# Concurrency Theory Seminar Paper: Undecidability of bisimilarity for Petri Nets and some related problems

Tobias Salzmann
Saarland University

January 30, 2012

## 1 Disclaimer

This seminar paper for the seminar "Concurrency Theory" is strongly based on the paper "Undecidability of bisimilarity for Petri nets and some related problems" by Petr Jančar from 1995. Most of the content (excluding some minor explanations and examples) belongs to Petr Jančar or it's respective owners.
This paper is basically a reformulation of parts of the original paper, allthough there are some one-to-one citations, for the sake of readability.

## 2 Introduction

Petri nets are a common model for concurrent systems. They allow a very natural and intuitive description of components and the way they interact with each other. While the rules to describe them are relatively simple, evaluating those can quickly lead to a (finitely branching) transition system with a infinite state space and a complicated structure.
Comparing transition systems in terms of their behaviour is an important task in the field of model checking and automatic verification. One of the most important equivalences in this context is bisimulation equivalence or bisimilarity, the ability for two systems to simulate each other step by step. Finding bisimilar states in a transition system may make difficult tasks easier, which is crucial when dealing with infinite state spaces.
It seems to be a natural question to ask whether bisimilarity on Petri nets is decidable. This (as well as the equivalent problem of decidability of language equivalence) has to be answered with no, for which we provide a proof in this paper. The proof itself is a reduction from the halting problem on counter machines introduced by Minsky [Minsky]. The first thing we will try is to construct a net which simulates a counter machine in a natural way which would make the proof almost trivial but isn't possible, because the branching commands in the counter machine can not be encoded properly using only ordinary Petri nets. We solve this by adding certain structures to the net to compensate this.
For the actual reduction, we use the so called bisimulation game, a mechanic

that allows us to establish bisimilarity or non-bisimilarity by giving perfect winning strategies for one of the two involved players.
The second main result of the paper is another reduction from the halting problem with a similar proof technique. We prove the undecidability of the reachability set containment problem by constructing nets for a given counter machine. This time we will add a subnet which encodes the so far taken branches in the original counter machine, enforcing language (non)-equivalence according to the halting-behaviour of the counter machine.
In the last part of the paper, we will have a quick look at two decidable subclasses of (pairs of) Petri nets. We will also see a construction which encodes the bisimulation game on two given nets in a so called game net, making the rules simpler as before.

# 3 Definitions and basic propositions

**Definition** $\mathbb{N} = \mathbb{N}_0$ is the set of nonnegative integers.

**Definition** $A^* := \bigcup_{k=0}^{\infty} A^k$ is the set of finite sequences of elements in $A$.

## 3.1 Nets

**Definition** A *net* is a tuple $\Sigma = (P, T, F)$ and a *labelled* net is a tuple $\Sigma = (P, T, F, L)$ where :

- $P$ is a finite set of *places*.
- $T$ is a finite set of *transitions* disjoint to $P$.
- $F : (P \times T) \cup (T \times P) \to \mathbb{N}_0$ is a *flow function*.
  If $F(x, y) = m > 0$ we say there's an *arc* with multiplicity $m$ between $x$ and $y$.
- $L : T \to A$ is a *labelling*, where $A$ is a countable set of *actions*.
  We will sometimes use L on finite sequences of actions meaning the function $L^* : T^* \to A^*, L^*(t_1 \dots t_n) := L(t_1) \dots L(t_n)$

**Definition** A *marking* of a labelled net $\Sigma$ = (P,T,F,L) is a function $M : P \to \mathbb{N}_0$.
If $M(p) = k$ we speak of $k$ *tokens* being on place $p$.
The set of markings $\mathscr{M}(\Sigma)$ is isomorphic to $\mathbb{N}_0^{|P|}$.

**Definition** A *(labelled) Petri net* is a pair $(\Sigma, M_0)$ where $\Sigma$ = (P,T,F) ($\Sigma$ = (P,T,F,L)) is a (labelled) net and $M_0$ is an *initial marking*.
A transition $t \in T$ is *enabled* at a marking $M$, denoted as $M \xrightarrow{t}_\Sigma$, if for every place $p \in P$ the following holds:

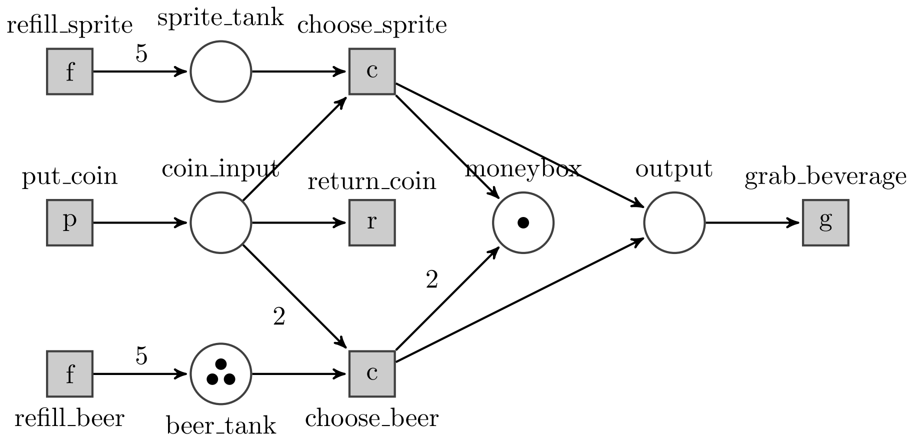
$$M(p) \geq F(p, t)$$

A transition $t \in T$ enabled at $M$ may *fire*, yielding a marking $M'$ given by:

