Checking Regular Properties of Petri Nets

Petr Jančar*
Department of Computer Science
University of Ostrava
Dvořákova 7
701 00 Ostrava 1
Czech Republic
email: jancar@osu.cz

Faron Moller†
Swedish Institute
of Computer Science
Box 1263
164 28 Kista
Sweden
email: fm@sics.se

Abstract

In this paper we consider the problem of comparing an arbitrary Petri net against one whose places may contain only a bounded number of tokens (that is, against a regular behaviour), with respect to trace set inclusion and equivalence, as well as simulation and bisimulation. In contrast to the known result that language equivalence is undecidable, we find that all of the above are in fact decidable. We furthermore demonstrate that it is undecidable whether a given Petri net is either trace equivalent or simulation equivalent to any (unspecified) bounded net.

1 Introduction

This paper is concerned with narrowing the gap between decidable and undecidable classes of Petri nets with respect to the verification of various properties. For example, it is known that for bounded Petri nets, (that is, regular or finite-state automata), all standard behavioural properties (such as language equivalence) are decidable, whereas for Petri nets with at most two unbounded places, all of these are undecidable [10]; and for nets in which every transition has a single input place—the so-called Basic Parallel Processes BPP—bisimulation equivalence is decidable [2] whereas all other standard equivalences are undecidable [7, 9].

In this paper we consider the problem of comparing an arbitrary Petri net against one with bounded places. This problem may arise for example if the specification of a system is given as a regular behaviour yet the system itself is implemented by a

*The first author is supported by the Grant Agency of the Czech Republic, Grant No. 201/93/2123; and also received partial support from Esprit Network EXPRESS in order to visit the Swedish Institute of Computer Science, during which time the research reported here was carried out.

†The second author is supported by Esprit Basic Research Action No. 7166, "CONCUR2".

more general Petri net. The purpose for having a general Petri net implementation of a regular system could be to reduce the complexity of the system; a small Petri net with some number of unbounded places may exhibit a regular behaviour which can be represented only by very large bounded Petri nets.

Although we are considering regular properties, the class of problems considered here is nontrivial. Petri nets lack the universal computing power of Turing machines—for which the checking of such regular properties is clearly undecidable—but they may weakly simulate Turing machines [10], a state of affairs which provides for many undecidability results. As an example, it has long been known that language equivalence is undecidable in this case [20], even if we restrict attention to Petri nets with only one unbounded place.

Despite this initial negative result, our decidability results are generally positive. We first demonstrate that the simulation preorder is decidable in both directions, thus also providing the decidability of simulation equivalence. We then demonstrate the decidability of trace set inclusion in both directions, thus also providing the decidability of trace set equivalence. Finally we provide a proof of the decidability of bisimulation equivalence.

We then consider the problem of deciding if a given Petri net is either trace equivalent or simulation equivalent to any (unspecified) regular behaviour. At this point our results become negative. In contrast to the known result that the regularity of the set of traces of a net with an injective labelling (that is, where the transitions have unique labels) is decidable [20], we demonstrate the undecidability of both of these problems. However, we leave the corresponding question regarding bisimilarity open.

2 Preliminary Definitions

In this section we define the framework in which we shall work, in particular, the definitions and relevant results regarding finite labelled Petri nets (place/transition nets). We start by fixing some alphabet Σ, which is a set of observable actions which will label the transitions of our nets. We let $\mathcal{N}$ denote the set of nonnegative integers, and A^* the set of finite sequences of elements of the set A.

A *(finite, labelled, place/transition Petri) net* is a tuple $N = \langle P, T, F, M_N, \ell \rangle$ where

- P and T are finite disjoint sets of *places* and *transitions*, respectively;
- $F : (P \times T) \cup (T \times P) \to \mathcal{N}$ is a *flow function*: for $F(x, y) > 0$, there is an *arc* from x to y with *multiplicity* $F(x, y)$;
- $M_N : P \to \mathcal{N}$ is the *(initial) marking* or *state*, where a marking (state) associates a number of *tokens* with each place; and
- $\ell : T \to \Sigma$ is a *labelling*, which associates an action to each transition. ℓ will also be understood in a broader sense, denoting the homomorphic extension $\ell : T^* \to \Sigma^*$.

