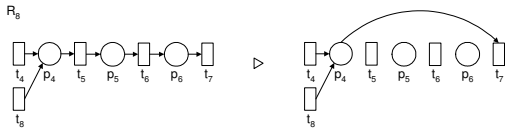

Fig. 4. Rewriting rule R_8

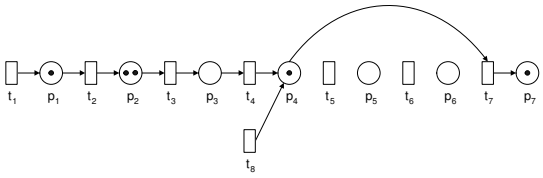

Fig. 5. New state reached after applying rule R_8 to the state in Figure 3

4 Implementation of Marked-Controlled Reconfigurable Nets with Petri Nets

We want to prove that marked-controlled reconfigurable nets are equivalent to Petri nets. This equivalence will ensure that all fundamental properties of Petri nets are still decidable for marked-controlled reconfigurable nets. This model is thus amenable to automatic verification tools. The translation of a marked-controlled reconfigurable net into an equivalent Petri net will considerably increase the net size and so it will be more efficient to directly implement the methods of verification of properties of Petri nets on the original model.

At first glance, it might seem they are not equivalent because of the set of rewriting rules of controlled reconfigurable nets. When a rewriting rule is applied in a marked-controlled reconfigurable net, a change in configuration takes place (let $\Gamma_i[r\rangle\Gamma_j$ denote the change in configuration due to rewriting rule r from Γ_i to Γ_j), i.e., a change in net structure. To obtain an equivalent Petri net, these changes in configuration must be present (i.e., all possible configurations must be represented in the net). We can therefore deduce that the number of configurations must be finite to be represented.

Let $Conf(N) = \{\Gamma_0, \Gamma_1, \ldots, \Gamma_k\}$ denote the set of configurations of a marked-controlled reconfigurable net N, where Γ_0 is the configuration of the initial state $\gamma_0 = (\Gamma_0, M_0)$ (the initial configuration). We can then easily construct an equivalent Petri net $\widetilde{N} = (\widetilde{P}, \widetilde{T}, \widetilde{F}, \widetilde{M_0})$ whose set of places is $\widetilde{P} = P \cup \{q_0, \ldots, q_k\}$, i.e., places of the original marked-controlled reconfigurable net together with one specific place attached to each possible configuration. We take as many copies of the set of transitions as the number of configurations, let $\{q_0, \ldots, q_k\} \times T$. We can imagine places and transitions of a configuration $\Gamma_i = (P, T, F_i)$ 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 flow relations of the represented configuration). Then we set up flow relations so that the configuration

Γ_i is represented by the plane of transitions $\{q_i\} \times T$ and the (shared) plane of places P:

$$\widetilde{F}(p,(q_i,t)) = F_i(p,t) \text{ where } \Gamma_i = (P,T,F_i)$$
$$\widetilde{F}((q_i,t),p) = F_i(t,p) \text{ where } \Gamma_i = (P,T,F_i)$$

Place q_i, which is associated to configuration Γ_i, contains at most one token, and it is marked in the states that are associated to this configuration. Thus, we set

$$\widetilde{F}(q_i,(q_j,t)) = \begin{cases} 1 & \text{if } i=j \\ 0 & \text{otherwise} \end{cases} \qquad \widetilde{F}((q_j,t),q_i) = \begin{cases} 1 & \text{if } i=j \\ 0 & \text{otherwise} \end{cases}$$