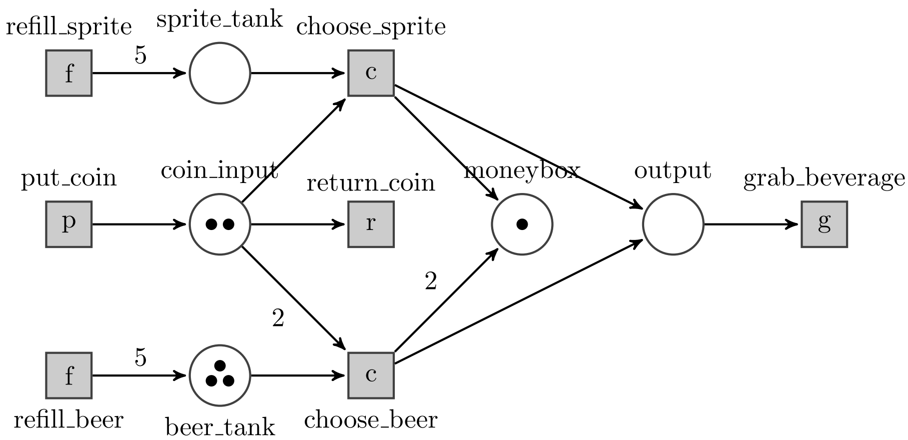
$$M'(p) = M(p) - F(p, t) + F(t, p)$$

We denote this as $M \xrightarrow{t}_{\Sigma} M'$.
For $a \in A$, $M \xrightarrow{a}_{\Sigma}$ means that there is a $t \in T$ with $L(t) = a$ such that $M \xrightarrow{t}_{\Sigma}$ and $M \xrightarrow{a}_{\Sigma} M'$ means that there is a $t \in T$ with $L(t) = a$ such that $M \xrightarrow{t}_{\Sigma} M'$.
For $\sigma = \sigma_1 \dots \sigma_n \in T^* \cup A^*, M \xrightarrow{\sigma}_{\Sigma} M'$ means that there are $M_0, \dots, M_n$ such that $M = M_0, M_n = M'$, and $\forall i \in \{1, \dots, n\}, M_{i-1} \xrightarrow{\sigma_i}_{\Sigma} M_i$.

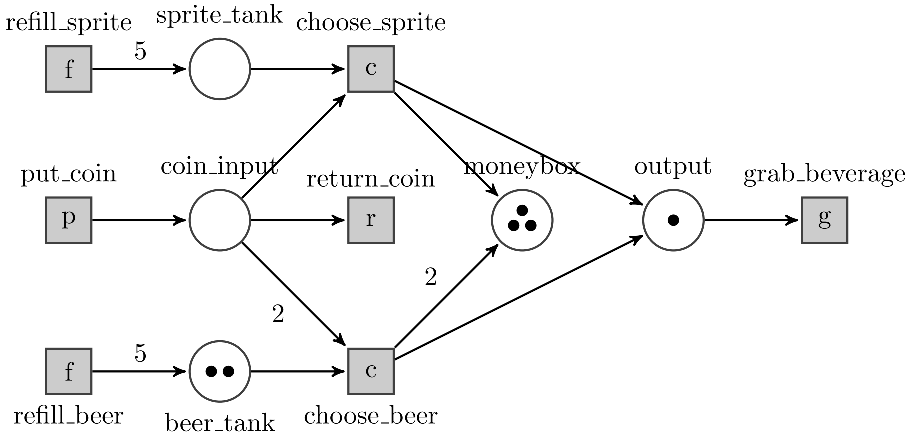
This is an example of a labelled Petri net modeling a refillable beverage vending machine selling beer for 2 and sprite for 1 coins. Places are drawn as circles, transitions as squares with their labelling inside, arcs as arrows labelled with their multiplicity (omitted if =1).
The initial marking is represented by the dots=tokens in the places. At the beginning, the transitions refill_sprite, refill_beer and put_coin are enabled.




The next image shows the Petri net after firing put_coin twice. Now, choose_beer is also enabled.




After firing choose_beer, it is in the following state:




**Definition** For a Petri net $N = (\Sigma, M_0)$, the *reachability set* of $N$ is defined as $\mathscr{R}(N) = \{M | M_0 \xrightarrow{\sigma}_\Sigma M, \sigma \in T^*\}$.

**Definition** For a Petri net $N = ((P, T, F, L), M_0)$, a place $p \in P$ is *unbound* if for every $n \in \mathbb{N}$ there is a marking $M \in \mathscr{R}(N)$ such that $M(p) > n$.

**Definition** For a labelled Petri net $N = (\Sigma, M_0)$, the *language* or set of *traces* of $N$ is defined as $\mathscr{L}(N) = \{w \in A^* | M_0 \xrightarrow{w}_\Sigma M, M \in \mathscr{M}(\Sigma)\}$.
Two labelled Petri nets $N_1, N_2$ are *language equivalent* if $\mathscr{L}(N_1) = \mathscr{L}(N_2)$.

## 3.2 Bisimulations and Bisimilarity

**Definition** For two labelled nets $\Sigma_1 = (P_1, T_1, F_1, L_1)$, $\Sigma_2 = (P_2, T_2, F_2, L_2)$ a relation $R \subset \mathscr{M}(\Sigma_1) \times \mathscr{M}(\Sigma_2)$ is a *bisimulation*, if for every $(M_1, M_2) \in R$ and every $a \in A$ the following conditions hold:

1. For every $M_1' \in \mathscr{M}(\Sigma_1)$ s.t. $M_1 \xrightarrow{a}_\Sigma M_1'$
   there is $M_2' \in \mathscr{M}(\Sigma_2)$ s.t. $M_2 \xrightarrow{a}_\Sigma M_2'$ and $(M_1', M_2') \in R$
2. For every $M_2' \in \mathscr{M}(\Sigma_2)$ s.t. $M_2 \xrightarrow{a}_\Sigma M_2'$
   there is $M_1' \in \mathscr{M}(\Sigma_1)$ s.t. $M_1 \xrightarrow{a}_\Sigma M_1'$ and $(M_1', M_2') \in R$

**Definition** For two labelled nets $\Sigma_1 = (P_1, T_1, F_1, L_1)$, $\Sigma_2 = (P_2, T_2, F_2, L_2)$, two markings $M_1 \in \mathscr{M}(\Sigma_1)$, $M_2 \in \mathscr{M}(\Sigma_2)$ are *bisimilar* or *bisimulation equivalent*, denoted as $M_1 \simeq M_2$, if there is a bisimulation containing $(M_1, M_2)$.

Two labelled Petri nets $N_1 = (\Sigma_1, M_{0,1})$, $N_2 = (\Sigma_2, M_{0,2})$ are *bisimilar* or *bisimulation equivalent*, denoted as $N_1 \simeq N_2$, if $M_{0,1} \simeq M_{0,2}$.

**Definition** A *bisimulation game* consists of two *players*, which we call *attacker* and *defender* and two labelled Petri nets $N_1$ and $N_2$. Each *round* is divided in two *turns*:

1. The attacker chooses one of the nets and fires an enabled transition $t$ , changing the marking appropriately.
2. The defender fires a transition $t'$ in the other net with the same label as $t$, changing the marking too.

