Petri Nets and the Equivalence Problem

Yoram Hirshfeld*
Tel Aviv University
LFCS, University of Edinburgh

Abstract. We prove that trace equivalence is undecidable for the very small and natural subclass of Petri nets in which no transition requires more than one token; this subclass corresponds exactly to the calculus BPP of Christensen for which bisimulation equivalence is known to be decidable. We present this result along with a survey of decidability results for various notions of equivalence with respect to various subclasses of Petri nets.

1 Introduction

The main aims of concurrency theory are to provide means of presenting concurrent computational systems and to provide techniques for verifying the correctness of a process so described. Correctness is typically defined by an equivalence relation: a specification and an implementation are presented, and they are checked for equivalence. Thus for example an important and well studied model for describing systems is *Petri nets*, and there have been many notions of equivalence between nets, such as *trace* equivalence, which holds between two nets which can perform the same traces or sequences of events. An important research topic in the development of concurrency theory is thus the study of the decidability of equivalence relations over classes of concurrent systems. In this paper, we present decidability results for various notions of equivalence with respect to subclasses of Petri nets, proving in particular that trace equivalence is undecidable for a small and simple yet very natural class of Petri nets.

From the point of view of concurrency theory the simplest Petri nets are those in which no cooperation between places is needed in order to fire a transition; every transition is activated by a single token, and the tokens may flow freely through the net independently of each other. We shall call such a net *a communication free net.* It is very difficult to program such a net to perform any particular computation, let alone in any sense to simulate all Turing machines. It is therefore remarkable that the ingenious method of Jančar [12] can be developed and exploited to prove our main result: that trace equivalence is undecidable for communication free nets.

For the proof of this theorem we note first that it suffices to prove that trace *inclusion* is undecidable. We take as a standard computing device Minsky's *counter machines* [16] and we associate with each counter machine M a

* Currently on sabbatical leave from Tel Aviv University.

communication free net N which *intentionally* simulates the machine: that is, there is a *proper* way to execute the net simulating the machine. (Unavoidably this net also has infinitely many ways to execute improperly.) We then modify this net to get a pair of nets N_1 and N_2 such that every trace of N_1 is a trace of N_2 if and only if M halts on input 0. This is achieved by enabling the first net N_1 to perform a new exclusive action when the *"halt"* state is reached, and enabling N_2 to reach a new universal state whenever N_1 acts improperly. The last modification neutralizes the effect of improper moves, while the former modification assures that the proper traces of N_1 are included in the traces of N_2 if and only if the corresponding Minsky machine halts. The construction is carried out carefully in order to avoid side effects caused by improper executions of N_1. This proves that trace inclusion is undecidable and the undecidability of trace equivalence follows.

This result complements the recent result [4] that *bisimulation* equivalence is decidable for communication free nets. For the class of all Petri nets trace equivalence [8, 9] and bisimulation equivalence [12] are both known to be undecidable. On the other hand for the class of unlabeled (one-to-one labeled) Petri nets both equivalences coincide and they are decidable as their decidability problem can be reduced to the decidability of the reachability relation. This picture is enriched by adding a third equivalence, that of *reachability* equivalence. In contrast to trace equivalence, this equivalence is undecidable for unlabeled nets [8, 9] but it is decidable for communication free nets [5]. In Figure 1, we summarize these results.

	trace equivalence	bisimulation equivalence	reachability equivalence
Unlabelled Petri Nets	YES	YES	NO
General Petri Nets	NO	NO	NO
Communication Free Nets	NO	YES	YES

Fig. 1. Decidability results

As for all other equivalences in the linear time – branching time spectrum of [6], in each of the three classes their decidability is the same as for trace equivalence. This is naturally so for the class of unlabeled nets and for the class of all Petri nets, and it is also so in the case of communication free nets: in [11] Hüttel extends the proof of our main theorem to show that they are all undecidable. This leaves bisimulation equivalence as unique amongst this spectrum of equivalences.

The class of communication free nets arose very naturally in the following setting. Milner and Hennessy showed in [10] that bisimulation equivalence is axiomatizable for finite state processes. The communication free nets, which are identical to the *Basic Parallel Processes* of [2], proved to be a natural extension to infinite state processes for which this equivalence is still decidable and indeed even axiomatizable [4]. Our main theorem shows that these processes are a nontrivial generalization of the finite state processes. As discussed in [2] the Basic Parallel Processes are also closely related to the Basic Process Algebra of [1] and similar decidability and undecidability results hold for the class of Basic Process Algebra [7].

