Reconfigurable Nets, a Class of High Level Petri Nets Supporting Dynamic Changes*

Eric Badouel[§] and Javier Oliver[¶]

[§] IRISA, Campus de Beaulieu, 35042 Rennes Cedex, France.
E-mail: Eric.Badouel@irisa.fr.

[¶] DSIC, UPV, Camino de Vera s/n, 46071 Valencia, Spain.
E-mail: fjoliver@dsic.upv.es.

Abstract. We introduce a class of high level Petri nets, called reconfigurable nets, that can dynamically modify their own structure by rewriting some of their components. Boundedness of a reconfigurable net can be decided by constructing its coverability tree. Moreover such a net can be simulated by a self-modifying Petri net. The class of reconfigurable nets thus provide a subclass of self-modifying Petri nets for which boundedness can be decided. Delayed dynamic changes within workflow systems in the sense of [8] can then be handled in an extension of van der Aalst's workflow nets [2]. For this class (the *reconfigurable workflow nets*), a notion of soundness has been defined that can also be verified using the coverability tree construction.

Keywords: Reconfigurable Nets, Workflow Systems, Boundedness, Self-Modifying Petri Nets.

1 Introduction

Since their introduction in the early sixties [17], Petri nets have come to play a pre-eminent role in the formal study of the behaviour of concurrent and distributed systems. Let us mention some of the attractive features that have made this model successful. First of all, like vector addition systems or commutative semi Thue systems, Petri nets are a very simple and natural extension of automata. Therefore the study of their mathematical properties becomes a manageable task; in particular much effort have been devoted to decidability and complexity issues for Petri nets. Second, a lot of techniques and automated tools support the verification of properties of systems modelled by Petri nets. For instance one can decide by constructing its coverability tree whether a Petri net is bounded, i.e. whether it is a finite state system. Reduction techniques and linear algebra techniques have also received wide attention. Third, Petri net is a graphical tool that can easily be used for the description and the design of concurrent systems.

* This work was partially supported by the H.C.M. Network *Express* and by CICYT, TIC 95-0433-C03-03.

Recently, Petri nets have been used for modelling *Computer Supported Cooperative Work* (CSCW) [9, 5]. These applications, also called *groupware* applications, involve distributed systems of agents (computer systems or humans) which cooperate to solve a global task and whose structure can dynamically change. More precisely, we consider in this paper *workflow systems*. These systems aim to support the realization of work procedures by a group of collaborating agents by coordinating the flow of tasks within the distributed system. As in [2] we define a workflow net as a Petri net with a specific input place and a specific output place. A token in the input place corresponds to a new *case* entering the system, the structure of the Petri net describe the set of *tasks* required to process this case and the order in which these tasks can be executed (taking the distributed nature of the system into account). Finally a token in the output place witnesses the termination of the case. An important feature of these workflow systems is their ability to manage dynamic change: the structure of the Petri net should be allowed to vary as a case proceeds within the system. Petri nets however do not offer a direct way to express processes whose structure evolves along computations. For that reason they have been used in conjunction with sets of rewriting rules in order to cope with workflow systems: such a system is locally described by a Petri net while rewriting rules allow for the modification of the structure of the Petri net. The purpose of this paper is to introduce *reconfigurable nets* which is a class of high level Petri nets that can dynamically modify their own structures by rewriting some of their components thus supporting dynamic changes within workflow systems. Reconfigurable nets is a very natural extension of Petri nets, we therefore have confidence that many of the theoretical results and automated tools that exist for Petri nets could be used or adapted for them. For instance, we show in this paper that we can decide the boundedness property for reconfigurable nets using a variant of coverability trees.

The Dynamic nets of Asperti and Busi [1] is also a model that allows for dynamic changes. In Dynamic nets tokens are names for places, an input token of a transition can be used in its postset to specify a destination, and moreover the creation of new nets during the firing of a transition is also possible. The work of Asperti and Busi recasts in the context of net theory ideas and concepts that originated in the π-calculus [15] and the related join-calculus [10, 11]. It is our opinion that the intricacy of this model leaves little hope to obtain significant mathematical results and/or automated verification tools in a close future. Moreover this model is probably too sophisticated to be easily used for the modelling and design of groupware applications.

Self-modifying nets introduced by Valk [18, 19] and their subclass of *stratified Petri nets* [3] is another extension of Petri nets. Dynamicity is introduced there via self-modification. More precisely the flow relations between places and transitions in self-modifying nets are linear functions of the marking. Techniques of linear algebra used in the study of the structural properties of Petri nets can be adapted to this extended framework; in particular each transition may be associated with a matrix and the modification of the marking due to a sequence of firable transitions can be coded by the corresponding product of matrices. Even though self-modifying nets constitute a small variation in the Petri net paradigm, some important properties are lost, e.g. we cannot decide the boundedness of self-modifying nets. Moreover even if this class of nets has a clean and concise definition it is hopeless to describe and design realistic systems directly in this formalism.

It is our opinion that self-modifying nets is a very reasonable attempt to add mobility in Petri nets but that they should be used rather as a back-end model. Reconfigurable nets on the other hand constitute a more direct formalisation of the manner in which groupware applications are described using a combination of Petri nets and rewriting rules as in [8]; moreover we describe in this paper a translation of reconfigurable nets into equivalent self-modifying nets. Therefore reconfigurable nets can be viewed as a subclass of self-modifying nets for which boundedness can be decided.

The rest of the paper is organized as follows. Reconfigurable nets are introduced in Section (2) and we indicate how they can be used for modelling workflow nets with a dynamic structure, the so-called reconfigurable workflow nets. We show in Section (3) that we can decide whether a reconfigurable net is bounded by constructing its coverability tree, and that the soundness of a reconfigurable workflow net implies its boundedness and can also be verified using the coverability tree. We also prove an analogue of a result of [2] showing that the soundness of a reconfigurable workflow net reduces to the boundedness and liveness of the reconfigurable net obtained by adding an extra transition connecting the output place of the reconfigurable workflow net to its input place. In Section (4) we show that any reconfigurable net can be simulated by a self-modifying net. Finally we conclude in Section (5).