This is repeated until one of the players isn't able to fire a transition anymore.
If the defender eventually isn't able to defend an attack, the attacker wins the game.
If the attacker eventually isn't able to attack anymore or the game goes on infinitely, the defender wins the game.

**Proposition 3.1** *$N_1, N_2$ are bisimilar if and only if the defender has a* defending strategy *in a bisimulation game.*

***Proof*** *Let's assume $N_1$ and $N_2$ are in related states of a bisimulation $R$ prior to a round of a bisimulation game. If the attacker fires a transition $t$ in one net, yielding a state $M$, the defender is, according to the definition of bisimulation relations, able to fire a equally labelled transition $t'$ in the other net, yielding a marking $M'$, such that $M\ R\ M'$.*
*If $N_1, N_2$ are bisimilar, there is such a relation $R$ containing their initial markings, hence the defender is always able to respond to the attacker's turn using the pairs in $R$ as a strategy.*
*The other way round, if the defender has a defending strategy, the union of all pairs of states occuring in successfully defended games (using this strategy) yield a bisimulation between $N_1$ and $N_2$.* □

**Proposition 3.2** *If $N_1, N_2$ are bisimilar then they are language equivalent.*

***Proof*** *Considering a bisimulation game on two bisimilar Petri nets where the attacker fires a sequence of actions (word) in one net, the defender is always able to fire the same sequence (word) in the other net.* □

### 3.3 Counter Machines, Decidability and the Halting Problem

**Definition** A *counter machine* $C$ with nonnegative *counters* $c_1, \ldots, c_m$ is a program of the form:

$$1 : COMM_1; 2 : COMM_2; \ldots; n : COMM_n$$

where $COMM_n$ is a $HALT$-command and $COMM_1, \ldots, COMM_{n-1}$ are of one of the two following types:

1. $c_j + +$; goto $k$
2. if $c_j = 0$ then goto $k_1$ (else $c_j - -$; goto $k_2$)

where $k, k_1, k_2 \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$.
The set of *branching states* is defined as $BS := \{i | COMM_i$ is of type $2\}$.
Note that for $m \geq 2$ , counter machines are equally powerful as turing machines.
(They can simulate each other by using fancy constructions and encodings)

**Definition** A decision problem $P$ is *decidable* if there is a counter machine (turing machine, algorithm) $M$ such that given an instance $I$ of $P$ as an input, $M$ is guaranteed to halt and *decide* whether or not $P$ is true for $I$.

**Definition** The *halting problem* on counter machines is described by:
Instance: a counter machine $C$ with m counters and $n$ commands , an Input $I$ .
Question: Does $C$ eventually reach state $n$ and terminate?

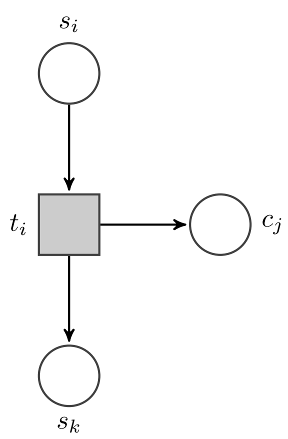
**Proposition 3.3** *The halting problem on counter machines is undecidable.*

***Proof*** *(sketch)*
*Since counter machines are as powerful as turing machines, there is a universal counter machine $U$ which, given any counter machine $M$ and input $I$ (encoded in a suitable way), can simulate the behaviour of $M$ on $I$.*
*We now modify $U$ to $U'$ by forcing it into a diverging trap state if the simulated machine leaves value 1 on its first counter, otherwise to halt.*
*Assume there is a counter machine $M_{U'}$ which decides the halting problem for $U'$ and a given input $I$, w.l.o.g leaving value 1 on a counter iff its instance evaluates to true.*
*If we now run $U'$ on $M_{U'}$ and any input $I$, by construction $U'$ halts iff $U'$ does not halt, which is a contradiction.*
*So the halting problem for counter machines is undecidable.* □

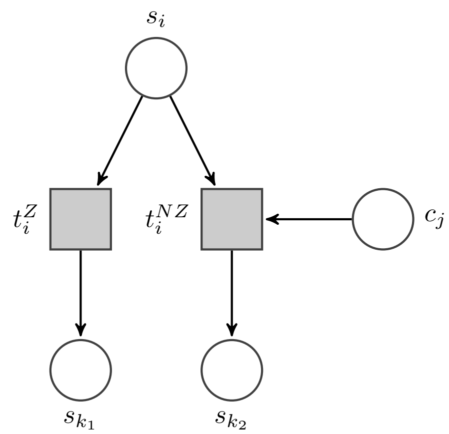
More on counter machines can be found in [Minsky].

**Definition** The basic net $\Sigma_C = (P, T, F)$ for a counter machine $C$ with $m$ counters and $n$ commands is constructed in the following way:

1. Set $P := \{s_1, s_2, \ldots, s_n, c_1, c_2, \ldots, c_m\}$
2. For every $i$ such that $COMM_i = c_j ++$; goto $k$:
   Add transition $t_i$ to $T$ as well as arcs $(s_i, t_i), (t_i, s_k), (t_i, c_j)$ to $F$.

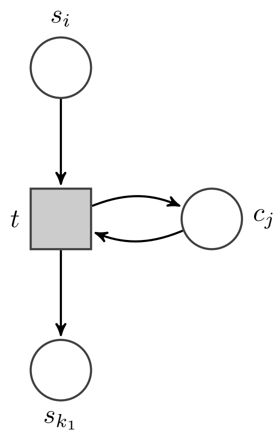


3. For every $i$ such that $COMM_i =$ if $c_j = 0$ then goto $k_1$ (else $c_j - -$; goto $k_2$):
   Add transitions $t_i^Z$ and $t_i^{NZ}$ to $T$ as well as arcs $(s_i, t_i^Z), (t_i^Z, s_{k_1})$ for the $c_j = 0$ branch and $(s_i, t_i^{NZ}), (t_i^{NZ}, s_{k_2}), (c_j, t_i^{NZ})$ for the non-zero alternative.