Then it is only necessary to represent the change in configurations. For that purpose, it suffices to add one extra transition r_{ij} for each change in configuration $\Gamma_i[r\rangle\Gamma_j$ that has a flow relation given by $\widetilde{F}(q_i,r_{ij}) = \widetilde{F}(r_{ij},q_j) = 1$, $\widetilde{F}(p,r_{ij}) = \widetilde{F}(r_{ij},p) = \mathbb{M}(p)$ if $p \in C$ and $\widetilde{F}(p,r_{ij}) = \widetilde{F}(r_{ij},p) = 0$, otherwise. So, the set of transitions is $\widetilde{T} = (\{q_0,\ldots,q_k\} \times T) \cup \widetilde{R}$ where $\widetilde{R}$ is the set of transitions such that $r_{ij} \in \widetilde{R}$ if $\exists r \in \mathcal{R}$ such that $\Gamma_i[r\rangle\Gamma_j$ in $G(N)$. The resulting Petri net initially is marked as: $\widetilde{M}_0(p) = M_0(p)$ where $p \in P$ and $\widetilde{M}_0(q_i) = \begin{cases} 1 & \text{if } i=0 \\ 0 & \text{otherwise} \end{cases}$.

To show the correspondence between marked-controlled reconfigurable nets and Petri nets some considerations might be taken into account:

- If $\widetilde{M} : \widetilde{P} \to \mathbb{N}$ is a reachable marking of $\widetilde{N}$ then $\sum_{i=0}^{k} \widetilde{M}(q_i) = 1$. So, from the subset of transitions $\{q_0,\ldots,q_k\} \times T \in \widetilde{T}$ only the transitions in $\{q_i\} \times T$ were enabled.
- We denote $\gamma_{\widetilde{M}} = (\Gamma_{\widetilde{M}}, M)$ where $M : P \to \mathbb{N}$ is a marking of $\Gamma_{\widetilde{M}}$ and $\Gamma_{\widetilde{M}} = \Gamma_i$ is the configuration associated to $\widetilde{M}$ such that $\widetilde{M}(q_i) = 1$. Inversely, if $\gamma = (\Gamma, M)$ is a reachable state of N, we associate the mapping $\widetilde{M}_\gamma : \widetilde{P} \to \mathbb{N}$ with $\widetilde{M}_\gamma(p) = M(p)$ if $p \in P$, $\widetilde{M}_\gamma(q_i) = 1$ if $\Gamma = \Gamma_i$ and $\widetilde{M}_\gamma(q_i) = 0$ otherwise.
- $\widetilde{M}_{(\gamma_{\widetilde{M}})} = \widetilde{M}$, $\gamma_{(\widetilde{M}_\gamma)} = \gamma$, $\gamma_{(\widetilde{M}_0)} = (\Gamma_0, M_0)$ and $\widetilde{M}_{\gamma_0} = \widetilde{M}_0$.

Proposition 2. *If $\gamma = (\Gamma, M)$ is a reachable state of N then*

1. $\gamma[r\rangle\gamma' \iff \widetilde{M}_\gamma[r_{ij}\rangle\widetilde{M}_{\gamma'}$ *where* $\gamma = (\Gamma_i, M)$ *and* $\gamma' = (\Gamma_j, M)$.
2. $\gamma[t\rangle\gamma' \iff \widetilde{M}_\gamma[(q_i,t)\rangle\widetilde{M}_{\gamma'}$ *where* $\gamma = (\Gamma_i, M)$ *and* $\gamma' = (\Gamma_i, M')$.

The previous proposition shows that:

1. If we change from state $\gamma = (\Gamma_i, M)$ to state $\gamma' = (\Gamma_j, M)$ in a marked-controlled reconfigurable net due to the firing of a rewriting rule, what differs in the equivalent Petri net is the marking of places q_i and q_j: $\widetilde{M}_\gamma(q_i) = 1 \Longrightarrow \widetilde{M}_{\gamma'}(q_i) = 0$ and $\widetilde{M}_\gamma(q_j) = 0 \Longrightarrow \widetilde{M}_{\gamma'}(q_j) = 1$. Also in the other direction.

2. If we change from state $\gamma = (\Gamma_i, M)$ to state $\gamma' = (\Gamma_i, M')$ in a marked-controlled reconfigurable net due to the firing of a transition t, what changes in an equivalent Petri net is the marking of places $p \in P$ involved by the firing of transition (q_i, t), and vice versa.