2 Reconfigurable Nets

Reconfigurable nets are high level Petri nets supporting dynamic changes within workflow systems. For instance one can admit local changes in the scheduling of the tasks required to process a case which is currently flowing in the system. If the case is an order request from a customer the involved tasks may be Order Check, Inventory Check, Credit Check, Shipping, Billing, and Archiving [8]. A dynamic change may then enable the parallel execution of two tasks (e.g. Shipping and Billing) that were previously performed sequentially in some order; it may also refine some task into more elementary tasks with a prescribed ordering. We assume however that the set of involved tasks as well as the set of local changes are known in advance and can be listed. This assumption implies that the set of *tasks instances* is finite. This set constitutes the set of transitions of the reconfigurable net. The switching from one configuration to another one due to a local change is taken care of by the introduction of a new kind of place content which denotes whether a place does exist or not in the current state of the system.

Definition 1 *A reconfigurable net is a structure $N = (P, T, F, R)$ where $P = \{p_1, \ldots, p_m\}$ is a non empty and finite set of places, $T = \{t_1, \ldots, t_n\}$ is a non empty and finite set of transitions disjoint from P ($P \cap T = \emptyset$), $F : (P \times T) \cup (T \times P) \to \mathbb{N}$ is a weighted flow relation, and $R = \{r_1, \ldots, r_k\}$ is a finite set of structure modifying rules. A structure modifying rule is a map $r : P_1 \to P_2$ whose domain and codomain are disjoint subsets of places ($P_1, P_2 \subseteq P$ and $P_1 \cap P_2 = \emptyset$). A marking of net N is a map $M : P \to \mathbb{N} \cup \{\alpha\}$ where $\alpha \notin \mathbb{N}$, when $M(p) = \alpha$ place p is said not to exist in marking M whereas $M(p) = n \in \mathbb{N}$ expresses that p exists in marking M and has value n. We let $\mathcal{M}$ denote the set of markings of net N. We let $E = T \cup R$ denote the set of events of the reconfigurable net. We let $M[e > M'$ denote the fact that event e is enabled in marking M and that the net reaches marking M' when firing this event. This transition relation is defined as follows. A transition $t \in T$ is enabled in marking M if:*

$$\forall p \in P \qquad M(p) \neq \alpha \Rightarrow M(p) \geq F(p, t)$$

When transition t is fired in marking M, the resulting marking $M[t > M'$ is such that $\forall p \in P$

$$\begin{array}{l} M(p) = \alpha \Rightarrow M'(p) = \alpha \\ M(p) \neq \alpha \Rightarrow M'(p) = M(p) - F(p,t) + F(t,p) \end{array}$$

A structure modifying rule $r \in R$ is enabled in marking M if:

$$\begin{array}{ll} \forall p \in P_1 & M(p) \neq \alpha \\ \forall p \in P_2 & M(p) = \alpha \end{array}$$

The firing of this enabled rule r produces the new marking M' defined as:

$$\begin{array}{ll} \forall p \in P_1 & M'(p) = \alpha \\ \forall p \in P_2 & M'(p) = \sum \{M(q) \mid q \in P_1 \ \wedge \ r(q) = p\} \\ \forall p \in P \backslash (P_1 \cup P_2) & M'(p) = M(p) \end{array}$$

A marked reconfigurable net is a reconfigurable net together with an initial marking.

The firing policy of transitions is like in the Petri net obtained by discarding the non existing places. This Petri net is called a *configuration* of the reconfigurable net. As long as no structure modifying rule take place, the reconfigurable net behaves exactly like this Petri net. Structure modifying rules produce a structure change in the net by removing existing places and creating new ones, thus moving the system from one configuration to another one. When a place is removed, the tokens of this place do not disappear, but they are moved to other places of the net. Hence, the number of tokens remains constant through the application of structure modifying rules. The rule defines how tokens should be moved in the net. Places of set P_2 which are not in the range of r are places created by the structure modifying rule and containing initially no token. Roughly speaking a reconfigurable net can be seen as a bunch of Petri nets (its configurations) which correspond to the various *modes* of operation of the system. The structure modifying rules allow to switch from one mode of operation to another one while it modifies the current marking accordingly: work cases are not processed during the firing of a structure modifying rule, therefore there is no creation nor disappearing of tokens, these tokens are simply displaced from vanishing places to created ones. Thus a system modelled by a reconfigurable net has the ability of dynamically change its own structure when certain conditions are met. For instance if the content of some place becomes too large (there is a large amount of work cases waiting for being processed) one can duplicate the ouput transitions of this place, technically we replace this place by a new one having twice as many output transitions playing the same role as the output transitions of the original place. Using reconfigurable nets one can also easily implement the *delayed dynamic changes* of [8]. These dynamic structural changes also called *synthetic cut-over changes* are defined as follows, quoted from [8].

> *[in a synthetic cut-over change] both the old and the new change regions are maintained in the new procedure. This ensures that tokens already in the old change region will continue their progression as if the change did not take place immediately (which justifies the attribute delayed). However tokens evolving in the context of the old change region will never enter the old change region (but possibly new change region); that is to say that in view of these tokens the change is immediate.*

We can illustrate synthetic cut-over change with the example of Fig. 1. This

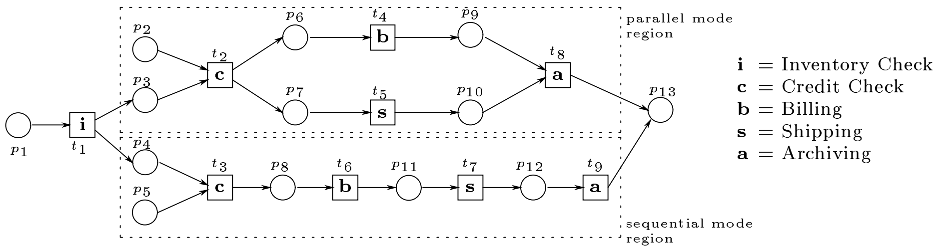

Fig. 1. synthetic cut-over change

reconfigurable net describes how to proceed an order request from a customer, there are two modes of operation corresponding to distinct regions in the graphical representation of the net, one in which the Billing and Shipping operations are processed sequentially and the other in which they are processed in parallel. The structure modifying rule $r : \{p_4; p_2\} \to \{p_3; p_5\}$ given by $r(p_4) = p_3$ and $r(p_2) = p_5$ permits to switch from the sequential mode of operation to the parallel mode of operation. Conversely the structure modifying rule r^{-1} realizes the switching in the converse direction. Figure 2 represents a fragment of the marking graph of this reconfigurable net, a place is graphically represented in a given state if and only if that place exists in the current marking (i.e. its value is different from α). Observe that when we switch from the sequential mode to the parallel mode the tokens which are in the old region (sequential mode region) *continue their progression as if the change did not take place* but the tokens in the context (in place p_1) will now enter the new region (the parallel mode region).