A transition t is *enabled* at a marking M, denoted by $M[t\rangle$, if $M(p) \geq F(p,t)$ for every $p \in P$. A transition t enabled at a marking M may *fire* yielding the marking M', denoted by $M[t\rangle M'$, where $M'(p) = M(p) - F(p,t) + F(t,p)$ for all $p \in P$. For any $a \in \Sigma$, by $M \xrightarrow{a}$ (and $M \xrightarrow{a} M'$) we mean that $M[t\rangle$ (and $M[t\rangle M'$) for some t with $\ell(t) = a$. These definitions can be extended homomorphically to finite sequences of transitions $\sigma \in T^*$ and finite sequences of actions $w \in \Sigma^*$.

The *reachability set* of a net N is defined as

$$\mathcal{R}(N) = \{ M : M_N[\sigma\rangle M \text{ for some } \sigma \in T^* \}.$$

We shall often refer to the states reachable from some marking M by $\mathcal{R}(M)$; in this case, the underlying net will be clear from the context. A place $p \in P$ is *unbounded* if for any $n \in \mathcal{N}$ there is $M \in \mathcal{R}(N)$ such that $M(p) > n$. Note that there exists an algorithm (based on that presented in [12]) which determines the unbounded places of a given net, and thus in particular whether or not a net is bounded, that is, represents a finite-state behaviour.

The *trace set* of a net N is defined as

$$\mathcal{T}(N) = \{ w \in \Sigma^* : M_N \xrightarrow{w} \}.$$

Note that by a trace here we mean simply a sequence of actions, as used for example by van Glabbeek [5], and not the partially ordered structures used by Mazurkiewicz [15]. As with reachability sets, we shall often refer to the trace set of a marking M of a net by $\mathcal{T}(M)$; again, in such a case the underlying net will be clear from the context. Two nets N_1, N_2 are *trace equivalent* if $\mathcal{T}(N_1) = \mathcal{T}(N_2)$.

The standard automata-theoretic notion of the *language* of a net N is defined with respect to a given finite set $\mathcal{F}$ of final states:

$$\mathcal{L}_{\mathcal{F}}(N) = \{ w \in \Sigma^* : M_N \xrightarrow{w} M \text{ for some } M \in \mathcal{F} \}.$$

In fact, using standard techniques [18] we can assume this set $\mathcal{F}$ to be the singleton set 0 consisting only of the zero marking; given any net defining a language based on a particular finite set of markings, we can effectively construct a net accepting the same language (minus the empty word) based on the singleton zero-marking set. We then have the following result from [20].

Theorem 2.1 (Valk and Vidal-Naquet) *It is undecidable whether $\mathcal{L}_0(N) = \mathcal{L}_0(R)$, where N is a net with only one unbounded place and R is a bounded net.*

We can actually show in this case that $\mathcal{L}_0(N) \subseteq \mathcal{L}_0(R)$ is decidable, using a reduction to the (decidable) problem of the inclusion of the language of a given push-down automaton within a given regular set [8]. Hence we can deduce from the above proposition that $\mathcal{L}_0(R) \subseteq \mathcal{L}_0(N)$ is undecidable.

We now present standard definitions of simulation preorder and bisimulation equivalence. Given two nets N_1 and N_2 a relation S between their markings is a *simulation* if for all $\langle M_1, M_2 \rangle \in S$ and for all $a \in \Sigma$:

- if $M_1 \xrightarrow{a} M_1'$ then $M_2 \xrightarrow{a} M_2'$ for some M_2' such that $\langle M_1', M_2' \rangle \in S$.

It is a *bisimulation* if for all $\langle M_1, M_2 \rangle \in S$ and for all $a \in \Sigma$:

- if $M_1 \xrightarrow{a} M_1'$ then $M_2 \xrightarrow{a} M_2'$ for some M_2' such that $\langle M_1', M_2' \rangle \in S$; and
- if $M_2 \xrightarrow{a} M_2'$ then $M_1 \xrightarrow{a} M_1'$ for some M_1' such that $\langle M_1', M_2' \rangle \in S$.

Given markings M_1 and M_2 from two nets N_1 and N_2, M_1 is *simulated* by M_2, written $M_1 \preceq M_2$, if they are related by some simulation, and they are *bisimilar*, written $M_1 \sim M_2$, if they are related by some bisimulation. We shall also refer to a net as being simulated by, or bisimilar to, another net if their initial markings are so related. It is easily confirmed that these two relations are respectively a preorder and an equivalence.

