Modeling Concurrent Systems: Reconfigurable Nets

Eric Badouel
Éc. Nat. Sup. Polytechnique
Yaoundé, Cameroun
ebadouel@polytech.uninet.cm

Marisa Llorens and Javier Oliver
DSIC, UPV, 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 subject to dynamic changes using extensions of Petri nets. We introduce the* reconfigurable nets*. The expressiveness of reconfigurable nets and Petri nets is equivalent, but with reconfigurable nets we can model easily and directly systems that change its structure dynamically. The change of configuration amounts to the modification of the flow relations of the places in the domain of the involved rule according to the rule and independently of the context in which this rewriting applies.*

Keywords: Concurrent systems, Petri nets, Graph rewriting, Dynamic changes.

1 Introduction

Reconfigurable nets are an extension of Petri nets [11][12]. Petri nets are a formalism to model, analyze, simulate, control and evaluate the behavior of distributed and concurrent systems. But this formalism does not offer a direct way to address some modeling issues like dynamic changes, multiple operating modes of operations, etc. Extensions of Petri nets have been designed to allow an easy formalization of such features. The gain in terms of modeling power is usually paid by a loss of decidable properties. A trade-off needs to be found between expressiveness and computability.

The fundamental aim of this work is the modeling and verification of concurrent systems subject to dynamic changes. It is important that the mechanism for handling dynamic changes in such systems be explicitly represented into the model so that at each stage of product development, designers can experiment the effect 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 and global system, e.g., by some exception handling mechanism, designed and added to the model describing the system normal behavior. Thus we favor an internal and incremental over an external and uniform description of changes, and a local over a global handling of changes.

Reconfigurable nets are a subclass of net rewriting systems that we have introduced in [3]. The proposed model arises from two different lines of research both conducted in the field of the Petri net formalism and aiming at enhancing the expressiveness of the basic model of Petri nets. The first class of models covers various proposals for merging Petri nets with graph grammars [7, 13, 6] while the second category, best represented by Valk's Self-Modifying Nets [14][15], considers Petri nets whose flow relations can vary at runtime. Both propositions lead to expressive models that have definite benefit 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 could be implemented in the context of these models. Reconfigurable nets try to combine the most relevant aspects

of both preceding approaches while constituting a class of models for which each of the preceding fundamental properties are decidable. This model should then be amenable to automatic verification.

In Section 2 we recall the definition of Net Rewriting System of [3]. This definition combines Petri nets with graph rewriting systems. In Section 3 we present our model of reconfigurable nets and we show an example of this model. Finally, we conclude in Section 4.

2 Net Rewriting Systems

This section recalls the model of *net rewriting system* of [3]. First, we fix 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 $Cod(R) = XR$.

Definition 1 *A* net rewriting system *[3] 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, $\Gamma_0 = (P_0, T_0, F_0)$ is a Petri net and $M_0 : P_0 \to \mathbb{N}$ is a marking associated with Γ_0. A* rewriting rule *$r \in \mathcal{R}$ is a structure $r = (L, R, \tau, {}^\bullet\tau, \tau^\bullet)$ 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 ($P_L\tau \subseteq P_R$, $\tau P_R \subseteq P_L$, $T_L\tau \subseteq T_R$, $\tau T_R \subseteq T_L$) and 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.*

A configuration *of a net rewriting system N is a Petri net $\Gamma = (P, T, F)$. A* state *of a net rewriting system N is a pair (Γ, M) consisting of a Petri net $\Gamma = (P, T, F)$ (configuration) together with a marking $M : P \to \mathbb{N}$ for Γ, hence it is a marked Petri net. The pair (Γ_0, M_0) is called the initial state of the net rewriting system. An* event *of a net rewriting system is either a transition or a rewriting rule: $E = T \cup R$.*

In order to apply a rewriting rule r to a configuration Γ one must first identify a full subnet Γ' of Γ isomorphic to the left-hand side of the rule. The elements of Γ (places or transitions) which do not belong to Γ' constitute the *context* of the rule. In order that the rule be enabled it is further required that an element x' of Γ' may have an element x of its preset that belongs to the context only if x' belongs to the input interface of the rule and, symmetrically, that an element x' of Γ' may 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' by 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 by the right-side one but also to transfer the tokens in Γ' to R (hence its name). Notice that, since the transfer relation can be any sort of relation, tokens may in this manner be duplicated or may disappear.