The paper is organised as follows. In sections 2 and 3 we recall the definitions of Petri nets and Minsky machines, and we define the class of communication free nets. In section 4 we demonstrate how a net weakly simulates a Minsky machine. The undecidability of trace inclusion and equivalence for communication free nets is proved in section 5, and in section 6 this undecidability result is compared to other decidability results concerning Petri nets.

Acknowledgements: I am grateful for the hospitality and cooperation that I experience at the Laboratory for the Foundation of Computer Science at Edinburgh University. Faron Moller and Søren Christensen introduced me to the questions addressed here and cooperated in the research [3, 4]. Jančar's paper [12] both inspired and enabled the proof of the main theorem and I learned a lot from him in the period that he wrote the paper. The work benefitted also from discussions with other colleagues: Robin Milner, Colin Stirling, Hans Hüttel and Javier Esparza. Special thanks to Faron Moller who was helpful in many ways and who presented this paper at the conference.

2 Petri Nets and Communication Free Nets

We shall follow the definitions for Place/Transition Petri nets given in [17] with a slight change of emphasis.

A *(finite, unweighted, labeled) net* is a quintuplet $N = \langle P, T, \Sigma, F, \ell \rangle$ where P, T and Σ are finite sets (of *places*, *transitions* and *actions* respectively), $F \subseteq (P \times T) \cup (T \times P)$ (is the *flow relations*) and $\ell : T \longrightarrow \Sigma$ (is the *labeling*).

Given a transition t of a net, the *preset* ${}^\bullet t$ and the *postset* $t^\bullet$ of t are defined as

$$\begin{aligned} {}^\bullet t &\overset{\text{def}}{=} \{ p \in P \;:\; (p,t) \in F \}; \\ t^\bullet &\overset{\text{def}}{=} \{ p \in P \;:\; (t,p) \in F \}. \end{aligned}$$

The preset ${}^\bullet p$ and postset $p^\bullet$ of a place p are similarly defined.

A *marking* of the net N is a map $m : P \longrightarrow \omega$ which assigns to every place $p \in P$ a natural number $m(p)$ representing the *number of tokens on* p in marking m. A *marked net* is a pair $S = \langle N, m \rangle$ where m is a marking of the net N.

Let $S = \langle N, m \rangle$ be a marked net. A transition t is *enabled in* S (or *enabled by* m) if $m(p) > 0$ for every $p \in {}^\bullet t$. If the transition t is enabled in S then we say that S may *fire* t (thus performing the action $\ell(t)$) and evolve into $S' = \langle N, m' \rangle$, where

$$m'(p) = \begin{cases} m(p) - 1 & \textit{if } p \in {}^\bullet t \textit{ and } p \notin t^\bullet; \\ m(p) + 1 & \textit{if } p \notin {}^\bullet t \textit{ and } p \in t^\bullet; \\ m(p) & \textit{otherwise.} \end{cases}$$

In this case, we write $S \xrightarrow{\ell(t)} S'$.

If $S \xrightarrow{a_1} S_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} S_n$ then the string $a_1 a_2 \cdots a_n$ is a *trace* of S. If furthermore S_n does not enable any transition then this trace is a *completed trace* of S. The *language* $L(S)$ is the set of traces in S.

The class of simple Petri nets in which we are particularly interested is the following.

Definition: A net is a *communication free net* if $|{}^\bullet t| \leq 1$ for every transition t.

3 Minsky Machines

Minsky machines [16] represent the ultimate in unstructured programs. They have only two kinds of instructions and they use only two counters. Every instruction is some form of a labeled **goto** instruction.

Formally, a *Minsky machine* is a sequence of labeled instructions

$$\begin{aligned} \mathtt{L}_0 &: \mathtt{comm}_0 \\ \mathtt{L}_1 &: \mathtt{comm}_1 \\ &\cdots \\ \mathtt{L}_{n-1} &: \mathtt{comm}_{n-1} \\ \mathtt{HALT} &: \mathtt{halt} \end{aligned}$$

where each of the first n instructions is either of the form

```
L : c:=c+1; goto L′
```

or of the form

```
L : if c=0 then goto L′
    else c:=c-1; goto L″
```

In the above, c is one or other of two counters $\mathtt{c}_0$ or $\mathtt{c}_1$.