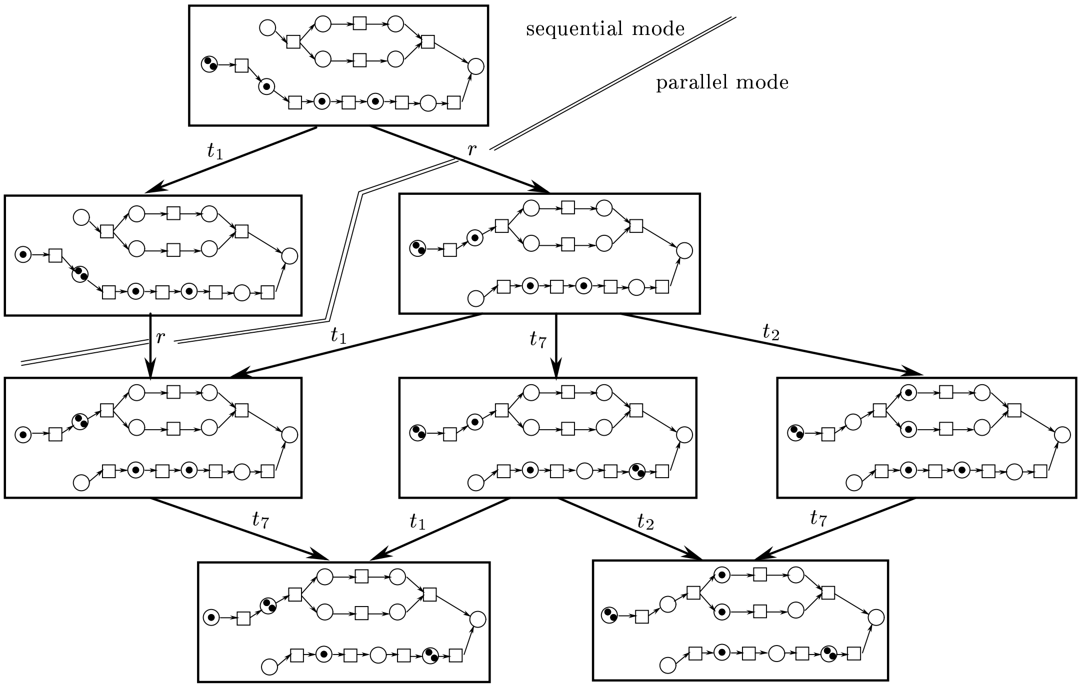

Fig. 2. (part of) the marking graph of the reconfigurable net of Fig. 1

The set of places that exists in marking M, let $D(M) = \{p \in P | \; M(p) \neq \alpha\}$, is termed the *domain* of M. Two markings are said to be equivalent when they have the same domain: $M_1 \equiv M_2 \Leftrightarrow D(M_1) = D(M_2)$. A *mode of operation* is an equivalence class for $\equiv$, it can be identified with a subset $D \subseteq P$ of places. Usually, a reconfigurable net is implicitly attached with a fixed subset of modes of operation. In the above example one has two modes of operation, the sequential mode and the parallel mode, whose respective domains are $P \setminus \{p_3; p_5\}$ and $P \setminus \{p_2; p_4\}$; any marking whose domain is different from these two sets intuitively should not correspond to any state of the system. Moreover as in [2] we consider that nets that model workflow systems have two distinctive places, an *input place* i which is a source place i.e. a place with no pre-transitions: $\forall t \in T \quad F(t, i) = 0$; and an *ouput place* o which is a sink place i.e. a place with no post-transitions

$\forall t \in T \quad F(o,t) = 0$. These places correspond respectively to the beginning and the termination of the processing of a case. In the example of Fig. 1 the input place is place p_1 and the output place is place p_{13}. We end up with the notion of a reconfigurable workflow net which is an adaptation of van der Aalst's notion of workflow net [2].

Definition 2 *A reconfigurable workflow net $\mathcal{N} = (N, \mathcal{O}, i, o)$ is a reconfigurable net $N = (P, T, F, R)$ with an explicit set of modes of operation $\mathcal{O} \subseteq 2^P$ and two distinguished places $i, o \in P$ where i is a source place and o is a sink place. Every mode of operation $\Omega \in \mathcal{O}$ contains places i and o; moreover the set $\mathcal{O}$ is strongly connected in the sense that every $\Omega \in \mathcal{O}$ derives from every other $\Omega' \in \mathcal{O}$ by a (finite) sequence $r_1, \ldots, r_n$ of structure modifying rules, where Ω derives from Ω' by a rule $r : P_1 \to P_2$ when $\Omega = (\Omega' \setminus P_1) \cup P_2$. Finally if Ω and Ω' are subsets of places such that $\Omega = (\Omega' \setminus P_1) \cup P_2$ for some structure modifying rule $r : P_1 \to P_2$, then $\Omega \in \mathcal{O} \Leftrightarrow \Omega' \in \mathcal{O}$.*

The set of modes of operation of a reconfigurable workflow net is therefore a connected component of the directed graph whose vertices are the subsets of places and whose arcs are pairs (Ω, Ω') such that $\Omega = (\Omega' \setminus P_1) \cup P_2$ for some structure modifying rule $r : P_1 \to P_2$; and moreover this component is strongly connected. If $\Omega \in \mathcal{O}$ is a mode of operation, we let i_Ω stand for the marking of domain Ω with one token in place i and no token elsewhere, and similarly o_Ω is the marking of domain Ω with one token in place o and no token elsewhere. If $U \subseteq E$ is a subset of events (usually T, R or E itself), we let $M[U\!>\!M'$ if marking M' can be reached from marking M by firing a sequence of events in U. Because of the strong connectedness we can restore any particular mode of operation before starting (or after finishing) the processing of a case: i.e. $i_\Omega[R\!>\!i_{\Omega'}$ and $o_\Omega[R\!>\!o_{\Omega'}$ for any pair $\Omega, \Omega' \in \mathcal{O}$ of modes of operation.

A token in the input place corresponds to a case entering the system. This case then *flows* through the system until a token appearing in the output place indicates the termination of this case. In the meantime the role played by the marking is twofold. On the one hand, it accounts for the current (distributed) state of progress of the case; on the other hand, it encodes the current state of the system which processes the case. Without loss of generality we can assume that the state of the system be empty when starting a new case. When the case terminates the system should have recovered its initial state in order to be ready to process a new case. The mode of operation may however have changed, but this is not significant because a case should be unaware of the current mode of operation of the system. Finally, since such a system is intended to process cases endlessly it should not have a degraded behaviour: if a transition becomes dead in some state (i.e. it cannot be fired from this state on), then this transition might have been discarded in the first place! These requirements are captured in the following definition which is an adaptation of the similar definition for workflow nets [2]. We let $M \sqsubseteq M'$ when M and M' are markings with the *same domain* D such that $\forall p \in D \quad M(p) \leq M'(p)$ and we let $\mathcal{M}$ denote the set of markings.

