

Can I Find a Partner?

Peter Massuthe¹, Alexander Serebrenik², Natalia Sidorova², and Karsten Wolf³

¹ Humboldt-Universität zu Berlin, Institut für Informatik
Unter den Linden 6, 10099 Berlin, Germany

`massuthe@informatik.hu-berlin.de`

² Eindhoven University of Technology

Department of Mathematics and Computer Science

P.O. Box 513, 5600 MB Eindhoven, The Netherlands

`{a.serebrenik, n.sidorova@tue.nl}@tue.nl`

³ Universität Rostock, Institut für Informatik

18051 Rostock, Germany

`karsten.wolf@informatik.uni-rostock.de`

Abstract. We study *open nets* as Petri net models of web services, with a link to the practically relevant language WS-BPEL. For those nets, we investigate the problem of *serviceableness* which we consider as fundamental as the successful notion of *soundness* for workflow nets, i.e. Petri net models of business processes and workflows. While we could give algorithmic solutions to the serviceableness problem for subclasses of open nets in earlier work, this article shows that the problem is in general undecidable.

Key words: Web Service; Petri net; Open net; Serviceableness; Theory of Computation.

1 Introduction

Service oriented computing (SOC) and service oriented architectures are about to become established paradigms for the design of interorganizational workflows and for the programming-in-the-large. In practice, languages like WS-BPEL [1] have been developed and are increasingly used. These developments are accompanied by considerable efforts to support SOC with formal models.

One purpose of a formal model for a web service is the capability of performing formal analysis. In this article, we study the analysis problem of *serviceableness* for open nets. Open nets [2] are a special subclass of classical Petri net models that explicitly model communication with the environment. An open net is called *serviceable* if there exists an environment such that the cooperation of the net and the environment leads to the desired final state.

In our previous work [3,4] we have proposed decision procedures for serviceableness for restricted classes of open nets. In this article, we provide some evidence that these restrictions are in part essential. We show that *serviceableness of open nets is undecidable in the general case*. To prove the undecidability

we use a reduction from the language inclusion problem for Petri nets, known to be undecidable [5].

Related work. The notion of serviceableness is related to two well-known behavioral notions. First of all, serviceableness has been inspired by the well-studied notion of *soundness* for workflow nets, Petri nets used to model workflows [6]. The adaptation of soundness to the web service case has been discussed in [7, 3, 8, 4]. Moreover, the studied problem is in fact a control theoretic problem where the given open net assumes the role of the plant while the partner serves as controller. Unfortunately, classical control theory [9, 10] seems not to provide a result that is immediately applicable to our specific setting.

Reduction from a well-known undecidable problem is a common approach to prove undecidability. Reduction from the Petri net language inclusion problem has been used, e.g., in [11] for showing undecidability of some equivalence problems in a similar setting involve open nets. However, there is no direct implication in either direction between the results of [11] and ours.

2 Basic Definitions

In this paper we consider a subclass of Petri nets, called *open nets*. We recall basic notions related to Petri nets and then introduce the subclass of interest.

2.1 Petri Nets

\mathbb{N} denotes the set of natural numbers.

Let P be a set. A *bag* (*multiset*) m over P is a mapping $m : P \rightarrow \mathbb{N}$. We identify a bag with all elements occurring only once with the set containing the elements of the bag. The set of all bags over P is denoted by \mathbb{N}^P . We use $+$ and $-$ for the sum and the difference of two bags and $=, <, >, \leq$ and \geq for the comparison of bags, which are defined in a standard way. We overload the set notation, writing \emptyset for the empty bag and \in for the element inclusion. We write e.g. $m = 2[p] + [q]$ for a bag m with $m(p) = 2$, $m(q) = 1$, and $m(x) = 0$ for all $x \notin \{p, q\}$. As usual, $|m|$ and $|S|$ stand for the number of elements in bag m and in set S , respectively.

For (finite) *sequences* of elements over a set P we use the following notation: The empty sequence is denoted with ϵ ; a non-empty sequence can be given by listing its elements.

Next we introduce a number of notions related to Petri nets.

Definition 1 (Petri net). A Petri net N over a fixed set of labels Act is a tuple $\langle P, T, F, \Lambda \rangle$, where: (1) P and T are two disjoint non-empty finite sets of places and transitions respectively; (2) $F : (P \times T) \cup (T \times P) \rightarrow \mathbb{N}$ is a flow relation mapping pairs of places and transitions to the naturals; (3) $\Lambda : T \rightarrow Act \cup \{\tau\}$ is a labeling function that maps transitions of T to action labels from Act or to the special label τ denoting a silent action.