If we now, given input values $x_1, \ldots, x_m$, place one token in $s_1$ and $x_1, \ldots, x_m$ tokens in $c_1, \ldots, c_m$, the resulting Petri net seems to be a pretty good modeling for $C$, because in every reachable state, there will be exactly one token among $s_1, \ldots, s_m$ representing the program counter of C.
The only major problem with the construction is that for i $\in BS$ the $t_i^Z$ transitions may fire independently of the state of the corresponding counter, while $COMM_i$ requires it to be 0. Due to the fact that we may specify lower, but not upper bounds on the number of tokens on places as requirements to fire transitions, this can't be fixed using (this type of) Petri nets. In order to prove our desired main results, we will use a trick called dc-transitions:

**Definition** Adding a *dc-transition* (dc meaning "definitely cheating") to a net $\Sigma_C$ for $i \in BS$ and $COMM_i =$ if $c_j = 0$ then goto $k_1$ (else $c_j - -$; goto $k_2$) means adding a transition $t$ and arcs $(s_i, t), (t, s_{k_1}), (t, c_j), (c_j, t)$.



# 4 Undecidability of Bisimilarity on Petri Nets

**Theorem 4.1** *For a counter machine $C$ with $m$ counters and $n$ commands and input values $x_1 \dots x_m$, there are labelled Petri nets $N_1, N_2$ such that the following conditions are equivalent:*

*(a) $C$ does not halt on $x_1 \dots x_m$*

*(b) $N_1, N_2$ are bisimilar*

*(c) $\mathscr{L}(N_1) = \mathscr{L}(N_2)$*

*(d) $\mathscr{L}(N_1) \subseteq \mathscr{L}(N_2)$*

***Proof*** *$N_1, N_2$ are constructed as follows:*

1. *Start with $\Sigma_C$.*
2. *Add places $p, p'$.*
3. *For each $i \in BS$, add two dc-transitions $t'_i, t''_i$ and arcs $(p, t'_i), (t'_i, p'), (p', t''_i), (t''_i, p)$.*
4. *Add transition $t_F$ and arcs $(s_n, t_F), (p, t_F)$.*
5. *Choose $L$ in a way such that $L(t_F) \neq L(t_i)$ for each $i \in \{1, ..., n-1\}$ and $L(t_i^Z) = L(t'_i) = L(t''_i)$ for each $i \in BS$.*
6. *Put 1 token in $s_1$ and $x_1 \dots x_m$ tokens in $c_1 \dots c_m$.*
7. *To get $N_1$, put 1 token in $p$.*
   *To get $N_2$, put 1 token in $p'$.*

*Since we proved ((b) $\Rightarrow$ (c)) earlier and ((c) $\Rightarrow$ (d)) is trivial, we will show ((a) $\Rightarrow$ (b)) and ((d) $\Rightarrow$ (a)), using the earlier mentioned in the bisimulation game on $N_1, N_2$ in order to prove the theorem:*

*((d) $\Rightarrow$ (a)) :*
*In order to show this, we will prove the equivalent implication ($\neg$(a) $\Rightarrow \neg$(d)) by providing a winning strategy for the attacker, if $C$ halts on $x_1 \dots x_m$ .*
*The strategy consists of firing the "legal" sequence $\sigma = \sigma_{i_1}\sigma_{i_2} \dots \sigma_{i_q} t_F$ in $N_1$ corresponding to the execution of $C$ on $x_1 \dots x_m$, which leaves the defender no choices. He has to fire the same sequence of transitions in $N_2$, especially he will never be able to fire a dc-transition. As a result, the attacker will eventually not be able to fire $t_F$, lacking a token in $p$.*
*So, $L(\sigma) \in \mathscr{L}(N_1)$ but $L(\sigma) \notin \mathscr{L}(N_2)$.*

*((a) $\Rightarrow$ (b)) :*
*To prove this, we will give a defending strategy for the defender, if $C$ does not halt on $x_1 \dots x_m$:*

- *If the attacker makes a "legal" move, there is no choice and the defender has to make the same move.*

- *If the attacker makes an "illegal" move, meaning there is a token on $s_i$, $COMM_i =$ if $c_j = 0$ then goto $k_1$ (else $c_j - -$; goto $k_2$), there is at least one token on $c_j$ and the attacker fires $t_i^Z$, $t_i'$ or $t_i'$ in one of the nets.*
  *If the attacker has not cheated before, there are four cases:*
  - *$t_i^Z$ in $N_1$: respond by taking $t_i''$ in $N_2$*
  - *$t_i^Z$ in $N_2$: respond by taking $t_i'$ in $N_1$*
  - *$t_i'$ in $N_1$: respond by taking $t_i^Z$ in $N_2$*
  - *$t_i''$ in $N_2$: respond by taking $t_i^Z$ in $N_1$*

  *If the attacker cheated before, respond by taking the same transition in the other net.*

*As long as the attacking player only makes legal moves, the defender is able to follow the same sequence of transitions, since the "critical" transition $t_F$ will not be reached. So the attacker has to cheat at least once. As the defender responds according to the strategy, the resulting marking of both nets will be identical after the first round where the attacker cheated. From now on, the defender may simply copy the moves of the attacker to win.* □