Definition 3 *A reconfigurable workflow net is sound if the following conditions are met.*
(i) *Every processing of a case can terminate:*

$$\forall M \in \mathcal{M} \quad \forall \Omega \in \mathcal{O} \quad i_\Omega[E{>}M \Rightarrow \exists M' \in \mathcal{M}\ \exists \Omega' \in \mathcal{O}\ (M[E{>}M' \ \wedge\ o_{\Omega'} \sqsubseteq M')$$

(ii) *Every termination of a case restores the initial state of the system (but possibly the mode of operation):*

$$\forall M \in \mathcal{M} \quad \forall \Omega, \Omega' \in \mathcal{O} \quad (i_\Omega[E{>}M \ \wedge\ o_{\Omega'} \sqsubseteq M) \Rightarrow M = o_{\Omega'}$$

(iii) *There are no dead transitions:*

$$\forall t \in T \quad \forall \Omega \in \mathcal{O} \quad \exists M, M' \in \mathcal{M} \quad i_\Omega[E{>}M \ \wedge\ M[t{>}M'$$

Observe that the processing of a case may not terminate and that a change of mode of operation may be required in order to reach termination. The above definition states some properties of the expected behaviour of a workflow system processing an individual case but it says nothing about that system when several cases are being processed concurrently. In real applications every case has an *identity*, an agent performing a task within a worflow system knows which case he is currently processing. We may consider therefore that every new case entering the system is given a *colour* and that this colour is distinct from the colours of the other cases currently flowing through the system. The resulting colored Petri net behaves as follows: in order to fire, a transition is only allowed to pick from its input places tokens of the same colour, it then produces tokens in its ouput places of that same colour. This means that the concurrent processing of multiple cases is represented as the *non interfering* superimposition of the processings of the individual cases. Therefore a reconfigurable workflow net is considered only with respect to an individual case and this justifies the above definition. A weak form of interference between cases exists however due to the fact that the current state of the system may have an effect on the decision as to whether an allowed structure modifying rule should be invoked. For example if the number of tokens in a certain place exceeds a given threshold one may regulate the flow by invoking some structure modifying rule, the converse modification may be invoked latter when the content of that place goes under another threshold; in that sense the manner in which a case is processed may be influenced by the other cases. Notice that such thresholds do not appear in our definition of structure modifying rules which reflect the fact that the decisions concerning the invocation of these rules are *external* to the system. Moreover this interference concerns only the modes of operation which are used when processing a case and our formalism allows the designer of the system to make sure that the processing of a case is insensitive to the dynamic changes occurring within the system. We can if necessary enrich the description of the net by adding extra places and transitions in order to ensure that the processing of cases are independent of the dynamic changes. For instance, one can identify a list of properties that characterize the fact that the case has been correctly processed. In general such a property corresponds to the completion of a task. These properties are represented by extra places $p_{n+1}, \ldots, p_{n+k}$. Another extra place is introduced as the new output place of the enriched net. This place is filled by an extra transition whose preconditions are the places $p_{n+1}, \ldots, p_{n+k}$ together with the old output place. In that manner a token in the new output place indicates that the case has been correctly processed regardless of the dynamic changes that may have occurred.

3 Boundedness of a Reconfigurable Net

The reachability tree of a marked reconfigurable net is the tree whose root is labelled with the initial marking and such that if V is an arbitrary vertex of that tree labelled with marking M, the arcs originating in V are in bijective correspondence with the firings $M[e\!>\!M'$ and the arc associated with $M[e\!>\!M'$ is labelled with event e and has its extremity labelled with M'. The reachability tree is thus the "unfolding" of the marking graph of the marked net. If the net is unbounded this tree is infinite. Similar to what is done for ordinary Petri nets, a finite approximation of the reachability tree called the *coverability tree* can be constructed. Two properties are at the basis of the algorithm of Karp and Miller [14]. They correspond to the two following propositions.

Proposition 4 *The order relation between markings is a well-ordering.*

Proof: We recall (see e.g. [6]) that an order relation $(X, \leq)$ is a well-ordering if for every infinite sequence $(x_i, i \in \mathbb{N})$ indices $i < j$ can be found such that $x_i \leq x_j$, equivalently if every infinite sequence in X has an infinite increasing subsequence. The usual ordering on $\mathbb{N}$ is a well-ordering. Moreover, by extracting subsequences iteratively (n times), we notice that if $(X, \leq)$ is a well-ordering, then X^n with the pointwise ordering is also a well-ordering. Finally, $\mathbb{N} \cup \{\alpha\}$ with the order $x \leq y \Leftrightarrow (x = y = \alpha \vee [x, y \in \mathbb{N} \ \wedge \ x \leq y])$ is a well-ordering. Indeed, if $(x_i, i \in \mathbb{N})$ is a sequence in $\mathbb{N} \cup \{\alpha\}$, then we can extract a subsequence which is constantly equal to α or an increasing sequence of integers according whether we respectively have an infinite number of indices $i \in \mathbb{N}$ such that $x_i = \alpha$, or $x_i \in \mathbb{N}$. Therefore the order between markings is a well-ordering. ∎

Proposition 5 *The firing rule is monotone:* $\forall e \in E(M_1[e\!>\!M_2 \ \wedge \ M_1 \sqsubseteq M_1') \ \Rightarrow \ \exists M_2' \ (M_1'[e\!>\!M_2' \ \wedge \ M_2 \sqsubseteq M_2')$*; moreover* $|M_1'| - |M_1| = |M_2'| - |M_2|$*. Thus if* $M[u\!>\!M'$ *with* $M \sqsubseteq M'$ *then the sequence* $u \in E^*$ *of firings can be reproduced, i.e.* $\exists M[u^n\!>\!M^{(n)}$ *for every* $n \in \mathbb{N}$*, and then* $|M^{(n)}| = |M| + kn$ *where* $k = |M'| - |M|$.