A Minsky machine M starts executing with the value $x \in \omega$ in counter $\mathtt{c}_0$, 0 in counter $\mathtt{c}_1$, and the control at the label $\mathtt{L}_0$. When the control is at label $\mathtt{L}_k$ ($0 \leq k < n$), the machine executes instruction $\mathtt{comm}_k$, modifying the contents of the counters and transferring the control to the appropriate label mentioned in the instruction. The machine halts if and when the control reaches the label HALT. When this happens the number stored in $\mathtt{c}_0$ is the result of the computation for x and it is denoted by $M(x)$. Every partial computable function f is computed by some Minsky machine M_f in the sense that M_f halts for x if and only if $f(x)$ is defined, and if it halts then $M_f(x) = f(x)$. Since the halting problem on input 0 is undecidable we have the following.

Basic Undecidability Result: There is no algorithm to decide whether a given Minsky machine will halt when started with the value 0 in counters $\mathtt{c}_0$ and $\mathtt{c}_1$.

4 The Net Simulation of a Minsky Machine

We associate with every Minsky machine M a marked communication free net $S_M = \langle N_M, m_M \rangle$ which simulates M in a weak sense. This is a modification of Jančar's construction in [12] which in turn follows the construction in [8, 9]. The net N has two *counter places* which will be denoted by $\mathtt{c}_0$ and $\mathtt{c}_1$, as well as a *control place* L for each machine label L.

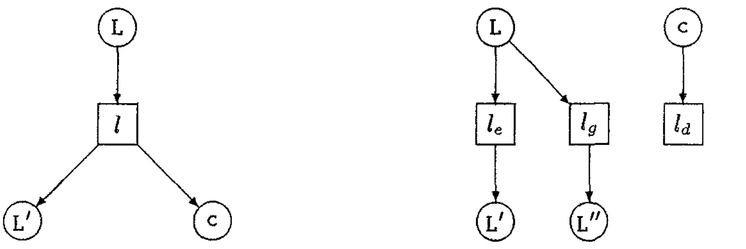
For every machine instruction of the form

```
L : c:=c+1; goto L'
```

there is a transition t labelled l with ${}^\bullet t = \{\mathtt{L}\}$ and $t^\bullet = \{\mathtt{L}', \mathtt{c}\}$. For every machine instruction of the form

```
L : if c=0 then goto L'
    else c:=c-1; goto L''
```

there are three transitions t_1, t_2 and t_3, labelled ℓ_e (*equal* 0), ℓ_g (*greater than* 0) and ℓ_d (*discard*) respectively, with ${}^\bullet t_1 = {}^\bullet t_2 = \{\mathtt{L}\}$ and ${}^\bullet t_3 = \{\mathtt{c}\}$, and with $t_1^\bullet = \{\mathtt{L}'\}$, $t_2^\bullet = \{\mathtt{L}''\}$ and $t_3^\bullet = \{\}$. The corresponding fragments of the net for these two instruction types are depicted graphically in Figure 2.



```
L : c:=c+1; goto L'
```

```
L : if c=0 then goto L'
    else c:=c-1; goto L''
```

Fig. 2. Petri net fragments for Minsky machine instructions

The initial marking has one token on the control place $\mathtt{L}_0$ and none elsewhere. It is easy to see that every stage of an execution of S_M has exactly one control place marked and that executions which observe the following restrictions simulate M (in a way that will be formalized):

- Fire the l_e transition only when the relevant counter place c is empty.
- Always fire the relevant l_d transition after firing the l_g transition.

Formally we have the following definition and proposition:
Definition: An execution $S_M \xrightarrow{a_0} S_1 \xrightarrow{a_1} \cdots \xrightarrow{a_n} S_n$ is a *proper run* if it satisfies:

1. If a_i is l_e then the appropriate counter place is empty;
2. If a_i is l_g (for $i < n$) then a_{i+1} is l_d corresponding to the relevant counter discard, and if a_i is l_d then $i > 0$ and a_{i-1} is l_g.

Proposition: Let M be a Minsky machine and let $S_M = \langle N_M, m_M \rangle$ be the corresponding marked net.