Definition 2 *The* state graph of a 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:*

- firing of a transition: *arcs from state (Γ, M) to state (Γ', M') labeled with transition t when transition t can fire in the net Γ at marking M and leads to marking M': $(\Gamma, M) \xrightarrow{t} (\Gamma', M') \iff (\Gamma = \Gamma'$ and $M[t\rangle M'$ in $\Gamma)$;*
- change of configuration: *arcs from state (Γ, M) to state (Γ', M') labeled with rule $r = (L, R, \tau, {}^\bullet\tau, \tau^\bullet) \in \mathcal{R}$, when there exists a* full embedding *(see [3]) $f : L \to \Gamma$ such that for all $x \notin f(L)$ and $y \in L$ one has $x \in {}^\bullet f(y) \Rightarrow y \in Dom({}^\bullet\tau)$*

and $x \in f(y)^\bullet \Rightarrow y \in Dom(\tau^\bullet)$ *and the following holds where* $\Gamma = (P, T, F)$ *and* $\Gamma' = (P', T', F')$*:*

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

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 to avoid clashes.*

$$M'(p) = \begin{cases} M(p) & \text{if } p \notin R \\ \sum_{p' \in\ \tau p} M(p') & \text{if } p \in R \end{cases}$$

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

Let us present a simple example of a net rewriting system.

Example 3 (Printers) *The net rewriting system of Fig. 1 models the printing of several copies of the same job by 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 only could be printed sequentially (one after the other). The rewriting rule triple offers the possibility of printing each of them by a different printer. 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 ${}^\bullet\tau$ *and* $\tau^\bullet$ *are, respectively:*

$${}^\bullet\tau = \{(p\_ini, \{p\_1, p\_2, p\_3\})\}$$

$$\tau^\bullet = \{(p\_end, p\_end)\}$$

Figure 2 shows the new state (Γ', M') *due to the change of configuration caused by rewriting rule triple.*

Proposition 4 *Net rewriting systems are Turing powerful.*

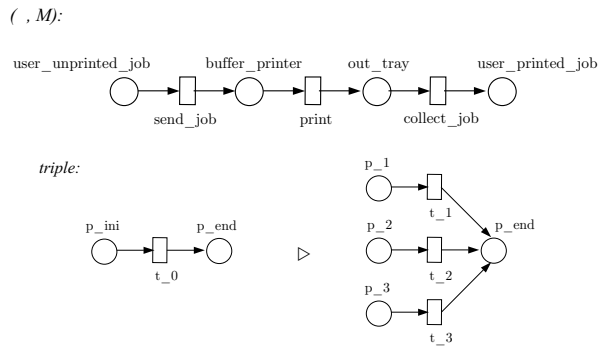

Figure 1: Net Rewriting System modeling a system of printers

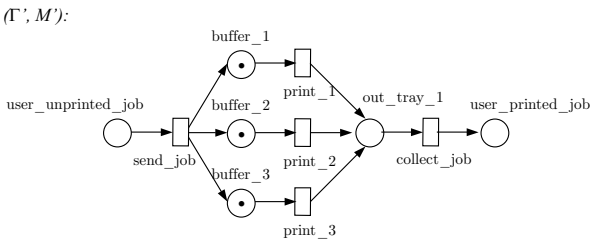

Figure 2: Change of configuration due to rewriting rule *triple*

3 Reconfigurable Nets

In [4] we have introduced a class of high level Petri nets, called *reconfigurable nets*, that can dynamically modify their own structure by rewriting some of their components so that it can support the description of dynamic changes in concurrent and distributed systems, in particular, they can be used to describe the workflow systems [8]. A reconfigurable net is a Petri net with local structural modifying rules performing the replacement of one of its subnets by another subnet. The tokens in a deleted place are transferred to a created one. These nets were used for modeling dynamic changes within workflow systems, as in [8]. It was shown that boundedness of a reconfigurable net can be decided by constructing a simplified form of Karp and Miller's coverability tree; however this construction did not allow to decide whether a given place of the net is bounded. Finally, a translation of a reconfigurable net into an equivalent self-modifying