Proof: If $e = t \in T$ we have $(M_1[t\!>\!M_2 \ \wedge \ M_1 \sqsubseteq M_1') \ \Rightarrow \ \exists M_2' \ (M_1'[t\!>\!M_2' \ \wedge \ M_2 \sqsubseteq M_2')$ and $M_1' - M_1 = M_2' - M_2$ (the property of constant effect) as these properties hold for Petri nets. If $e = r \in R$, the first property holds trivially while the property of constant effect is weaken: as token are moved from some places to other places we only have conservation of the total number of tokens, i.e. $|M_1'| - |M_1| = |M_2'| - |M_2|$. ∎

The key observation for the algorithm of Karp and Miller for Petri nets is that, because of the property of constant effect, when markings M and M' and a sequence of firings $u \in T^*$ can be found such that $M[u\!>\!M'$ and $M \sqsubseteq M'$, one can deduce $M[u^n\!>\!M^{(n)}$ for every $n \in I\!N$. Moreover, for every place $p \in P$ such that $M(p) < M'(p)$ one has $M^{(n)}(p) = M(p) + nk$ where $k = M'(p) - M(p) > 0$ and therefore this place is not bounded. Because of the weak form of the property of constant effect, this observation no longer holds for reconfigurable nets. However if one is not concerned with the boundedness of any particular place of the net but with the boundedness of the net itself (i.e. whether there exists some place in the net that is unbounded), then the above propositions are sufficient and boundedness can be verified using the following simplified version of coverability tree.

Definition 6 *The coverability tree of a marked reconfigurable net (N, M_0) is constructed by the following algorithm:*

- *Initially the tree is reduced to its root labelled M_0 and tagged as a "new" vertex.*
- *While "new" vertices exist, do the following:*
  - *Select a new vertex V, let M be its label.*
  - *For every firing $M[e\!>\!M'$ do the following:*
    - *Create a new vertex V' labelled M' and an arc from V to V' labelled e.*
    - *If there exists some node V'' on the path from the root to vertex V whose label M'' is such that $M'' \sqsubseteq M'$ then*
      - *If $M'' = M'$ then tag vertex V' "old" else tag it "unbounded".*

      else tag V' "new".
  - *Withdraw V from the set of "new" vertices.*

Proposition 7 *The coverability tree of a marked reconfigurable net is finite.*

Proof: Since the order relation on the set of markings is a well-ordering, the coverability tree of a marked reconfigurable net contains no infinite branch. Since moreover, each vertex has at most $|E|$ successors we deduce by König lemma that this tree is finite. ∎

Proposition 8 *A marked reconfigurable net is bounded if and only if no vertex of its coverability tree is tagged "unbounded".*

Proof: If the coverability tree contains no vertex tagged "unbounded" then the set of labels of its vertices coincides with the set of markings of the reconfigurable net reachable from the initial state (label of the root), therefore the marked reconfigurable net is bounded. If on the contrary the coverability tree contains some vertex V' tagged "unbounded". Thus there exists some vertex V on the path from the root such that $M_0[u\!>\!M[v\!>\!M'$ where u labels the path from the root to vertex V, v labels the path from vertex V to vertex V', $M \sqsubseteq M'$ and $M \neq M'$. Then by Prop. 5, $M[v^n\!>\!M^{(n)}$ with $|M^{(n)}| = |M| + kn$ where $k = |M'| - |M| > 0$. Since there are finitely many places the net is unbounded. ∎

Corollary 9 *The boundedness of reconfigurable net is decidable.*

A reconfigurable workflow net N is said to be bounded if the marked reconfigurable net (N, i_Ω) for $\Omega \in \mathcal{O}$ some mode of operation is bounded. This definition does not depend on the choice of $\Omega \in \mathcal{O}$ because $i_\Omega[R\!>\!i_{\Omega'}$ for every pair $\Omega, \Omega' \in \mathcal{O}$.

Proposition 10 *A sound reconfigurable workflow net is bounded and we can decide whether a reconfigurable workflow net is sound.*

Proof: If (N, i_Ω) is not bounded then as seen in the proof of Prop. (8) there exists markings M and M' such that $i_\Omega[E\!>\!M$, $M[E\!>\!M'$, $M \sqsubseteq M'$ and $M(p) < M(p')$ for some place p. Since N is sound one has $M[u\!>\!o_{\Omega'}$ for some sequence $u \in E^*$. By monotony $M'[u\!>\!M''$ with $o_{\Omega'} \sqsubseteq M''$ and $M'' \neq o_{\Omega'}$ which contradicts the fact that N is sound. Once boundedness has been checked, the properties i to *(iii)* of Def. 3 may be checked directly on the coverability tree which then coincide with the reachability tree. ∎

However as noted by Hack in [12] "*The size of Karp and Miller's construction in their decision procedure for boundedness and coverability can grow as fast as Ackermann's function of the size of the Petri net*" which shows the intractability of the verification of soundness property via the construction of the coverability tree. Van der Aalst showed in [2] that soundness of a workflow net reduces to boundedness and liveness of a Petri net obtained by adding an extra transition connecting its output place to its input place. Since it is possible to decide boundedness and liveness of free-choice Petri nets in polynomial time [4], he deduced therefrom that soundness of free-choice workflow nets can be decided in polynomial time. We show that van der Aalst's construction can be carried to reconfigurable nets with no significant changes, unfortunately one cannot directly deduce therefrom a polynomial time algorithm for the decision of soundness of reconfigurable workflow nets all of whose configurations are free-choice Petri nets.

Definition 11 *If $\mathcal{N} = (N, \mathcal{O}, i, o)$ is a reconfigurable workflow net where $N = (P, T, F, R)$ and $\Omega \in \mathcal{O}$ is a mode of operation, we let $\overline{N}_\Omega = (\overline{N}, i_\Omega)$ be the marked reconfigurable net consisting of the reconfigurable net $\overline{N} = (P, \overline{T}, \overline{F}, R)$ and initial marking i_Ω where $\overline{T} = T \cup \{t^*\}$ with $t^* \notin T$ a new transition and the extended flow relation $\overline{F} : (P \times \overline{T}) \cup (\overline{T} \times P) \to I\!N$ is given by*

$$\overline{F}(t,p) = \begin{cases} F(t,p) & \textit{if} \quad t \in T \quad \textit{and} \quad p \in P \\ -1 & \textit{if} \quad t = t^* \quad \textit{and} \quad p = o \\ 1 & \textit{if} \quad t = t^* \quad \textit{and} \quad p = i \\ 0 & \textit{otherwise} \end{cases}$$

$\overline{N}$ is obtained from N by adding a new transition t^* which is enabled when a case has reached termination (there is one token in the output place) and then removes that case to the system and introduces a new one (by adding one token in the input place).