- If M halts on input 0 then S_M has a unique completed proper run. It reaches a marking m for which $m(\mathtt{HALT}) = 1$ and $m(\mathtt{c}_0) = M(0)$.
- If M does not halt on input 0 then all the proper runs of S_M are prefixes of a single infinite proper run which never reaches the control state HALT

Note that besides the unique maximal proper run there may be infinitely many improper runs.

5 The Main Theorem

Theorem: Language inclusion and equivalence are undecidable for communication free nets.
Proof: Due to a simple observation which was pointed out by Hans Hüttel it suffices to prove that language inclusion is undecidable. We shall show this for nets initially marked by a single token since we need only such nets for the proof of the theorem and their treatment is somewhat simpler. For every pair $S_1 = \langle N_1, m_1 \rangle$ and $S_2 = \langle N_2, m_2 \rangle$ of communication free nets let $S_1 + S_2$ be the net $\langle N, m \rangle$ such that N is the disjoint union of N_1 and N_2 together with a new place p_0 (a similar construction works for general Petri nets but its description is simpler for communication free nets). For every transition t which is enabled by a place originally marked in any of the two nets we add a new transition labeled with the same action and having $\{p_0\}$ as preset and exactly the same postset as t. The marking m has a unique token in p_0 (and none elsewhere). It is easy to see that we have the language equation $L(S_1 + S_2) = L(S_1) \cup L(S_2)$. In particular

$$L(S_1) \subseteq L(S_2) \textit{ if and only if } L(S_1 + S_2) = L(S_2).$$

Therefore it suffices to show that language inclusion is undecidable. We shall do so by associating with every Minsky machine M a pair of communication free nets S_1 and S_2 for which

$$L(S_1) \subseteq L(S_2) \textit{ if and only if } M \text{ does not halt (on input 0).}$$

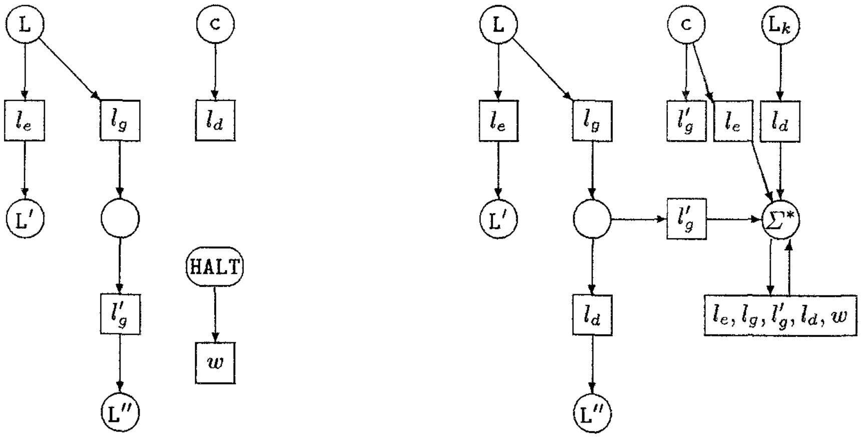
Given this, language inclusion and language equivalence are undecidable by the basic undecidability result above.

S_1 and S_2 will be obtained by modifying S_M to achieve the following:

- S_1 will have a new exclusive transition w that is enabled by the final control place HALT. Therefore $L(S_1) \subseteq L(S_2)$ only if M does not halt.

- S_2 will have a new universal place Σ^* which once reached can execute any string of actions.
- S_2 can reach the universal place as soon as S_1 performs an improper move but not before. This will neutralize the effect of improper moves of S_1 without enabling S_2 to improperly benefit from his new found powers.

The most important ingredient in the modification is the break of each l_g transition into two halves, using the second half to force the accompanying single discard required. In modifying S_1, for every transition labeled l_g we add a new unlabeled place and a new transition labeled l'_g to be wired as in Figure 3*(a)*.



(a) Net fragment for S_1. *(b)* Net fragment for S_2.

Fig. 3. Petri net fragments for instructions

To obtain S_2 from S_M is a little more complicated. Again we add for every l_g transition a new unlabeled place and a new transition but we label the new transition with the action l_d and we label the previous transition l_d with the label l'_g.