net was also presented. This class of reconfigurable nets appears as a subclass of self-modifying nets for which boundedness can be decided. However this translation uses features like reset arcs whereas boundedness of the class of Petri nets with reset arcs is, in general, undecidable. This suggested that a simpler translation should be possible at least for particular subclasses of reconfigurable nets. In a subsequent study [5], we restricted our attention to the subclass of reconfigurable nets, termed *reversible*, whose structure modifying rules are bijections. This assumption was quite natural since the systems that we intend to model are supposed to run forever and that should not have degraded behaviors: reversibility means that in any accessible configuration it should be possible to reach the initial configuration, if the system is not so then it means that it can lost some of this abilities (this is the meaning of "degraded behavior") but we are interested only in the asymptotic behaviors of the system and each of these, which correspond to strongly connected components of the initial system, are reversible. We proved in [5] that Boundedness, Reachability and Liveness are decidable for reversible reconfigurable nets. This extra assumption of reversibility allows us to reformulate the definition of reconfigurable nets in a more direct manner as follows:

Definition 5 *A* reconfigurable net *(first attempt) is a net rewriting system such that the transfer relation τ is a bijection and the interfaces $\tau^\bullet = {}^\bullet\tau$ are the restriction of the transfer relation τ to the set of transitions, i.e., $\tau^\bullet = {}^\bullet\tau = T_L \times T_R \cap \tau$.*

The former condition asserts that the set of places and transitions is left unchanged by rewriting rules, the latter condition states that a change of configuration amounts to the modification the flow relations of the places in the domain of the involved rule according to this rule and independently of the context in which this rewriting applies. Thus the above definition may in turn be reformulated more directly (i.e. without resorting to net rewriting systems) using the following definition which we shall consider, from now on, as the official definition of reconfigurable nets (since reversibility will always be an implicit assumption).

Definition 6 *A* reconfigurable net *is a structure $N = (P, T, R, \gamma_0)$ 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 ($P \cap T = \emptyset$), $R = \{r_1, \ldots, r_h\}$ is a finite set of rewriting rules, and γ_0 is the initial state.*

A rewriting rule *$r \in R$ is a structure $r = (D, {}^\bullet r, r^\bullet)$ where $D \subseteq P$ is the domain of r, ${}^\bullet r : D \to (T \to \mathbb{N})^2$ and $r^\bullet : D \to (T \to \mathbb{N})^2$ are the preconditions and postconditions of r.*

A configuration *of a reconfigurable net is a Petri net $\Gamma = (P, T, F)$.*

A state *γ of a reconfigurable net is a pair $\gamma = (\Gamma, M)$ where Γ is a configuration and $M : P \to \mathbb{N}$. Hence a state is a marked Petri net associated with the set of places and transitions of the reconfigurable net.*

The events *of a reconfigurable net are its transitions together with its rewriting rules: $E = T \cup R$.*

We represent a rewriting rule using formal sums notations 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 7 *The* configuration graph *$G(N)$ of a reconfigurable net $N = (P, T, R, \gamma_0)$ is the labeled directed graph whose nodes are the configurations, and such that one has an arc from configuration Γ to configuration Γ' labeled with rule $r = (D, {}^\bullet r, r^\bullet) \in R$, which we denote $\Gamma[r\rangle\Gamma'$, if and only if the following hold:*

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

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

The behavior of a reconfigurable net is then given by its state graph.

Definition 8 *The* state graph *of a reconfigurable net $N = (P, T, R, \gamma_0)$ 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') labeled with transition t when transition t can fire in the net Γ at marking M and leads to marking M';*
- change of configuration: *arcs from state (Γ, M) to state (Γ', M) labeled with rule $r \in R$ if $\Gamma[r\rangle\Gamma'$ is a transition of the configuration graph of N, i.e. the set of transitions (labeled arcs) of the state graph of N is given by*

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

It is obvious from the previous definition that the evolution of a system modeled by a reconfigurable net depends on what event takes place. If a transition is fired, only the net marking is affected (as in Petri nets) whereas if a rewriting rule is applied, the net structure is modified and the marking continues being the original marking, i.e., changes of configurations are orthogonal to the behavior of the underlying net. On the other hand, in the general case of net rewriting systems, the application of a rewriting rule not only affects to net structure but also to net marking and therefore the behavior is distinct to the underlying net. Nevertheless, since there are only modeling and simulating tools for this case and it is no difficult to restrict the behavior so that changes of configurations take place when some system local properties occur, it is possible to take into account these limits at simulator level.