Proposition 12 *The reconfigurable workflow net $\mathcal{N} = (N, \mathcal{O}, i, o)$ is sound if and only if the reconfigurable net $\overline{\mathcal{N}}_\Omega$ is live and bounded.*

Proof: We first notice that since each initial state i_Ω is reachable from any other initial state by structure modifying rules ($i_\Omega[R>i_{\Omega'}$), the reconfigurable net $\overline{\mathcal{N}}_\Omega$ is live and bounded if and only if $\overline{\mathcal{N}}_{\Omega'}$ is live and bounded for any $\Omega' \in \mathcal{O}$. We first show that if $\overline{\mathcal{N}}_\Omega$ is live and bounded then $\mathcal{N}$ is a sound reconfigurable workflow net. Since $\overline{\mathcal{N}}_\Omega$ is live, transition t^* is potentially firable in every reachable marking, i.e. condition *(i)* in Def. 3 is satisfied. If $\Omega, \Omega' \in \mathcal{O}$, we let $<\Omega, \Omega'>$ denote the set of integers $n \in \mathbb{N}$ for which there exists some marking M such that $i_\Omega[E \cup \{t^*\}>M$, $i_{\Omega'} \sqsubseteq M$, and $|M| = n+1$. That is to say, by Prop. 5, that $<\Omega, \Omega'>$ records all possible increases of the size of markings along computations in $\overline{\mathcal{N}}_\Omega$ from some marking greater than i_Ω to some marking greater than $i_{\Omega'}$. Therefore $(n \in <\Omega, \Omega'> \wedge m \in <\Omega', \Omega''>) \Rightarrow n+m \in <\Omega, \Omega''>$. Now $<\Omega, \Omega'> \neq \emptyset$. Actually since condition *(i)* in Def. 3 is satisfied, $i_\Omega[E>M$ for some M such that $o_{\Omega''} \sqsubseteq M$ for some $\Omega'' \in \mathcal{O}$; since $o_{\Omega''}[R>o_{\Omega'}$ and by Prop. 5 we deduce $M[R>M'$ with $o_{\Omega'} \sqsubseteq M'$ and then $M'[t^*>M''$ with $i_{\Omega'} \sqsubseteq M''$ as required. Therefore, since $\overline{\mathcal{N}}_\Omega$ is bounded, we deduce that $<\Omega, \Omega'> = \{0\}$ for all $\Omega, \Omega' \in \mathcal{O}$, and thus $(i_\Omega[E>M \;\wedge\; o_{\Omega'} \sqsubseteq M) \Rightarrow M = o_{\Omega'}$, i.e. condition *(ii)* in Def 3 is satisfied. Condition *(iii)* in Def 3 follows from the fact that $\overline{\mathcal{N}}_\Omega$ is live for every $\Omega \in \mathcal{O}$.

Conversely, let us assume that $\mathcal{N}$ is sound.

First we show that $\overline{\mathcal{N}}_\Omega$ is bounded. Since $\mathcal{N}$ is sound, the extended net $\overline{\mathcal{N}}$ returns to some initial state i_Ω when t^* fires, it is then enough to check that the marked reconfigurable net (N, i_Ω) is bounded for every $\Omega \in \mathcal{O}$. If this is not the case, then by construction of the coverability tree, we deduce there exist markings M_1 and M_2 such that $i_\Omega[E>M_1$, $M_1[E>M_2$, $M_1 \sqsubseteq M_2$, and $M_1 \neq M_2$. By soundness of $\mathcal{N}$, we deduce $M_1[u>o_{\Omega'}$ for some $u \in E^*$, and then by Prop. 5 $M_2[u>M_2'$ with $o_{\Omega'} \sqsubseteq M_2'$ and $M_2' \neq o_{\Omega'}$ (because $|M_2'| - 1 = |M_2| - |M_1| > 0$) which contradicts soundness of $\mathcal{N}$.

Second we show that $\overline{\mathcal{N}}_\Omega$ is live. Since $\mathcal{N}$ is sound, transition t^* is potentially firable in every reachable marking and its firing always leads to some initial state i_Ω, since moreover $i_\Omega[R>i_{\Omega'}$ for arbitrary pair of mode of operations, we deduce that net $\overline{\mathcal{N}}_\Omega$ is cyclic (every initial state i_Ω and thus any reachable marking is reachable from any reachable marking). Since moreover there is no dead transitions in $\overline{\mathcal{N}}_\Omega$ (by condition *(iii)* in Def. 3 and the fact that t^* is not dead) we deduce that this marked net is live. ∎

4 Reconfigurable Nets as Self-Modifying Nets

The purpose of this section is to show that reconfigurable nets are self-modifying nets. Self-modifying nets [18, 19] are generalizations of place/transition nets where the flow relation between a place and a transition depends on the marking.

Definition 13 *A self-modifying net is a structure $N = (P, T, F)$ where $P = \{p_1, \ldots, p_m\}$ is a non empty and finite set of places, $T = \{t_1, \ldots, t_n\}$ is a non empty and finite set of transitions disjoint from P, and $F : (P \times T) \cup (T \times P) \to \mathbb{N}^{P_*}$ is the flow relation where $P_* = P \cup \{*\}$ and $* \notin P$. A vector $\varphi \in \mathbb{N}^{P_*}$ can be represented by a formal sum $\varphi = \lambda_0 + \sum_{i=1}^{m} \lambda_i \cdot p_i$ where the constant coefficient is the entry corresponding to the fictituous place: $\lambda_0 = \varphi(*)$ and $\lambda_i = \varphi(p_i)$. A marking of net N is a map $M : P \to \mathbb{N}$. If $M \in \mathbb{N}^P$ is a marking and $\varphi \in \mathbb{N}^{P_*}$, we let $\varphi(M) = \lambda_0 + \sum_{i=1}^{m} \lambda_i \cdot M(p_i)$ denote the evaluation of the affine function φ in marking M. We let $M[t{>}M'$ when transition t is enabled in marking M and leads to marking M'. This transition relation is given by:*

$$M[t{>}M' \Leftrightarrow \forall p \in P \;\; M(p) \geq F(p,t)(M) \quad \wedge \quad M' = M - F(p,t)(M) + F(t,p)(M)$$

A marked self-modifying net is a self-modifying net together with an initial marking.