We present nets with the usual graphical notation.

Given a transition $t \in T$, the *preset* $\bullet t$ and the *postset* $t \bullet$ of t are the *bags* of places where every $p \in P$ occurs $F(p, t)$ times in $\bullet t$ and $F(t, p)$ times in $t \bullet$. Analogously we write $\bullet p, p \bullet$ for pre- and postsets of places. We also write $\bullet_N(t)$ and $(t)_N^\bullet$ to explicitly denote the Petri net.

A marking m of N is a bag over P ; markings are states (configurations) of a net. A pair (N, m) is called a *marked* Petri net. A transition $t \in T$ is *enabled* in marking m if and only if $\bullet t \leq m$. An enabled transition t may *fire*. This results in a new marking m' defined by $m' \stackrel{\text{def}}{=} m - \bullet t + t \bullet$. In this case we also write $m \xrightarrow{\Lambda(t)} m'$. For a sequence of action names $\sigma = a_1 \dots a_n$ we write $m \xrightarrow{\sigma} m'$ when $m = m_1 \xrightarrow{a_1} m_2 \dots \xrightarrow{a_{n-1}} m_n = m'$. We say that m' is *reachable* from m if and only if there exists $\sigma \in \text{Act}^*$ such that $m \xrightarrow{\sigma} m'$. The *strong language* $\mathfrak{L}(N, m_0)$ of a marked Petri net $\langle N, m_0 \rangle$ is defined as $\{\sigma \mid \sigma \in \text{Act}^*, \exists m : m_0 \xrightarrow{\sigma} m\}$. We denote the set of all markings reachable in net N from marking m as $\mathcal{R}_N(m)$. Furthermore, we write $m \Longrightarrow m'$ when $m = m'$ or $m \xrightarrow{\tau} \dots \xrightarrow{\tau} m'$. We also write $m \xRightarrow{a} m'$ if $m \Longrightarrow m_1 \xrightarrow{a} m_2 \Longrightarrow m'$. The *weak language* $\mathfrak{L}^\tau(N, m_0)$ of a marked Petri net (N, m_0) is defined as $\{\sigma \in \text{Act}^* \mid \exists m : m_0 \xRightarrow{\sigma} m\}$.

2.2 Open Nets

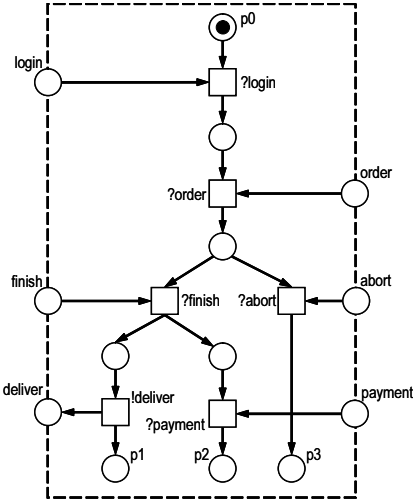
Open nets are a special class of Petri nets intended for modelling asynchronously communicating components. Interface ports are modelled as specially indicated places. Components can then be connected to each other by merging (some of) their interface places.

Definition 2 (Open net). *An open net N is a tuple $\langle P, T, F, \Lambda, P_i, P_o \rangle$, where:*
(1) $\langle P \cup P_i \cup P_o, T, F, \Lambda \rangle$ is a Petri net; (2) P, P_i and P_o are disjoint, called internal, input and output places; (3) $F(t, p_i) = F(p_o, t) = 0$ for any $p_i \in P_i, p_o \in P_o$, and $t \in T$; (4) for any $p_i \in P_i$ there exists $t \in T$ such that $F(p_i, t) \geq 1$; (5) for any $p_o \in P_o$ there exists $t \in T$ such that $F(t, p_o) \geq 1$.