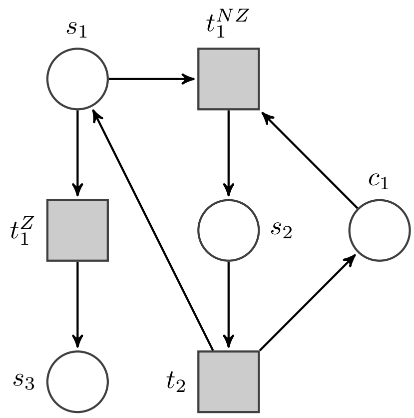
## 4.1 Example

The counter machine with 3 commands and 1 counter $C :=$

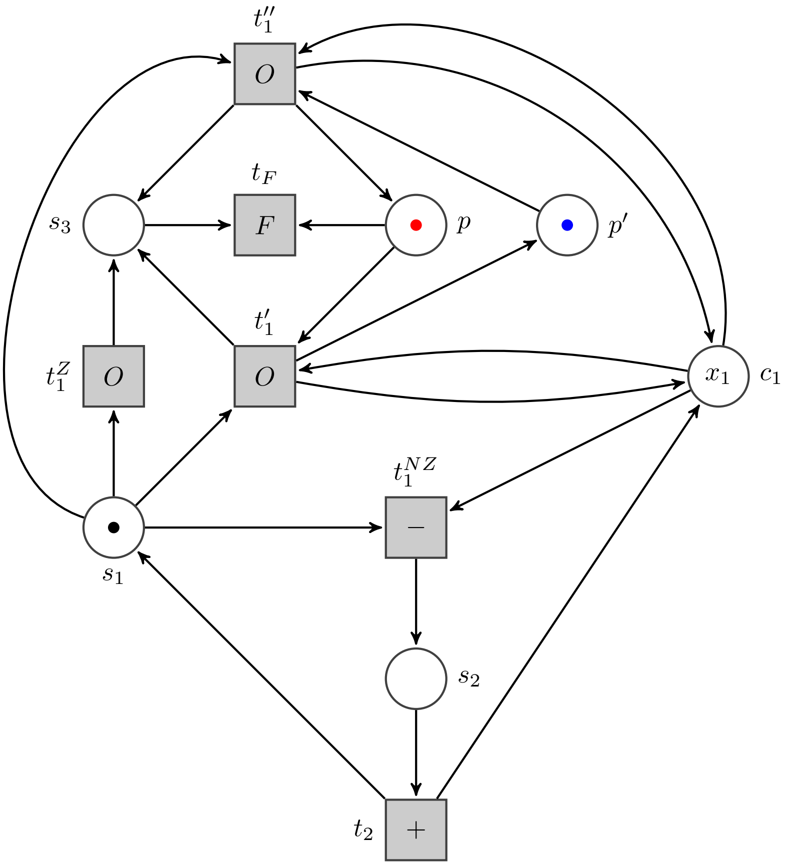
$1:$ if $c_1 = 0$ then goto 3 else ($c_1 - -$; goto 2);
$2:$ $c_1 + +$; goto 1;
$3:$ HALT;

only halts for the initial input $x_1 = 0$.

Consider $A = \{+, -, O, F\}$.
It's basic net $\Sigma_C$ for $C$ is constructed as follows:




If we now add places $p, p'$, transitions $t_1', t_1'', t_F$ and the arcs according to the construction, we get the following Petri nets:




(The red token is only placed on $N_1$, the blue only on $N_2$, the others on both nets)
Now let's have a look at two different values for $x_1$:

$x_1 = 0$:
The described winning strategy for the attacker is:
Fire $t_1^Z$ in $N_1$. The defender has to respond by firing $t_1^Z$
Fire $t_F$ $N_1$. The defender has no transitions to fire, the attacker wins.
$\Longrightarrow$ The Petri nets are not bisimilar and the word $OF$ is in the language of $N_1$, but not in the language of $N_2$

$x_1 = 1$:
The described defending strategy is:
As long as the attacker fires $t_1^{NZ}$ or $t_2$, do the same.
As soon as he fires $t_1^Z, t_1', t_1''$ the first time, react by firing the transition which equalizes the markings.
Copy every move of the attacker from now on, the defender wins.
$\Longrightarrow$ The Petri nets are bisimilar and language equivalent.

**Theorem 4.2** *Bisimilarity and language equivalence on Petri nets are undecidable.*

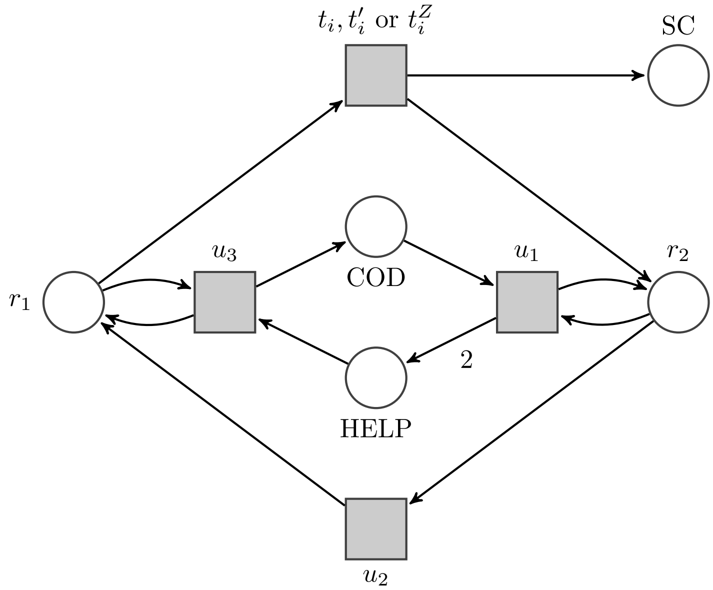
***Proof*** *There is a counter machine $C$ and an input $I$, such that the halting problem is undecidable.*
*The previous theorem describes a reduction of the halting problem on counter machines to both the bisimilarity- and the language-equivalence problem for petri nets, hence they are both undecidable.* □

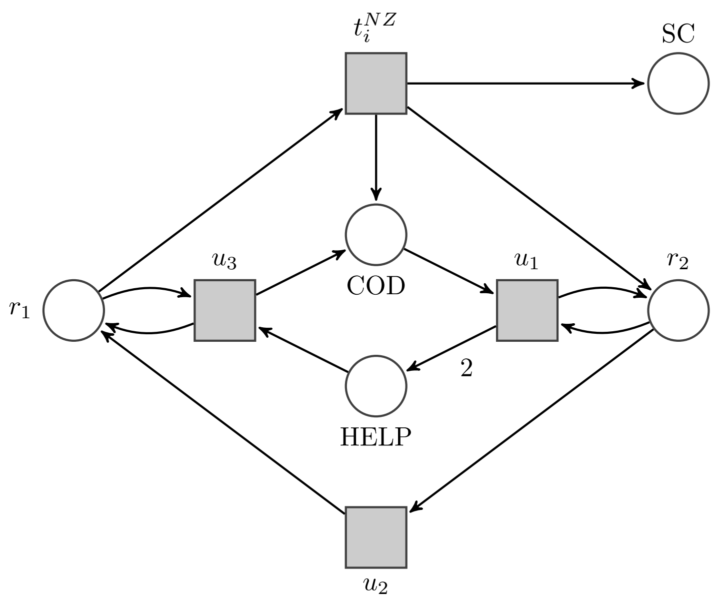
**Remark** Since there is a relevant counter machine C with only two counters, even for the subclass of labelled Petri nets with only 2 unbounded places, the two problems are undecidable.