Hence we have established that:

Proposition 3. *The markings $\widetilde{M}_\gamma$ in which γ covers the reachable states of N are the reachable markings of $\widetilde{N}$, and the marking graph of $\widetilde{N}$ is isomorphic to the marking graph of N. In this sense, any marked-controlled reconfigurable net is equivalent to some Petri net.*

Proof. It is straightforward.

Thus, marked-controlled reconfigurable nets are equivalent to Petri nets, but they provide somewhat more compact representations of concurrent systems whose structure evolves at runtime. This can be observed in Fig. 6, which illustrates a fragment of the Petri net that is equivalent to the marked-controlled reconfigurable net in Example 2. To facilitate the understanding of the implementation of a marked-controlled reconfigurable net with a Petri net, we only show how to represent one change in configuration (the change of path in the second part of the net that avoids transitions t_5 and t_6 -rewriting rule R_8-) and only for the places and transitions involved. With this short fragment, we can imagine how big the entire Petri net is and how we can best model the system with a marked-controlled reconfigurable net. In general, if the original marked-controlled reconfigurable net $N = (P, T, \mathcal{R}, \gamma_0)$ has:

- n places,
- m transitions,
- r rewriting rules and
- the number of accesible configurations is $k + 1$,

the obtained equivalent Petri net $\widetilde{N} = (\widetilde{P}, \widetilde{T}, \widetilde{F}, \widetilde{M}_0)$ consists of:

- $n + (k + 1)$ places and
- $((k + 1) * m) + z$ transitions, where $z = \sum_{i=0}^{k} (|\Gamma_i^\bullet| + |^\bullet\Gamma_i|)$ and $|\Gamma_i^\bullet|$ $(|^\bullet\Gamma_i|)$ is the number of output (input) arcs from (to) the configuration Γ_i in the configuration graph of N, $G(N)$. That is, z is the number of transitions of $\widetilde{R}$.

This increase on the net size presumes that it may be more efficient to directly implement the methods of verification of properties of Petri nets on the marked-controlled reconfigurable net than on the equivalent Petri net. In any case, it is possible to study the properties of the net (place boundedness, reachability, deadlock, liveness and structural properties) using the existing analysis tools for Petri nets on the equivalent Petri net.

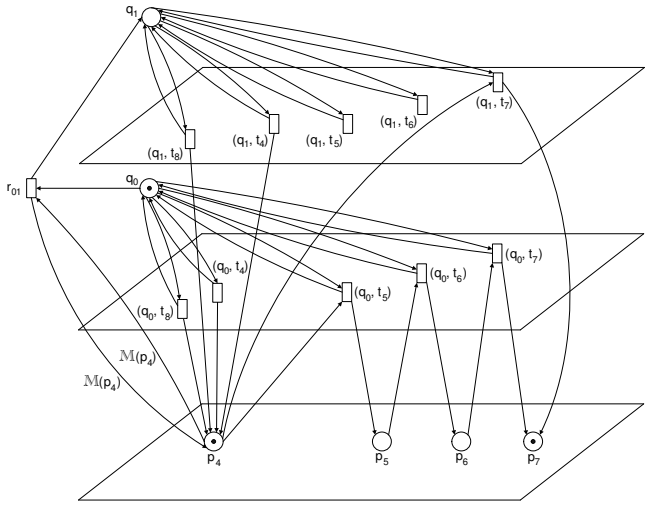

Fig. 6. Part of the Petri net that is equivalent to the Marked-controlled reconfigurable net in Example 2