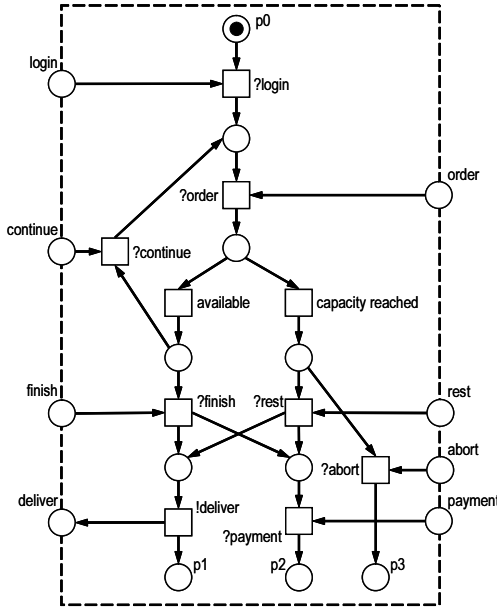
The input places P_i correspond to channels for outgoing messages, the output places P_o —to channels for the incoming messages.

As an example, Figure 1 shows two online shops modeled as open nets N_1 and N_2 . A customer of the top shop (a) may login first, followed by an order message. The customer then has the choice to confirm (finish) the order or to abort. In the first case, the shop sends the ordered goods to the customer (deliver) and expects the payment. The bottom shop (b) introduces a check whether the ordered good is in stock (modeled by transition available) or not (capacity reached). In the first case, the customer has the choice to place more orders (continue) or to finish whereas in the second case, he may choose to get all available goods delivered (rest) or to abort completely. If at least one item was bought, the shop sends the ordered goods to the customer (deliver) and expects the payment.

Definition 3 (Composition of open nets). *Let $N = \langle P, T, F, \Lambda, P_i, P_o \rangle$ and $N' = \langle P', T', F', \Lambda', P'_i, P'_o \rangle$ be two open nets with $P \cap P' = \emptyset, T \cap T' = \emptyset$,*



(a)



(b)

Fig. 1. Two online shops modeled as open nets N_1 (a) and N_2 (b).

$P_i \cap P'_i = \emptyset$ and $P_o \cap P'_o = \emptyset$, the composition $N \oplus N'$ is the open net $\langle P \cup P', T \cup T', F^\oplus, (P_i \cup P'_i) \setminus (P_o \cup P'_o), (P_o \cup P'_o) \setminus (P_i \cup P'_i) \rangle$, where $F^\oplus(x, y) = F(x, y)$ if $(x, y) \in (P \times T) \cup (T \times P)$, $F'(x, y)$ if $(x, y) \in (P' \times T') \cup (T' \times P')$, and 0 otherwise.

Note that the composition of an arbitrary number of components (open nets) is a component (open net) again.

For every component we identify a unique initial marking m_0 of the internal places, and a non-empty and possibly infinite set Ω of final markings of the internal places, which correspond to correct termination states of the components. The interface places should be empty both at the beginning and at the end of the process. Given two open nets $N = \langle P, T, F, \Lambda, P_i, P_o \rangle$ and $N' = \langle P', T', F', \Lambda', P'_i, P'_o \rangle$ with initial markings m_0 and m'_0 , respectively, the initial marking $m_0 \oplus m'_0$ of $N \oplus N'$ is defined as $(m_0 \oplus m'_0)(p) = m_0(p)$ if $p \in P$, $m_0 \oplus m'_0(p) = m'_0(p)$ if $p \in P'$ and $m_0 \oplus m'_0(p) = 0$ for $p \in (P_i \cup P_o \cup P'_i \cup P'_o)$. The set $\Omega \oplus \Omega'$ of final markings is defined as the set of the markings that correspond to the compositions of all pairs of markings from Ω and Ω' .

One clearly expects that a component built can operate properly at least in some environment. By proper operation we mean that the composition can still reach a final marking at any execution moment. We capture this requirement in the notion of *serviceableness* (cf. [4]).

Definition 4 (Serviceableness). *An open net $N = \langle P, T, F, \Lambda, P_i, P_o \rangle$ with the initial marking m_0 and the set of final markings Ω is called serviceable if there exists an open net $N' = \langle P', T', F', \Lambda', P'_o, P'_i \rangle$ with the initial marking m'_0 and the set of final markings Ω' such that for any marking $m \in \mathcal{R}_{N \oplus N'}(m_0 \oplus m'_0)$ there exists a marking $m_f \in \Omega \oplus \Omega'$ such that $m_f \in \mathcal{R}_{N \oplus N'}(m)$. N' as above is called a partner of N .*

Reconsider two online shops presented at Figure 1. We define the corresponding sets of final markings as $\Omega_{N_1} = \Omega_{N_2} = \{\mathbf{p1}, \mathbf{p2}, \mathbf{p3}\}$. It is easy to see that the open net corresponding to the top shop (a) is serviceable. The internal choice of the bottom shop (b), however, is not communicated to the customer causing the net to be non-serviceable: after ordering the first item, the customer had to guess whether to send continue/finish or to send abort/rest.