# 5 Undecidability of language containment for Petri nets

**Definition** For a counter machine $C$ with $n$ commands, $m$ counters $c_1, \ldots, c_m$ and an input $x = x_1, \ldots, x_m$, the Petri net $N_{C,x}$ is constructed as follows:

1. Begin with $\Sigma_C$.
2. For each $i \in BS$, add a dc-transition $t'_i$. We call the so far constructed transitions *counted*.
3. Add places COD, HELP, SC, $r_1$, $r_2$.
4. For each transition t, add arcs $(r_1, t), (t, r_2), (t, SC)$.
5. For each $t_i^{NZ}$, add transition ($t_i^{NZ}$,COD).
6. Add transitions $u_1, u_2, u_3$ and arcs (COD,$u_1$), ($r_2$,$u_1$), ($u_1$,$r_2$), ($r_2$,$u_2$), ($u_2$,$r_1$), (HELP,$r_3$), ($r_1$,$u_3$), ($u_1$,$r_3$), as well as ($u_1$,HELP) with multiplicity 2.






7. Put $x_1, \ldots, x_m$ tokens in $c_1, \ldots, c_m$, 1 token in $s_1$ and 1 token in $r_1$.

Consider a "legal" sequence $t_1 t_2 \ldots t_k$ of transitions corresponding to a finite prefix of the steps of the computation of $C$. It is not possible to fire $t$ in $N_{C,x}$. What we can do is firing a sequence $\sigma = t_1\sigma_1 t_2\sigma_2 \ldots t_k\sigma_k$, where for each $j \in \{1, \ldots, k\}$, $\sigma_i$ is of the form $(u_1)^{a_j} u_2 (u_3)^{b_j}$ for some $a_j, b_j$.
Assume, before firing $t_j$, there are $p$ tokens on COD and 0 tokens on HELP:

- If $t_j = t_i^{NZ}$, firing $t_j(u_1)^{n+1}u_2(u_3)^{n+1}$ will result in $2(p+1)$ tokens in COD.

- Else, firing $t_j(u_1)^n u_2(u_3)^n$ will result in $2p$ tokens in COD.

In both cases this is the maximal possible increasing of the number of tokens in COD.
If n is even, and we look at it as a binary number, these steps can be seen as setting the last digit to 1, iff $t_j = t_i^{NZ}$ and afterwards shifting the number by 1 to the left. Doing this repeatedly, firing the whole maximal sequence with k counted transitions will encode the steps taken by $C$ in COD, while SC keeps track of the number of counted transitions taken and will have k tokens on it at the end. We can now state the relevant property of such a maximized sequence:

**Lemma 5.1** *Given a maximal sequence of the form $\sigma = t_1\sigma_1 t_2\sigma_2 \ldots t_k\sigma_k$ where exactly $t_1, t_2, \ldots, t_k$ are counted transitions, if $M$ is the marking reached by firing $\sigma$ in $N_{C,x}$, $M'$ is not reachable by firing any other sequence. Furthermore, for any firable sequence $\sigma' \neq \sigma$ containing $k$ counted transitions and yielding a marking $M'$, $M'(COD) < M(COD)$.*

A proof for this can be found in the original paper.

**Theorem 5.2** *Given a counter machine $C$ with $n$ commands and $m$ counters, as well as an input $x$, there are two Petri nets $N_1, N_2$ such that the following statements are equivalent:*

*(a) $C$ does not halt on $x$.*

*(b) $\mathscr{R}(N_1) = \mathscr{R}(N_2)$*

*(c) $\mathscr{R}(N_1) \subseteq \mathscr{R}(N_2)$*

***Proof*** *We construct $N_1, N_2$ as follows:*

1. *Begin with $N_{C,x}$.*
2. *Add places $p, p'$*
3. *For each dc-transition $t'_i$, add arcs $(p, t'_i), (t'_i, p')$*
4. *To get $N_2$, add a transition $t_a$, arc $(t_n, t_a)$ and put a token in $p$.*
5. *To get $N_1$, take $N_2$, add a transition $t_b$ and arcs $(t_n, t_b), (p, t_b), (t_b, p')$.*

*Since $((b) \Rightarrow (c))$ is trivial, we will show $((a) \Rightarrow (b))$ and $((c) \Rightarrow (a))$ to prove the theorem:*

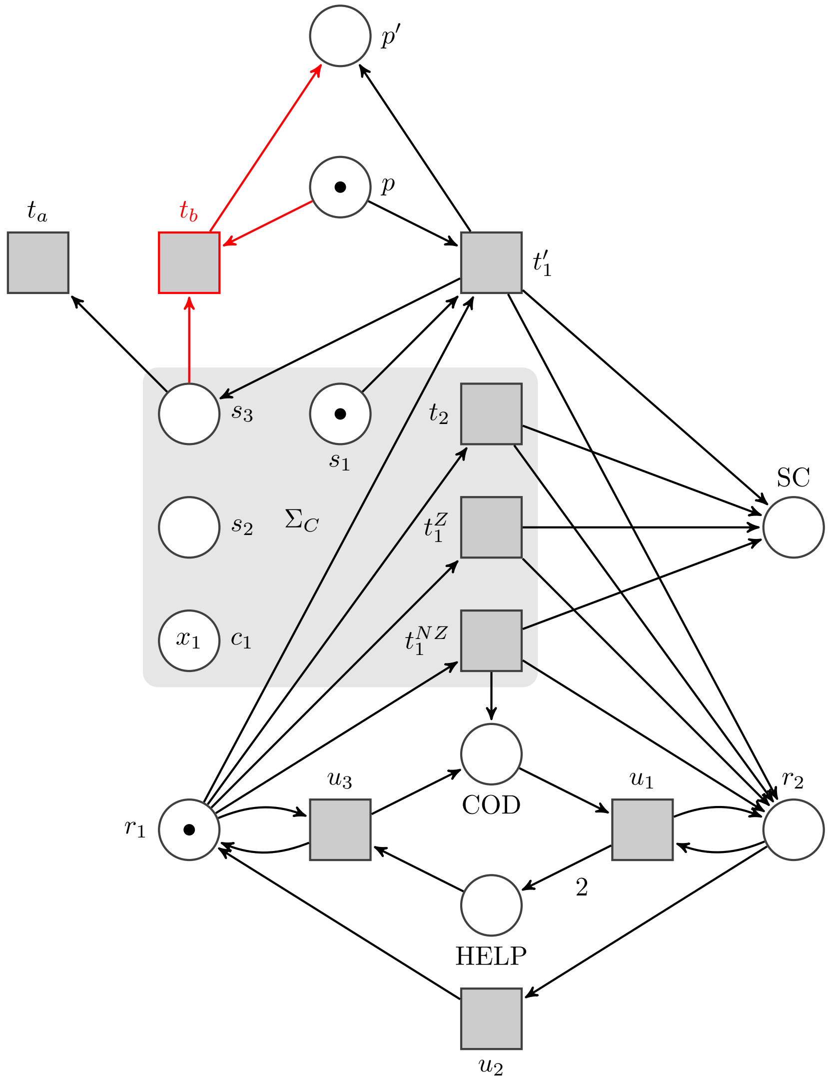
*$((c) \Rightarrow (a))$:*
*We prove $(\neg(a) \Rightarrow \neg(c))$ by showing that, if $C$ halts after $k+1$ computation steps , there is a marking $M$ that is reachable in $N_1$, but not in $N_2$:*
*Consider the maximized sequence $\sigma = t_1\sigma_1 t_2\sigma_2 \dots t_k\sigma_k$ mentioned earlier. Firing it in $N_1$ followed by $t_b$ yields a marking $M$ such that $M'(COD) = u$, $M'(SC) = k$, $M'(s_n) = 0$, $M'(p') = 1$. The only possibility for $N_2$ to reach a marking with $u$ tokens on $COD$ and $k$ tokens on $SC$ is by firing $\sigma$, which especially means, dc-transitions can't be taken. The only way to get rid of the token in $s_n$ is to fire $t_a$, which results in no enabled transitions and no token in $p'$. So $M$ is not reachable in $N_2$.*