We shall occasionally rely on a stratified characterisation of these two relations as presented as follows. We first stipulate that $M_1 \preceq_0 M_2$ and $M_1 \sim_0 M_2$ for all pairs $\langle M_1, M_2 \rangle$. Then for every $n \in \mathcal{N}$ we let $M_1 \preceq_{n+1} M_2$ whenever

- if $M_1 \xrightarrow{a} M_1'$ then $M_2 \xrightarrow{a} M_2'$ with $M_1' \preceq_n M_2'$;

and we let $M_1 \sim_{n+1} M_2$ whenever

- if $M_1 \xrightarrow{a} M_1'$ then $M_2 \xrightarrow{a} M_2'$ with $M_1' \sim_n M_2'$; and
- if $M_2 \xrightarrow{a} M_2'$ then $M_1 \xrightarrow{a} M_1'$ with $M_1' \sim_n M_2'$.

Again we freely apply these relations to nets by considering their initial marking. The relations $\preceq_n$ are easily seen to form a nonincreasing (with respect to subset inclusion) sequence of preorders, and the relations $\sim_n$ are equally easily seen to form a nonincreasing sequence of equivalences. Furthermore, all of these relations are easily decidable. The following is then a standard result [16].

Lemma 2.2 *$M_1 \preceq M_2$ if and only if $M_1 \preceq_n M_2$ for all $n \in \mathcal{N}$; and $M_1 \sim M_2$ if and only if $M_1 \sim_n M_2$ for all $n \in \mathcal{N}$.*

It is often useful to think of these relations in terms of two-player games [19]. The rules of the *bisimulation* game, in which the two players alternate moves, are described as follows.

1. The playing board is given by a pair of nets N_1 and N_2.
2. Player 1 chooses one of the nets and fires an enabled transition, changing the marking appropriately.
3. Player 2 responds by firing a transition with the same label in the other net.
4. A player wins the game if ever the other player cannot make a move.

The rules for the *simulation* game are identical except that the first player must always choose the first net in rule 2.

Lemma 2.3

1. *Player 1 has a winning strategy in the simulation game if and only if* $N_1 \not\preceq N_2$*; in other words, Player 2 has a defending strategy if and only if* $N_1 \preceq N_2$.

   More precisely, Player 1 may force a win within k *exchanges of moves if and only if* $N_1 \not\preceq_k N_2$;

2. *Player 1 has a winning strategy in the bisimulation game if and only if* $N_1 \not\sim N_2$*; in other words, Player 2 has a defending strategy if and only if* $N_1 \sim N_2$.

   More precisely, Player 1 may force a win within k *exchanges of moves if and only if* $N_1 \not\sim_k N_2$;

Proof: If there is a simulation (bisimulation) S relating the initial markings of the nets, then Player 2 is always able to respond to moves by Player 1 in such a way that the resulting pair of markings is contained in the relation S; hence she has a defending strategy.

Conversely, if Player 2 has a defending strategy then the collection of all pairs of markings which appear after every exchange of moves during any and all games in which Player 2 uses this defending strategy defines a simulation (bisimulation).

The proofs for the stratified play are easily carried out by induction on k. □

We shall occasionally make use of the following result known as Higman's Theorem [6]. A partially-ordered set $\langle A, \leq \rangle$ has the *finite basis property (fbp)* if every infinite sequence of elements of A has an infinite (not necessarily strictly) ascending subsequence. For example, the integers $\langle \mathcal{N}, \leq \rangle$ equipped with the usual ordering has the finite basis property.

Theorem 2.4 *If* $\langle A, \leq \rangle$ *has the fbp then so does* $\langle A^*, \widehat{\leq} \rangle$*, where*

$$\widehat{\leq} \;=\; \Big\{ \langle a_1 a_2 \cdots a_n,\, v_0 b_1 v_1 b_2 \cdots v_{n-1} b_n v_n \rangle \;:\; a_i, b_i \in A,\, v_i \in A^*,\, a_i \leq b_i \Big\}.$$

A simple instance of Higman's Theorem, known as Dickson's Lemma [3], is presented as follows (and can be proven independently of Higman's Theorem using an obvious induction on k.)