3 Undecidability of Serviceableness

In this section we present the main result: undecidability of serviceableness for open nets. To this end we start by recalling the well-known undecidability of Petri net language inclusion.

Theorem 1 (Theorem 10.2 from [5]). *Language inclusion for Petri nets is undecidable both for strong and weak languages.*

Therefore, to establish undecidability of the serviceableness we assume *ad absurdum* that it is decidable and show that language inclusion for Petri nets should be decidable as well, contradicting Theorem 1.

Theorem 2. *Serviceableness is undecidable.*

Proof. We prove undecidability by providing a construction for an open net $N = \langle P, T, F, \Lambda, P_i, P_o \rangle$ based on two given Petri nets $N_1 = \langle P_1, T_1, F_1, \Lambda_1 \rangle$ and $N_2 = \langle P_2, T_2, F_2, \Lambda_2 \rangle$ with the initial markings m_1 and m_2 respectively, such that N is serviceable if and only if the strong language of (N_1, m_1) is not included in the strong language of (N_2, m_2) . In other words, a partner, if exists, should “provide” N with a word in $\mathfrak{L}(N_1, m_1)$ and not in $\mathfrak{L}(N_2, m_2)$.

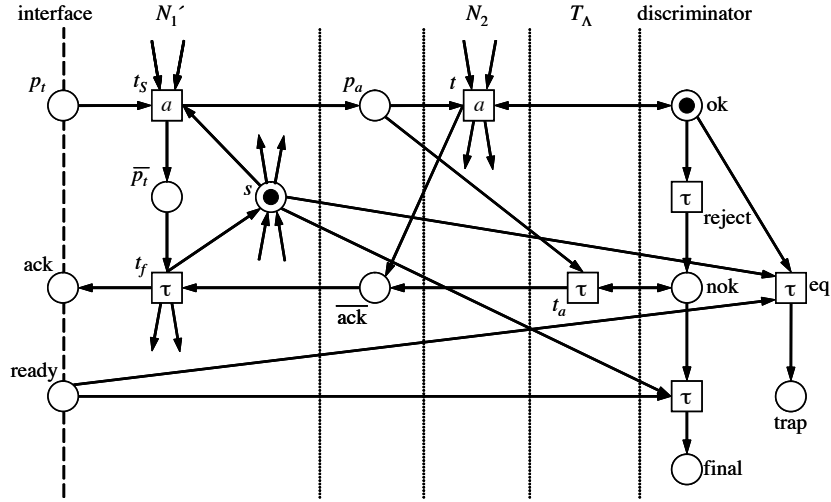


Fig. 2. Visualization of the construction.

Construction We build an auxiliary net $N'_1 = \langle P'_1, T'_1, F'_1, \Lambda'_1 \rangle$ by substituting every transition $t \in T_1$ by a sequence of the start-transition t_s and the finish-transition t_f connected via the place \overline{p}_t , such that $\bullet_{N'_1}(t_s) = \bullet_{N_1}(t)$, $(t_f)_{N'_1} = (t)_{N_1}$, and $(t_s)_{N'_1} = \bullet_{N'_1}(t_f) = [\overline{p}_t]$. Furthermore, we introduce an additional place s such that $\bullet s = \{t_f \mid t \in T_1\}$ and $s^\bullet = \{t_s \mid t \in T_1\}$. We further define $\Lambda'_1(t_s) = \Lambda_1(t)$, $\Lambda'_1(t_f) = \tau$ and $m'_1 = m_1 + [s]$. Clearly, $\mathfrak{L}(N_1, m_1) = \mathfrak{L}^\tau(N'_1, m'_1)$. Further, note that s enforces the alternation of start- and finish-transitions.