Next we add a universal place Σ^* and for every action σ we add a transition from Σ^* to itself labeled σ. Thus an execution that reaches Σ^* can be arbitrarily continued as long as we wish. For each form of misbehaviour of S_1 we add transitions leading into Σ^* to prevent S_1 from benefitting from the misconduct:

- For every l_e transition add a transition t labeled by l_e with the corresponding counter c as its preset and Σ^* as its postset. S_1 can gain nothing now by performing l_e when the corresponding counter is not empty and yet S_2 cannot reach Σ^* by a transition sequence on S_1 which avoids such misbehaviour.
- For every transition l_g we add a new transition labeled l'_g and having the corresponding unlabeled place as its preset and having Σ^* as its postset.

This will assure that S_1 will not benefit from forgetting to discard after l_g and continuing with the control transfer l'_g.

- Finally for every discard l_d and for every control place $\mathtt{L}_k$ we add a transition labeled by l_d with L as its preset and Σ^* as its postset. This will prevent S_1 from gaining by discarding at any stage except as the immediate continuation to an l_g transition. Because of the dual wiring of l'_g and l_d in the two nets, S_1 is also prevented from performing two discards l_d in a row, since by the time that the first one is completed the control in S_2 is already in L'', which has the power to retaliate.

This yields the wiring in Figure 3*(b)* and the proof is now a mere formality:

- Define a proper run of S_1 to be an execution sequence that has l_e only from a marking where the corresponding counter place is empty and that has occurrences of l_g, l'_g and l_d only as part of the succession $l_g l_d l'_g$. Then M halts if and only if S_1 has a proper run that reaches the control place $\mathtt{HALT}$.
- If a string of actions α is a proper run of S_1 then there is a unique run of α on S_2. If α reaches the marking m in S_1 then it reaches the matching marking m' in S_2. Here the matching marking to m is m itself if the last action of α differs from a discard l_d, and if it is a discard then m' has one token more on the corresponding counter and the control token on the corresponding L'' (while m has it still on the unlabeled place).
- If M halts then $L(S_1) \not\subseteq L(S_2)$ as the proper run α of S_1 that reaches the control place $\mathtt{HALT}$ does not reach Σ^* in S_2 and hence αw is in $L(S_1)$ but not $L(S_2)$.
- Finally if M does not halt then $L(S_1) \subseteq L(S_2)$. Let β be a run of S_1 and let α be its maximal proper prefix. If $\alpha = \beta$ then $\beta \in L(S_2)$. Otherwise α reaches matching markings in the two nets and the next action is improper. But as discussed above every improper move of S_1 from a matching pair of markings can be matched by the same action reaching Σ^* in S_2. Therefore S_2 can also complete the run β.

6 Related Decidability Results

We recall two other important equivalences in Petri nets and we shall summarize decidability results concerning three classes of Petri nets and ranging from Hack's classical work [8, 9] to some yet unpublished results by Esparza.

- A binary relation R between marked nets is a *bisimulation* if whenever $V\mathrm{R}W$ (or $W\mathrm{R}V$) and $V \xrightarrow{a} V'$ then also $W \xrightarrow{a} W'$ for some W' satisfying $V'\mathrm{R}W'$ (respectively $W'\mathrm{R}V'$). Two marked nets are *bisimulation equivalent* [15] if there is a bisimulation relating them.
- Two marked nets $S_1 = \langle N, m_1 \rangle$ and $S_2 = \langle N, m_2 \rangle$ with the same underlying net are *reachability equivalent* if every marking reachable in one of the nets is also reachable in the other (possibly by a different run).

Bisimulation equivalence is finer than language equivalence and is probably the most discussed equivalence in concurrency theory. Reachability equivalence on the other hand is one of the most discussed in Petri nets. An important way in which these equivalences differ from each other is in how they react with the labeling of a net:

- Unlabeled nets which are language equivalent are bisimulation equivalent.
- All labeled versions of a fixed unlabeled net are reachability equivalent.

We shall discuss the three equivalences in the class of all Petri nets and in the two (incomparable) subclasses that of unlabeled marked nets and that of communication free (marked) nets. Briefly we have the following, which is summarized in Figure 1.

- Bisimulation equivalence, language equivalence and reachability equivalence are all undecidable in the class of all Petri nets.
- Bisimulation equivalence and language equivalence are decidable for unlabeled Petri nets while reachability equivalence is undecidable.
- Bisimulation equivalence and reachability equivalence are decidable for communication free nets while language equivalence is undecidable.