*$((a) \Rightarrow (c))$:*
*Let $x$ be an input for $C$, such that $C$ does not halt on $x$:*
*The only difference between $N_1$ and $N_2$ is the existence of $t_b$ in $N_1$, hence it will suffice to show that, given a sequence $\pi = \pi_1\pi_2 \dots \pi_{j-1} t_b \pi_{j+1} \dots \pi_l$ of transitions in $N_1$, the reached marking $M_\pi$ is also reachable in $N_2$. Since $t_b$ requires tokens in and also removes tokens out of $s_n, p$, $\pi$ has the following properties:*

1. *$t_b$ occurs exactly once in $\pi$.*
2. *No counted transitions may occur in $\pi_{j+1} \dots \pi_l$.*
3. *No dc-transitions may occur in $\pi$.*

*Since $C$ does not halt, $\pi$ has to contain a cheating transition $t_i^Z$ prior to $t_b$. If we now replace $t_i^Z$ with $t'_i$ and $t_b$ with $t_a$, the resulting sequence will reach the same marking $M_\pi$ and be enabled in $N_2$.* □

## 5.1 Example




This is the construction of $N_1, N_2$ for the simple counter machine C from the last example (The arcs from $\Sigma_C$ are left out). The (red marked) transition $t_b$ is only part of $N_1$.

Let's look at different values for $x_1$:

$x_1 = 0$:
By firing $t_1^Z u_2 t_b$, a marking $M$ where $M(p') = 1$ is reachable in $N_1$. M is not reachable in $N_2$ . Note that this example does not show the necessity of the encoding subnet, because no nonzero-branch is taken in the original program.

$x_1 = 1$:
The only interesting markings are the ones reachable in $N_1$ with a word includ-

ing $t_b$. Since C does not halt and using $t'_1$ would disable $t_b$, all relevant paths which include $t_b$ also include $t_1^Z$ somewhere before. By substituting $t_1^Z$ by $t'_1$ and $t_b$ by $t_a$, we get a word which reaches the same marking and is also enabled in $N_2$.

**Theorem 5.3** *The reachability set containment problem on Petri nets is undecidable.*

***Proof*** *There is a counter machine C and a input I, such that the halting problem is undecidable. The previous theorem reduces the halting problem for counter machines to the reachability set containment problem for Petri nets, so the latter is undecidable.* □

**Remark** Since there is a relevant counter machine with only two counters, and besides from their respective places, only SC, COD and HELP are possibly unbounded in $N_1, N_2$, the result even holds for the subclass of petri nets with 5 unbounded places.

# 6 Decidability results

## 6.1 Deterministic nets

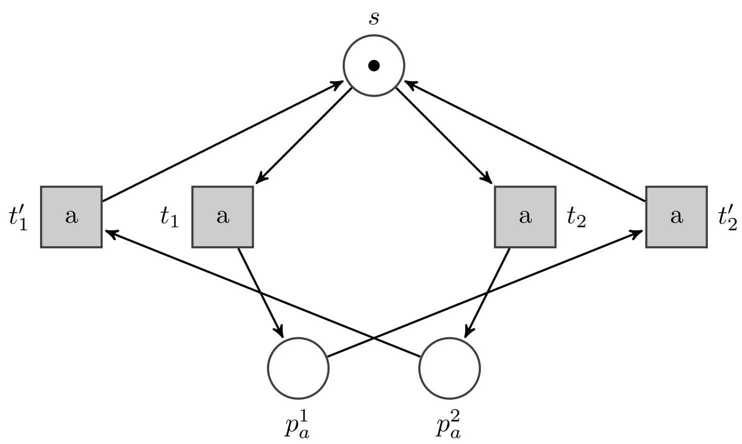
**Definition** A labelled Petri net $N = (\Sigma, M_0)$ is deterministic, if for every marking $M \in \mathscr{R}(M_0)$ and action $a$ , $|\{M'|M \xrightarrow{a}_\Sigma M'\}| \leq 1$.

**Definition** A labelled Petri net $N = (\Sigma, M_0)$ is deterministic up to bisimilarity, if for every marking $M \in \mathscr{R}(M_0)$, action $a$, and $M', M'' \in \{M'|M \xrightarrow{a}_\Sigma M'\}$, $M' \simeq M''$.

We begin with a construction which encodes the bisimulation game of two Petri nets in a so called game net:

**Definition** For two labelled Petri nets $N_1 = (P_1, T_1, F_1, L_1, M_{0,1})$ and $N_2 = (P_2, T_2, F_2, L_2, M_{0,2})$, such that $P_1 \cap P_2 = \emptyset$, $T_1 \cap T_2 = \emptyset$, the game net $N$ is constructed as follows:

1. For each transition $t$ of each net, add a copy transition $t'$ and also copy the relevant arcs. We refer to the so far constructed nets as $N'_1 = (P_1, T'_1, F'_1, L_1, M_{0,1})$, $N'_2 = (P_2, T'_2, F'_2, L_2, M_{0,2})$.
2. Take the union of $N'_1$ and $N'_2$ and add a place $s$ with 1 token on it.
3. For each action $a \in A$ and each pair of original transitions $(t_1, t_2) \in P_1 \times P_2$ with $L(t_1) = L(t_2) = a$, add two places $p_a^1, p_a^2$ and the arcs $(s, t_1), (t_1, p_a^1), (p_a^1, t'_2), (t'_2, s)$ as well as $(s, t_2), (t_2, p_a^2), (p_a^2, t'_1), (t'_1, s)$.