Next we connect N'_1 and N_2 in such a way that the firing of a transition t in N_2 can only directly follow the firing of a transition t_s in N'_1 with $\Lambda'_1(t_s) = \Lambda_2(t)$. For this purpose we add place p_a , where $a \in Act$, that has all transitions of N'_1 labelled with a as input transitions and all transitions of N_2 labelled with a as output transitions, namely $\bullet p_a = \{t \in T'_1 \mid \Lambda'_1(t) = a\}$ and $\{t \in T_2 \mid \Lambda_2(t) = a\} \subset p_a^\bullet$. The firings of transitions in N_2 move the token from a p_a place to the \overline{ack} place. The token on the \overline{ack} place allows N'_1 to proceed with a firing of the corresponding t_f transition.

Further we add a set of transitions T_Λ where every transition t_a corresponds to a label a from $\Lambda_1(T_1)$. The firing of t_a can substitute the firing of a correspond-

ingly labelled transition in N_2 , independently of the ability of N_2 to imitate the step of N'_1 . Similarly to the firings of transitions of N_2 , a transition t_a of T_A moves a token from the p_a place to the \overline{ack} place.

Now we add a discriminator construction (see Figure 2) whose functioning will result either in a token on place *trap* not belonging to any final marking, or in a token on place *final* that belongs to every final marking of N . The token on *trap* is reached only if net N_2 has repeated all the steps of N'_1 . Initially the discriminator contains a token on place *ok* while the *nok* place is empty, i.e. the firings of transitions of N_2 are not restricted, while the firings of transitions from T_A are forbidden. In case N_2 is not able to repeat a step of N'_1 , the only way to proceed is to move the token from *ok* to *nok*, thus enabling the firings of transitions from T_A and disabling the firing of transitions of N_2 forever. In this case we cannot reach the [*trap*] marking anymore but can reach the [*final*] marking. Note that the *reject* transition can also be taken when N_2 would be able to mimic the step of N'_1 . For any reachable marking m of N , $m(ok) + m(nok) + m(final) + m(trap) = 1$.

Finally, we extend our net with the interface places: for each transition t in T_1 one input place p_t , a special input place *ready* and a special output place *ack*. By placing tokens in p_t a partner “agrees” to a firing of t if it is enabled in N_1 . By placing a token in *ready* the partner indicates that the input is terminated and the discriminator may move a token to the *final* or *trap* places. The *ack* place serves for sending a confirmation from N'_1 that a step has been taken.

The initial marking of N is $m'_1 + m_2 + [ok]$. The set Ω of final markings is defined as the set of all markings of N that contain a token on the *final* place. Note that for every $a \in A_1(T_1)$ and any reachable marking m , $m(p_a) + m(\overline{ack}) + m(s) + m(final) + m(trap) = 1$.

Correctness proof (\Rightarrow): First we show that $\mathfrak{L}(N_1, m_1) \not\subseteq \mathfrak{L}(N_2, m_2)$ implies that N is serviceable. Since $\mathfrak{L}(N_1, m_1) \not\subseteq \mathfrak{L}(N_2, m_2)$, there exists a firing sequence $\sigma = t_1 t_2 \dots t_n \in T_1^*$ corresponding to a word $w \in \mathfrak{L}(N_1, m_1) \setminus \mathfrak{L}(N_2, m_2)$. Take the partner net $N^p = \langle P^p, T^p, F^p, A^p, P_i^p, P_o^p \rangle$ (see Figure 3) with the initial marking $[p_1]$ and the set of final markings $\{[q_{n+1}]\}$ where

- $P^p = \{p_1, \dots, p_{n+1}\} \cup \{q_1, \dots, q_{n+1}\}$,
- $T^p = \{u_1, \dots, u_{n+1}\} \cup \{v_1, \dots, v_n\}$,
- $P_i^p = \{ack\}$, $P_o^p = \{p_{t_j} \mid t_j \in \sigma\} \cup \{ready\}$ ⁴,
- $\bullet u_j = [p_j]$, $j = 1, \dots, n+1$; $u_j^\bullet = [q_j, p_{t_j}]$, $j = 1, \dots, n$; $u_{n+1}^\bullet = [q_{n+1}, ready]$,
- $\bullet v_j = [q_j, ack]$, $j = 1, \dots, n$; $v_j^\bullet = [p_{j+1}]$, $j = 1, \dots, n$,
- $A^p(x) = \tau$ for any $x \in T^p$.

Since $w \in \mathfrak{L}(N_1, m_1) \setminus \mathfrak{L}(N_2, m_2)$, every execution sequence will contain the firing of the *reject* transition, and thus will lead to *final* and not to *trap*, implying that N is serviceable.