For the class of all Petri nets the undecidability of reachability equivalence and language equivalence was proved in [8, 9] using a notion of weak computation of polynomials by nets and the undecidability for Diophantine equations ("Hilbert's tenth problem"). Jančar [12] proved the undecidability of bisimulation equivalence and gave simpler and more informative proofs to the previous results. His proof is the basis of our proof of our main theorem, but it also had to overcome one difficulty that we did not face in our proof. Since we needed to deal only with language inclusion there was no harm in letting the simulating net S_2 reach an all powerful state in retaliation to an improper transition by S_1. Dealing with bisimulation Jančar had to consider a symmetric situation and to make the punishment fit the crime; the reaction to an improper move (using our terminology) should make the responder bisimilar to the offender and no stronger. In fact Jančar's construction enables the responder to become isomorphic to the offender. Of course this construction relies heavily on the cooperation between places and it cannot be done for communication free nets.

For the class of unlabeled Petri nets the decidability of language equivalence was shown in [8, 9] to be equivalent to the decidability of the reachability problem (given a marked net and a new marking, to decide if the new marking is reachable in the net). Since reachability was later proved to be decidable [13, 14] it follows that language equivalence is decidable in this class. The decidability of bisimulation equivalence follows from this by remark 1 above. As for reachability equivalence, since it is undecidable for general Petri nets, it follows from remark 2 above that it is also undecidable for unlabeled nets.

For the class of communication free nets, our main theorem shows that language equivalence is undecidable. The decidability of bisimulation equivalence was proved in [4] using a tableau method which also provided an axiomatisation. The decidability of reachability equivalence was proved recently by Esparza

[5]. He uses the vector representation of markings, and for communication free nets he characterizes the reachable markings by a property that is expressible in Presburger Arithmetic. Therefore reachability equivalence is also expressible and is decidable together with all of Presburger Arithmetic.

Intermediate equivalences between language equivalence and bisimulation equivalence are discussed in [6]. For unlabeled nets they all coincide (and are decidable). For general Petri Nets they are undecidable together with the two extreme equivalences. For communication free nets where bisimulation equivalence is decidable and trace equivalence is not, Hans Hüttel [11] proved that all the equivalences other than the bisimulation equivalence are also undecidable.

References

1. J.C.M. Baeten, J.A. Bergstra and J.W. Klop. Decidability of bisimulation equivalence for processes generating context free languages. In Proceedings of PARLE 87, LNCS 259, pp93-114, Springer Verlag 1987.
2. S. Christensen. Decidability and Decomposition in Process Algebra. Ph.D Thesis, Edinburgh University, 1993.
3. S. Christensen, Y. Hirshfeld, F. Moller. Decomposability, decidability and axiomatisability for bisimulation equivalence on basic parallel processes. In Proceedings of LICS93, IEEE Computer Society Press.
4. S. Christensen, Y. Hirshfeld, F. Moller. Bisimulation equivalence is decidable for basic parallel processes. Proceedings of CONCUR93, 1993.
5. J. Esparza. A note on BPP processes. Unpublished manuscript, University of Edinburgh, 1993.
6. R. van Glabbeek. The linear time – branching time spectrum. In Proceedings of CONCUR90. LNCS 458, pp278-297, 1990.
7. J.F. Groote, H. Hüttel. Undecidable equivalences for basic process algebra. Information and Computation, 1993.
8. M. Hack. Decision problems for Petri nets and vector addition systems. MAC Technical Memo 53, MIT, 1975.
9. M. Hack. The equality problem for vector addition systems is undecidable. TCS 2, pp77-95, 1976.
10. M. Hennessy,, R. Milner. Algebraic laws for nondeterminism and concurrency. JACM, Vol 32, No 1, 1985.
11. H. Hüttel. Undecidable equivalences for basic parallel processes. University of Edinburgh, 1993.
12. P. Jančar. Decidability questions for bisimilarity of Petri nets and some related problems. Proceedings of STACS94, Caen. LNCS 775, Springer Verlag pp581-592, 1994.
13. S.R. Kosaraju. Decidability of reachability in vector addition systems. In Proceedings of the 6th Annual Symposium on the Theory of Computing, pp267-281, 1982.
14. E.W. Mayr. An algorithm for the general Petri net reachability problem. SIAM Journal of Computing 13, pp441-460, 1984.
15. R. Milner. Communication and Concurrence. Prentice Hall 1989.
16. M. Minsky. Computation: Finite and Infinite Machines. Prentice Hall, 1967.
17. W. Reisig. Petri Nets. Springer Verlag, 1982.