5 Related Work and Conclusions

In previous studies [3, 11–13], we have introduced net rewriting systems, which are an extension of Petri nets that are suited for the modeling, simulation and verification of concurrent systems that are subject to dynamic changes. Net rewriting systems can dynamically modify their own structure by rewriting some of their components. This rewriting depends only on the net topology. Here we have introduced marked-controlled net rewriting systems, an extension of net rewriting systems where the rewriting also depends on the net marking. Both models are based on two different lines of research that extend the basic model of Petri nets, making the description of dynamic changes in concurrent systems possible: *graph grammars* [6, 17, 4] and *Valk's self-modifying nets* [18, 19]. The rewriting rules of marked-controlled net rewriting systems are very similar to productions of graph grammars (they have left- and right-hand sides and interfaces), and the application of a rewriting rule is like a direct derivation in graph grammars (under certain conditions, once an occurrence (a *match*) of the left-hand side in a graph has been detected, it can be replaced by the right-hand side). As in self-modifying nets, we model a system which consists of a collection of Petri nets, called *configurations*, and a mechanism that allows the system to evolve from one configuration to another under certain circumstances.

Besides Valk's self-modifying nets, in the literature, there are some other models that allow for the description of complex, dynamic concurrent systems. The *mobile nets* of Asperti and Busi [1] originated from a merging of Petri nets with the name-managing techniques typical in π-calculus [14]; the *dynamic nets* of Buscemi and Sassone [5] inspired by the join calculus [9]. Both of these allow

the dynamic creation of components, as in our proposal. Other models are the Δ-*nets* of Gradit and Vernadat [10], a rewriting formalism which integrates the advantages of Petri nets and graph grammars, respectively, for behavior specification and topological transformations of a workflow, and the *POP formalism* introduced by Engelfriet, Leih and Rozenberg [8], and the related model of *cooperating automata* by Badouel, Darondeau and Tokmakoff [2], in which tokens are active elements with dynamic behavior. For all of them, as in our model, the description of changes is internal and incremental and their handling is local. Also, the idea of rewriting underlies all these proposals; the configuration of the system is described as a Petri net and a change in configuration is described as a graph rewriting rule which replaces the part of the system that matches the left-hand side of the rewriting rule by the corresponding right-hand side. With respect to the expressive power, all of them are Turing-equivalent, as our model. The model of marked-controlled net rewriting systems is closer to Petri nets.

The model of marked-controlled reconfigurable nets is introduced here as a specific marked-controlled net rewriting system where the transfer relation τ is a bijection. This model (even if formally equivalent to Petri nets) allows us to more precisely express systems in which structural dynamic changes can occur. However, automatic translation into Petri nets ensures that all the fundamental properties of Petri nets are still decidable for marked-controlled reconfigurable nets. This model is thus amenable to automatic verification tools. However, the expansion into equivalent Petri nets may significantly increase the size of the net. Therefore, it may be more efficient to directly implement the methods of verification of properties of Petri nets on the original model. This presumes that we can define the notions of covering graphs, linear invariants, siphons and traps directly for a marked-controlled reconfigurable net. These topics are the subject of our current research. In contrast, the entire class of marked-controlled net rewriting systems is Turing powerful, and thus automatic verification is no longer possible in this case. However, this model is still interesting as a modeling and simulation tool. For some of the models described above, software tools have been developed for editing and simulating systems using a graphical user interface. These tools provide software support for the design of prototypes for dynamic concurrent systems. We are currently implementing a simulator for marked-controlled net rewriting systems.

References

1. A. Asperti and N. Busi. Mobile Petri Nets. Technical report UBLCS-96-10, University of Bologna, Italy, 1996.
2. E. Badouel, Ph. Darondeau, and A. Tokmakoff. Modelling Dynamic Agents Systems with Cooperating Automata. In *Proc. Int. Conf. on Parallel and Distributed Processing Techniques and Applications* (PDPTA'99), pp. 11–17, USA, 1999.
3. E. Badouel, M. Llorens, and J. Oliver. Modelling Concurrent Systems: Reconfigurable Nets. In *Proc. Int. Conf. on Parallel and Distributed Processing Techniques and Applications* (PDPTA'03), vol. IV, pp. 1568–1574, USA, 2003.