Proposition 14 *Any marked reconfigurable net can be associated with a marked self-modifying net with isomorphic marking graph and whose set of transitions is the set of events of the reconfigurable net.*

Proof: The translation of a reconfigurable net into an equivalent self-modifying net is staightforward: we represent each place p of the reconfigurable net by three places $\exists_p$, $\neg\exists_p$ and p. The first two places are complementary 1-bounded places whose contents indicate whether place p exists in the current marking and the third place, also denoted p, has the same content than the original place p when this place exists. Figure 3 gives a sketch of the translation whose precise definition follows. Any reconfigurable net $N = (P, T, F, R)$ is associated

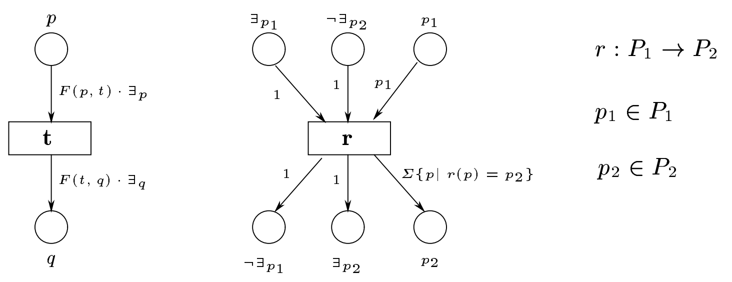

Fig. 3. translating a reconfigurable net into an equivalent self-modifying net

with a self-modifying net $\tilde{N} = (\tilde{P}, \tilde{T}, \tilde{F})$ defined as follows. The set of places $\tilde{P} = \exists P \cup \neg\exists P \cup P$ is the disjoint union of three copies of set P whose respective typical elements are noted $\exists_p$, $\neg\exists_p$ and p for p ranging in P. The set of transitions $\tilde{T} = T \cup R$ consists of the transitions of the original net together with its set of structure modifying rules, i.e. its set of events. Finally the flow relation $\tilde{F}$ is given by the following identities where $\tilde{p}$, t and $r : P_1 \to P_2$ range respectively in $\tilde{P}$, T and R.

$$\tilde{F}(\tilde{p},t) = \begin{cases} F(p,t)\cdot \exists_p \text{ if } & \tilde{p} = p \in P \\ 0 & \text{otherwise} \end{cases} \quad \tilde{F}(t,\tilde{p}) = \begin{cases} F(t,p)\cdot \exists_p \text{ if } & \tilde{p} = p \in P \\ 0 & \text{otherwise} \end{cases}$$

$$\tilde{F}(\tilde{p},r) = \begin{cases} 1 \;\; \text{if} \;\; \tilde{p} = \exists_{p_1} \wedge p_1 \in P_1 \\ \quad\;\; \text{or} \;\; \tilde{p} = \neg\exists_{p_2} \wedge p_2 \in P_2 \\ p_1 \; \text{if} \;\; \tilde{p} = p_1 \in P_1 \\ 0 \;\; \text{otherwise} \end{cases} \quad \tilde{F}(r,\tilde{p}) = \begin{cases} 1 \;\; \text{if} \;\; \tilde{p} = \neg\exists_{p_1} \wedge p_1 \in P_1 \\ \quad\;\; \text{or} \;\; \tilde{p} = \exists_{p_2} \wedge p_2 \in P_2 \\ \sum\{p|r(p) = p_2\} \;\; \text{if} \;\; \tilde{p} = p_2 \in P_2 \\ 0 \;\; \text{otherwise} \end{cases}$$

A marking M of the reconfigurable net N is associated with the marking $\tilde{M}$ of the self-modifying net $\tilde{N}$ given by

$$\begin{array}{ll} \tilde{M}(\exists_p) & = \text{if } M(p) \neq \alpha \text{ then } 1 \quad\;\;\;\; \text{else } 0 \\ \tilde{M}(\neg\exists_p) & = \text{if } M(p) \neq \alpha \text{ then } 0 \quad\;\;\;\; \text{else } 1 \\ \tilde{M}(p) & = \text{if } M(p) \neq \alpha \text{ then } M(p) \text{ else } 0 \end{array}$$

The above relations induce a bijective correspondance between the markings of N and those markings $\tilde{M}$ of $\tilde{N}$ such that $\forall p \in P \quad \tilde{M}(\exists_p), \tilde{M}(\neg\exists_p) \in \{0;1\}$ and $\tilde{M}(\exists_p) = 1 \Leftrightarrow \tilde{M}(\neg\exists_p) = 0$ and $\tilde{M}(\exists_p) = 0 \Rightarrow \tilde{M}(p) = 0$. A direct comparison of Def. 1 and Def. 13 shows that an event $e \in T \cup R$ of N is enabled in a marking M of the reconfigurable net N if and only if as a transition of the self-modifying net $\tilde{N}$ it is enabled in the associated marking $\tilde{M}$; moreover $M[e{>}M'$ in N if and only if $\tilde{M}[e{>}\tilde{M}'$ in $\tilde{N}$. Therefore the mapping $\tilde{(\cdot)}$ is an isomorphism between the marking graph of the marked reconfigurable net (N, M) and the marking graph of the marked self-modifying net $(\tilde{N}, \tilde{M})$ for any marking M of N. ∎

Boundedness is not decidable for self-modifying nets whereas it is decidable for reconfigurable nets. In order to better delimit the borderline between those self-modifying nets for which boundedness can be decided from those for which it cannot, let us precise some terminology. A pair $(p,t) \in P \times T$ is termed an *input arc* (with respect to transition t) if $F(p,t) \neq 0$. Similarly a pair $(t,p) \in T \times P$ such that $F(t,p) \neq 0$ is termed an *output arc*. An input arc (p,t) is an ordinary arc if $F(p,t) \in I\!N$, and it is a reset arc if $F(p,t) = p$. An output arc (t,p) is an ordinary arc if $F(t,p) \in I\!N$. A self-modifying net is a *post-self-modifying net* (respectively a *pre-self-modifying net*) [18] if every input arc (resp. ouput arc) is an ordinary arc; and it is a *Reset/Set nets with infinite capacity* [13] if every input arc is either an ordinary arc or a reset arc. Valk proved in [18] that boundedness of post-self-modifying nets can be decided. The same result has been erroneously stated for the class of Reset/Set nets with infinite capacity and for the class of pre-self-modifying nets. Indeed Dufourd [7] has proved recently that boundedness is indecidable for the class of Petri nets with reset arcs, i.e. for the class of self-modifying nets such that every input arc is either an ordinary arc or a reset arc and every output arc is an ordinary arc.