(\Leftarrow): Now suppose that $\mathfrak{L}(N_1, m_1) \subseteq \mathfrak{L}(N_2, m_2)$. Then N_2 can mimic any firing sequence of N'_1 and the discriminator can avoid the firing of the *reject* transition.

⁴ Note that if t_i coincides with t_j for some $1 \leq i < j \leq n$, the corresponding output places p_{t_i} and p_{t_j} coincide as well.

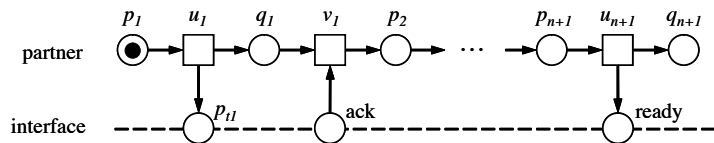


Fig. 3. Partner of the open net in Figure 2.

Note that any final marking of N contains a token on the place *final*, which can only be reached after the token on *ready* has appeared. However whenever a token on *ready* appears N is free to choose the firing of the transition eq leading to a non-final terminal marking. \square

4 Conclusions

We proved undecidability of the serviceableness problem for general open nets. We argued with a reduction from the language inclusion problem for Petri nets which is known to be undecidable.

Given the undecidability result stated in Theorem 2, to achieve decidability of serviceableness one has to restrict classes of open nets. In our previous work [3, 4] we have identified such classes and proposed decision procedures for them. In [4], we proposed an algorithm for deciding serviceableness in the case of acyclic open nets, based on the construction of a most permissive partner net. [3] proposes a construction of a most permissive partner for open nets with three restrictions: First, it assumes an open net with finitely many inner markings (inner markings abstract from tokens on interface places). Second, we only study deadlock freedom of the composed system which is weaker than serviceableness. Third, existential quantification is restricted to those partners where communication does never put more than k tokens on an interface place, for some given k . Despite the second restriction, we believe that serviceableness as such is decidable under the first and third restriction.

As the *future work* we consider investigating decidability of serviceableness for the following two classes of open nets. First of all, we plan to consider open nets with finitely many inner markings and partners that never put more than a certain number of tokens on an interface place. Unlike the class studied in [3] the value of k is not given. Second, we would like to investigate open nets with finitely many inner markings and partners that are not restricted in the number of tokens they put on interface places.

References

1. Alves, A., Arkin, A., Askary, S., Barreto, C., Bloch, B., Curbera, F., Ford, M., Goland, Y., Guzar, A., Kartha, N., Liu, C., Khalaf, R., Knig, D., Marin, M., Mehta, V., Thatte, S., Rijn, D.v.d., Yendluri, P., Yiu, A.: Web Services Business Process Execution Language Version 2.0. Committee Specification, OASIS (2007)

2. Hinz, S., Schmidt, K., Stahl, C.: Transforming BPEL to Petri nets. In: Proc. BPM 2005. Volume 3649 of LNCS. (2005) 220–235
3. Lohmann, N., Massuthe, P., Wolf, K.: Operating Guidelines for Finite-State Services. In: Proc. ATPN 2007. Volume 4546 of LNCS. (2007) 321–341
4. Schmidt, K.: Controllability of open workflow nets. In Desel, J., Frank, U., eds.: EMISA. Volume 75 of LNI., GI (2005) 236–249
5. Hack, M.H.T.: Decidability Questions for Petri Nets. PhD thesis, MIT (1976)
6. Aalst, W.v.d.: The application of Petri nets to workflow management. *J. Circuits, Systems and Computers* **8**(1) (1998) 21–66
7. Kindler, E., Martens, A., Reisig, W.: Inter-operability of Workshop Applications - Local Criteria for Global Soundness. In: Proc. BPM 2000. Volume 1806 of LNCS. (2000) 235–253
8. Martens, A.: Analyzing web service based business processes. In: Proc. FASE. Volume 3442 of Lecture Notes in Computer Science. (2005) 19–33
9. Cassandras, C., Lafortune, S.: *Introduction to Discrete Event Systems*. Kluwer (1999)
10. Ramadge, P., Wonham, W.: Supervisory control of a class of discrete event processes. *SIAM J. Control Optim.* **25**(1) (1987) 206–230
11. Vogler, W.: *Modular Construction and Partial Order Semantics of Petri Nets*. Springer-Verlag (1992)