4. P. Baldan. Modelling Concurrent Computations: From Contextual Petri Nets to Graph Grammars. PhD Thesis, University of Pisa TD-1/00, 2000.
5. M. Buscemi and V. Sassone. High-level Petri nets as type theories in the Join calculus. In *Proc. 4th I.C. on Foundations of Software Science and Computation Structures* (FoSSaCS'01), Springer, *LNCS*, vol. 2030, pp. 104–120, Italy, 2001.
6. A. Corradini. Concurrent Computing: From Petri Nets to Graph Grammars. Invited talk at the *Joint COMPUGRAPH/SEMAGRAPH Workshop on Graph Rewriting and Computation*, Elsevier, *ENTCS*, vol. 2, 1995. http://www.elsevier.nl/locate/entcs/volume2.html.
7. C. Dufourd, A. Finkel, and Ph. Schnoebelen. Reset nets between decidability and undecidability. In *Proc. Int. Coll. on Automata, Languages and Programming* (ICALP'98), Springer-Verlag, *LNCS*, vol. 1443, pp. 103–115, Denmark, 1998.
8. J. Engelfriet, G. Leih, and G. Rozenberg. Net Based Description of Parallel Object-based Systems, or POTs and POPs. *Workshop on Foundations of Object-Oriented Languages* (FOOL'90), Springer-Verlag, *LNCS*, vol. 489, pp. 229–273, Noordwijkerhout, Netherlands, 1991.
9. C. Fournet, G. Gonthier, J. Lévy, L. Maranget, and D. Rémy. A Calculus of Mobile Agents. In *Proc. 7th Int. Conf. on Concurrency Theory* (CONCUR'96), Springer-Verlag, *LNCS*, vol. 1119, pp. 406–421, Pisa, Italy, 1996.
10. P. Gradit, F. Vernadat, and P. Azéma. Layered Δ-Net Specification of a Workshop. In *Proc. Int. Conf. on Parallel and Distributed Processing Techniques and Applications* (PDPTA'99), vol. VI, pp. 2808–2814, USA, 1999.
11. M. Llorens and J. Oliver. Sistemas de Reescritura de Redes. In *XI Jornadas de Concurrencia*, pp. 237–250, Benicassim, Castellón (Spain), 2003.
12. M. Llorens. Redes Reconfigurables. Modelización y Verificación. Phd thesis, Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, Spain, 2003.
13. M. Llorens and J. Oliver. Structural and Dynamic Changes in Concurrent Systems: Reconfigurable Nets. In *IEEE Transactions on Computers*, vol. 53, no. 9, pp. 1147–1158, September 2004.
14. R. Milner, J. Parrow, and D. Walker. A Calculus of Mobile Processes. In *Journal of Information and Computation*, Academic Press, vol. 100, no.1, pp. 1–77, 1992.
15. T. Murata. Petri Nets: Properties, Analysis and Applications. In *Proc. of the IEEE*, vol. 77, no.4, pp. 541–580, 1989.
16. J.L. Peterson. Petri Net Theory and the Modeling of Systems. Englewood Cliffs, NJ. Prentice-Hall, 1981.
17. H. Schneider. Graph Grammars as a Tool to Define the Behavior of Processes Systems: From Petri Nets to Linda. In *Proc. 5th Int. Conf. on Graph Grammars and their Application to Computer Science*, pp. 7–12, Williamsburg, USA, 1994.
18. R. Valk. Self-modifying Nets, a Natural Extension of Petri Nets. In *Proc. Int. Coll. on Automata, Languages and Programming* (ICALP'78), Springer-Verlag, *LNCS*, vol. 62, pp. 464–476, Udine, Italy, 1978.
19. R. Valk. Generalizations of Petri Nets. In *Proc. 10th Symp. on Mathematical Foundations of Computer Science* (MFCS'81), Springer-Verlag, *LNCS*, vol. 118, pp. 140–155, Strbske Pleso, Czechoslovakia, 1981.