5 Conclusion

In this paper we have introduced a class of high level Petri nets, called *reconfigurable nets*, that can dynamically modify their own structures by rewriting some of their components. Boundedness of a reconfigurable net can be decided by constructing its coverability tree. Moreover such a net can be simulated by a self-modifying Petri net. The class of reconfigurable nets thus provide a subclass of self-modifying Petri nets for which boundedness can be decided. Delayed dynamic changes within workflow systems in the sense of [8] can then be handled in an extension of van der Aalst's workflow nets [2]. For this class (the *reconfigurable workflow nets*), a notion of soundness has been defined that can be verified using the coverability tree construction.

A reconfigurable net can be seen as a bunch of Petri nets: its configurations. A configuration of a reconfigurable net gives a description of the system for some mode of operation. It could be interesting to investigate the properties of a reconfigurable net in relationship with specific assumptions on its configurations, e.g. according whether they are acyclic, 1-safe or free-choice. The workflow net models of [5] for instance are acyclic, extended free-choice elementary net systems, whereas the worflow nets of [2] are usually assumed to be free-choice or almost free-choice. An open question in that direction is whether there exists a polynomial time algorithm for deciding the soundness property of free-choice reconfigurable workflow nets. Additional assumptions concerning the set of structure modifying rules may also be considered. For instance in a forthcomming paper we shall restrict our attention to the class of reconfigurable nets whose structure modifying rule $r : P_1 \to P_2$ are bijections. Under some extra assumption such a net can be simulated by a stratified Petri net [3], that is to say by a self-modifying Petri net for which a stratification of the set of places into layers exists so that the flow relations attached to a place involve only the content of places of lower layers. The self-modifying Petri nets that we have used to simulate reconfigurable nets were not stratified, and this was essential since reconfigurable nets unlike stratified Petri nets are in general not *reversible* in the sense that we cannot for each firing $M[e\!>\!M'$ deduce marking M from the data of event e and marking M'; reconfigurable nets are reversible however if all structure modifying rules are assumed to be bijective.

We have assumed, as it is implicitly done in [2] for workflow nets, that the system can identify, e.g. by using coloured tokens, each of the cases that are processed. Since a marking accounts at the same time for the current states of progress of the cases and the state of the system which processes these cases, this implies that the state of the system itself be multi-coloured, i.e. it is a vector of states associated with each of the cases currently flowing within the system. This assumption may be considered indesirable and we may seek for a non interference condition to be added to the definition of soundness that will ensure that the behaviour is not changed when colours are forgotten.

Finally, it might be interesting to investigate the notion of a *controlled* reconfigurable net which stands for a reconfigurable net together with a control part that regulate the flow in the system. This control would have the marking of the evolving net as input and for a set of allowed structure modifying rules as output.

References

[1] Asperti, A., and Busi, N., *Mobile Petri Nets.* Technical Report UBLCS-96-10, University of Bologna, Italy (1996).

[2] van der Aalst, W.M.P., *Verification of Workflow Nets.* Proceedings of icatpn'97, volume 1248 of Lecture Notes in Computer Science, Springer Verlag (1997) 407–426.

[3] Badouel, E., and Darondeau, Ph., *Stratified Petri Nets.* Proceedings of fct'97, volume 1279 of Lecture Notes in Computer Science, Springer Verlag (1997) 117–128.

[4] Desel, J., and Esparza, J., *Free Choice Petri Nets.* Volume 40 of Cambridge Tracts in Theoretical Computer Science. Cambridge University Press (1995).

[5] De Michelis, G., and Ellis, C.A., *Computer Supported Cooperative Work and Petri Nets.* Third Advanced Course on Petri Nets, Dagstuhl Castle, Germany (1996). To appear in Springer Verlag Lecture Notes in Computer Science.

[6] Diestel, R., *Graph Theory.* volume 173 of *Graduate Texts in Mathematics.* Springer Verlag (1996).

[7] Dufourd, C., Finkel, A., and Schnoebelen, Ph., *Reset nets between decidability and undecidability.* To appear in Proceedings of icalp'98.

[8] Ellis, C., Keddara, K., and Rozenberg, G., *Dynamic Change within Workflow Systems.* Proceedings of the Conference on Organizational Computing Systems, acm Press, New York (1995) 10–21.

[9] Ellis, C.A, and Nutt, G.J., *Modeling Enactment of Workflow Systems.* Proceedings of icatpn'93, volume 691 of Lecture Notes in Computer Science, Springer Verlag (1993) 1–16.

[10] Fournet, C., and Gonthier, G., *The reflexive chemical abstract machine and the join-calculus.* Proceedings of the 23^{rd} acm Symposium on Principle of Programming Languages, (1996).

[11] Fournet, C., Gonthier, G., Lévy, J.-J., and Rémy, D., *A Calculus of Mobile Agents.* Proceedings of concur'96, volume 1119 of Lecture Notes in Computer Science, Springer Verlag (1996) 406–421.

[12] Hack, M.H.T., *The recursive equivalence of the reachability problem and the liveness problem for Petri nets and vector addition systems.* Computation Structures Group Memo 107. Cambridge, Massachussetts: mit, Department of Electrical Engineering (1974).

[13] Heinemann, B., *Subclasses of Self-Modifying Nets.* In C. Girault and W. Reisig (Eds.) First European Workshop on Application and Theory of Petri Nets. Informatik Fachberichte, Springer (1982) 187–192.

[14] Karp, R.M., and Miller, R.E., *Parallel program schemata.* Journal of Computer and System Sciences vol. 3 (1969) 147–195.

[15] Milner, R., Parrow, J., and Walker, D., *A calculus of mobile Processes, I-II.* Information and Computation, vol. 100, no 1, (1992) 1–40 and 41–77.

[16] Murata, T., *Petri Nets: Properties, Analysis and Applications.* Proceeding of the ieee, 77(4) (1989) 541–580.

[17] Petri, C. A., *Kommunikation mit Automaten*, Schriften des IIM Nr. 2, Institut für Instrumentelle Mathematik, Bonn (1962). *English translation*: Technical Report RADC-TR-65-377, Griffiths Air Force Base, New York, vol. 1, suppl. 1 (1966).

[18] VALK, R., *Self-Modifying Nets, a Natural Extension of Petri Nets.* Proceedings of Icalp'78, Lecture Notes in Computer Science vol. 62 (1978) 464–476.
[19] VALK, R., *Generalizations of Petri Nets.* Proceedings of MFCS'81, Lecture Notes in Computer Science vol. 118 (1981) 140–155.