Each change of configuration that could take place in a reconfigurable net is represented by a rewriting rule. To define these rules in a generic way, we can group places and transitions according to a designer decision. This doesn't change the nature of the considered model but it facilitates its description. We define the notion of *role*.

Definition 9 *Two places (transitions) play the same* role *if the designer groups them as the same kind of place (transition).*

Example 10 (Producers/Consumers) *Figure 3 represents two half-duplex communicating lines between two producers/consumers.*

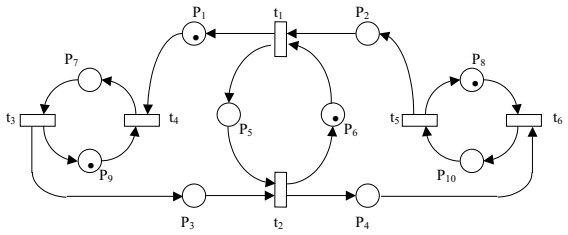

Figure 3: Two half-duplex communicating lines between two producers/consumers

In this reconfigurable net some changes of configuration could take place. Each of them is represented by a rewriting rule. In the example, we distinguish the role P, Q and S between places and the role A and B between transitions: $P = \{P_1, P_2, P_3, P_4\}$, $Q = \{P_5, P_6\}$, $S = \{P_7, P_8, P_9, P_{10}\}$, $A = \{t_1, t_2\}$ and $B = \{t_3, t_4, t_5, t_6\}$. We define four rewriting rules using the formal sums notation and we show in Fig. 4 the last one graphically.

1. *Change from sequential to parallel mode:*

$$R_1 : Q(A - A) + Q(-A + A) \rhd Q(\emptyset) + Q(\emptyset)$$

2. *Change of sense in one communicating line:*

$$R_2 : P(-A) + P(A) \rhd P(A) + P(-A)$$

3. *Change from producer/consumer to consumer:*

$$R_3 : S(B-B)+P(-B)+S(-B+B)+P(B) \rhd S(B - B) + P(B) + S(-B + B) + P(\emptyset)$$

4. *Change from producer/consumer to producer:*

$$R_4 : S(B-B)+P(-B)+S(-B+B)+P(B) \rhd S(B - B) + P(-B) + S(-B + B) + P(\emptyset)$$

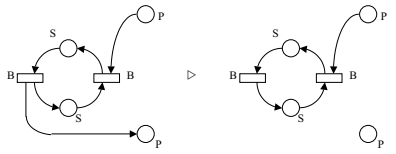

Figure 4: Rewriting rule to change from producer/consumer to producer

Figure 5 shows part of the state graph of the reconfigurable net example. We can appreciate that when we apply a rewriting rule we obtain a state in which only the flow relations are changed (the marking is the original marking); whereas if a transition is fired, the marking changes and the structure is not modified.

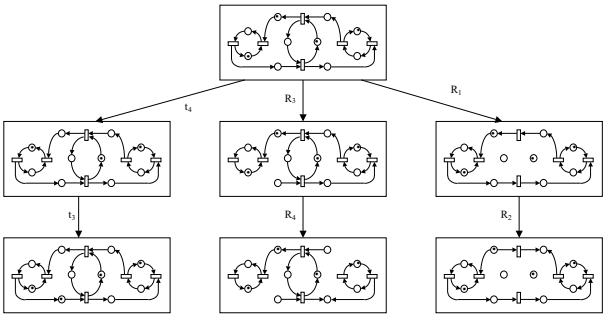

Figure 5: (Part of) the state graph of the reconfigurable net

Proposition 11 *Any reconfigurable net is equivalent to a Petri net.*

4 Related Work and Conclusions

We have introduced the model of reconfigurable nets as a particular net rewriting system where the transfer relation τ is a bijection. This model, even if formally equivalent to Petri nets, allows us to express in an easier and more concise way 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 reconfigurable nets. This model is thus amenable to automatic verification tools.