A round of the bisimulation game on such a game net has much simpler rules: The attacker fires a enabled transition labelled with an action a from one of the original nets, moving the token from s to $p_a^1$ or $p_a^2$. The defender fires an enabled transition, which by construction belonged to the other net, moving the token back in s. If eventually a state is reached, where there is a token in $p_a^1$ or $p_a^2$, and no transition is enabled, the attacker wins, in each other case, the defender wins.

**Theorem 6.1** *The bisimulation problem is decidable on the class of pairs of Petri nets, where one of them is deterministic up to bisimilarity.*

***Proof*** *If $N_1, N_2$ are labelled Petri nets, $N_2$ is deterministic up to bisimilarity, and we are playing the bisimulation game on the game net $N$, the attacker may always choose transitions from $N_1$, leaving the defender no choice but to fire one of the transitions leading into one of some bisimilar states. Because of that, bisimilarity on the original nets is equivalent to non-reachability of a marking with no enabled transitions, where a token is in $p_a^1$ or $p_a^2$ in the game net $N$. [Jančar] shows that there is an algorithm deciding this question.* □

**Remark** Note that any one-to-one labelled Petri net is deterministic and any deterministic labelled Petri net is deterministic up to bisimilarity, so the theorem holds for them too.

## 6.2 Semilinear bisimulations

We may assume that any transition in a Petri net requires at least one token in a place to be fired (If not, we add a dummy place $d$ with a token on it, as well as arcs $(d, t), (t, d)$ for each transition $t$ where this does not hold). Furthermore, for two Petri nets, we may take the union of the underlying nets and extend their initial markings by 0 for the missing places to get two new Petri nets with the same behaviour as their original versions. Hence it is no restriction only to consider pairs of Petri nets with the same underlying net.
We will also use the notion of semi-decidability. A problem is semi-decidable, if there is an algorithm which halts if the question to the given instance of the problem has to be answered with yes. It is known that, if both a problem and it's negated problem (in our case bisimilarity and non-bisimilarity) are

semidecidable, the problem itself is decidable.
Labelled Petri nets are a special case of finitely branching transition systems, for which non-bisimilarity is indeed semi-decidable (see [Christensen]). So we only have to show that, for our desired subclass of Petri nets, bisimilarity is semidecidable.
What's still missing for the promised proof is the notion of semilinear sets:

**Definition** We call a set $B \subset \mathbb{N}_0^k$ linear, if there are a basis $b \in \mathbb{N}_0^k$ and periods $c_1, c_2, \dots, c_n \in \mathbb{N}_0^k$ such that $B = \{b + x_1c_1 + x_2c_2 + \dots + x_nc_n | x_1, x_2, \dots, x_n \in \mathbb{N}_0\}$

**Definition** A set is semilinear set if it is a finite union of linear sets.
A relation on $\mathbb{N}_0^n$ is semilinear if it is semilinear as a subset of $\mathbb{N}_0^{2n}$.

**Theorem 6.2** *For the class of pairs of labelled Petri nets where bisimilarity implies the existence of a semilinear bisimulation relating the initial markings, bisimilarity is decidable.*

***Proof*** *We need to show that bisimilarity is semidecidable:*
*Linear sets can be identified by a matrix* $(b, c_1, c_2, \dots, c_n) \in \mathbb{N}_0^{k \times n}$*, or as a vektor* $v \in \mathbb{N}_0^{kn+1} \subset \mathbb{N}_0^*$ *(+1 to keep track of the size of the matrix). This means that a finite union of linear sets can be identified with elements of* $(\mathbb{N}_0^*)^*$*, which is countable (enumerations can be constructed e.g. by using prime factorizations).*
*Let* $B_0, B_1, B_2 \dots$ *be a enumeration of all semilinear sets. We now can give an algorithmic way semi-deciding the bisimilarity problem:*

*Given* $N_1 = (\Sigma, M_1), N_2 = (\Sigma, M_2)$*:*
***for*** $i = 0, 1, \dots$ ***do***
    ***if*** *(*$B_i$ *is a bisimulation and* $M_1 \; B \; M_2$*) (*)*
        *return ("*$N_1, N_2$ *are bisimilar");*
*Verification of (*) is decidable, as for the case of a semilinear* $B_i$ *the defining conditions for bisimilarity can be transformed into formulas of the Presburger arithmetic, which is decidable (see [Oppen]);* □

# 7 Conclusion

In the original paper it is emphasized that Petri nets are very closely related to vector addition systems, meaning that most of the results can easily be extended to them. It is also mentioned that the bisimilarity results concerning deterministic nets can easily be extended to weak bisimilarity by modifying the game net. The game net itself reduces the bisimilarity problem (for the relevant subclass of Petri Nets) to the reachability problem. This can also be done the other way round, as it is showed in Lemma 4.3, so the two problems are equally hard.
If you are interested in these and other related topics, i reccomend reading the original paper, as many of them are mentioned and referenced there.
Besides from the results of the paper, P. Jančar provides some elegant proof techniques regarding undecidability results for Petri nets. The bisimulation game is a beautiful mechanic to establish existence or non-existence of a bisimulation. As it is not the most formally defined method, he also shows a way

to reduce the complexity of the game rules, namely the game net. A lot of the proof schemes can be adopted or generalized to prove other facts concerning decidability or bisimulations.

# References

[Minsky] M. Minsky, *Computation: Finite and Infinite Machines* (Prentice Hall, Englewood Cliffs, NJ, 1967).

[Christensen] S. Christensen, Y. Hirshfeld, F Moller, Bisimulation equivalence is decidable for all basic parallel processes, in: Proc. CONCUR'93, Lecture Notes in Computer Science, Vol. 715 (Springer, Berlin, 1992) 143-157.

[Jančar] P. Jančar, Decidability of a temporal logic problem for Petri nets, Theoret. Comput. Sci. 74 (1990) 71-93.

[Oppen] D.C. Oppen, A $2^{2^{2^{pn}}}$ upper bound on the complexity of Presburger Arithmetic, J. Comput. System Sci. 16 (1978) 323-332