By contrast, the class of net rewriting systems whose transfer relations are partial bijections are Turing powerful, and thus automatic verification is not longer possible for this larger class. There are some other models in the literature that allows for the description of complex dynamic concurrent systems: *Valk's self-modifying nets* [14][15], the *mobile nets* of Asperti and Busi [1], the Δ-*nets* of Gradit and Vernadat [10][16], the *POP formalism* introduced by Engelfriet, Leih and Rozenberg [9] and the related model of *cooperating automata* by Badouel, Darondeau and Tokmakoff [2]. As net rewriting systems, the idea of rewriting is underlying in all proposals and all of them are Turing powerful. But the model of net rewriting systems is closer to Petri nets and also more intuitive than the others.

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. of Parallel and Distributed Processing Techniques and Applications* (PDPTA'99), CSREA Press, pp. 11–17, Las Vegas, USA, 1999.

[3] E. Badouel, M. Llorens, and J. Oliver. Modelling Concurrent Systems using Reconfigurable Nets. In *Actes du 6eme Seminaire Atelier en Algèbre, Logique, et ses Applications*, Yaoundé, Cameroon, 2002.

[4] E. Badouel and J. Oliver. Reconfigurable Nets, a Class of High Level Petri Nets Supporting Dynamic Changes Within Workflow Systems. In *Proc. of Workflow Management: Net-based concepts, models, techniques and tools* (WFM'98), CSR 98/07, Eindhoven University of Technology, Department of Mathematics and Computing Science, pp. 129–145, Lisbon, Portugal, 1998.

[5] E. Badouel and J. Oliver. Dynamic Changes in Concurrent Systems: Modelling and Verification. Technical report, Inria Research Report PI-3708, France, 1999.

[6] P. Baldan. Modelling Concurrent Computations: From Contextual Petri Nets to Graph Grammars. PhD Thesis, University of Pisa TD-1/00, 2000.

[7] A. Corradini. Concurrent Computing: From Petri Nets to Graph Grammars. Invited talk at the *Joint COMPUGRAPH/SEMAGRAPH Workshop on Graph Rewriting and Computation*, Elsevier, *Electronic Notes in Theoretical Computer Science*, vol. 2, 1995.

[8] C. Ellis, K. Keddara and G. Rozenberg. Dynamic Change within Workflow Systems. In *Proc. Conf. on Organizational Computing Systems* (COOCS'95), ACM Press, Business Processes and Workflows, pp. 10–21, New York, 1995.

[9] 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, *Lecture Notes in Computer Science*, vol. 489, pp. 229–273, Noordwijkerhout, Netherlands, 1991.

[10] P. Gradit, F. Vernadat, and P. Azéma. Layered Δ-Net Specification of a Workshop. In the *Int. Conf. on Parallel and Distributed Processing Techniques and Applications* (PDPTA'99), vol. VI, pp. 2808–2814, Las Vegas, USA, 1999.

[11] T. Murata. Petri Nets: Properties, Analysis and Applications. In *Proc. of the IEEE*, vol. 77, no.4, pp. 541–580, 1989.

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

[13] H. Schneider. Graph Grammars as a Tool to Define the Behavior of Processes Systems: From Petri Nets to Linda. In the *4th Int. Conf. on Graph Grammars*, Williamsburg, USA, 1993.

[14] R. Valk. Self-modifying Nets, a Natural Extension of Petri Nets. In *Proc. of Int. Coll. on Automata, Languages and Programming* (ICALP'78), Springer-Verlag, *Lecture Notes in Computer Science*, vol. 62, pp. 464–476, Udine, Italy, 1978.

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

[16] F. Vernadat, K. Drira, and P. Azéma. An Integrated Description Technique for Distributed Cooperative Applications. In *Proc. of CESA'96 IMACS Multiconference, Symposium on Discrete Events and Manufacturing Systems*, pp. 608–612, Lille, France, 1996.