Lemma 2.5 $\langle \mathcal{N}^k, \leq \rangle$ *has the finite basis property: given an infinite sequence of vectors* $\vec{x}_1, \vec{x}_2, \vec{x}_3, \ldots \in \mathcal{N}^k$ *we can always find* i, j *with* $i < j$ *such that* $\vec{x}_i \leq \vec{x}_j$ *(where* $\leq$ *is considered pointwise).*

We shall actually use this lemma as applied to markings of nets, where we view markings as vectors, and define the ordering between markings accordingly: $M \leq M'$ if and only if $M(p) \leq M'(p)$ for all places $p \in P$.

3 Simulation Preorder

In this section we suppose R to be a bounded (finite-state) net with states (markings) ranged over by s and s' and initial state s_R; and N to be an arbitrary net with markings ranged over by M and M' and initial marking M_N. We shall demonstrate that it is decidable both whether or not $R \preceq N$ and whether or not $N \preceq R$. First we provide a technical definition.

Definition 3.1 *Let S be a finite set of pairs of markings of nets N_1 and N_2. A set E is a* (one-step) simulation expansion *of S if*

- *for every pair $\langle \alpha, \beta \rangle$ in S and for every transition $\alpha \xrightarrow{a} \alpha'$ there is a transition $\beta \xrightarrow{a} \beta'$ such that $\langle \alpha', \beta' \rangle \in E$;*
- *E is minimal: no proper subset of E satisfies the above.*

Note that there are only finitely many simulation expansions of a given finite set of pairs, and that each of these is finite. Also, some simulation expansion of any (finite subset of a) simulation relation must be contained in that simulation relation. Finally note that there exists an expansion of a set S if and only if $\alpha \preceq_1 \beta$ for every $\langle \alpha, \beta \rangle \in S$, so there is no expansion exactly when there is some $\langle \alpha, \beta \rangle \in S$ such that $\alpha \not\preceq_1 \beta$; and if for every pair $\langle \alpha, \beta \rangle$ in S there are no transitions $\alpha \xrightarrow{a} \alpha'$, then the only simulation expansion of S is $\emptyset$.

Theorem 3.2 *$R \preceq N$ is decidable.*

Proof: Define a tree with root node $\{ \langle s_R, M_N \rangle \}$ and such that the children of a node are precisely the simulation expansions of the node with the following modification: we shall omit any pair $\langle s, M \rangle$ such that some ancestor node contains a pair $\langle s, M' \rangle$ with $M' \leq M$. (Note that when $M' \leq M$, we have $s \preceq M'$ implies $s \preceq M$, as $M' \preceq M$.) The node corresponding to the empty set is deemed to be a leaf of the tree. From Dickson's Lemma 2.5, along with König's Lemma (that every infinite finite-branching tree has an infinite path through it), it follows that the tree must be finite. (Note that a nonempty node is a leaf if it has no simulation expansion—that is, there is some (s, M) such that $s \not\preceq M$.)

If this tree has a branch terminating with a leaf labelled by the empty set, then it is straightforward to demonstrate that the set

$$\left\{ \langle s, M \rangle \ : \ \langle s, M' \rangle \text{ appears in this branch for some } M' \leq M \right\}$$

is a simulation, thus demonstrating $R \preceq N$.

Conversely, if $R \preceq N$ then there must be a branch terminating with the empty set, as some branch must contain only subsets of a simulation relation containing the root node. □

To demonstrate the decidability of $N \preceq R$, we use the following technical notion.

Definition 3.3 *Let* $\mathrm{Rem}_p(N)$ *denote the net derived from* N *by deleting the place* p *along with all arcs leading into or out of it. For any marking* M *of* N*, we shall denote by* $\mathrm{Rem}_p(M)$ *the marking of* $\mathrm{Rem}_p(N)$ *which coincides with* M *on the non-*p *places.*

The net $\mathrm{Rem}_p(N)$ thus behaves like N under the assumption that the place p always contains as many tokens as the maximum multiplicity of any of its outgoing arcs; that is, a transition can never be inhibited by the lack of sufficient tokens on the place p.

Lemma 3.4

1. *If* $\mathrm{Rem}_p(M) \preceq s$ *then* $M \preceq s$.
2. *If* $\mathrm{Rem}_p(M) \not\preceq s$ *then we may compute a value* v *such that* $M[p \mapsto v] \not\preceq s$*, and hence* $M'[p \mapsto v] \not\preceq s$ *for* $M' \geq M$*. (Here we use* $M[p \mapsto v]$ *to denote the mapping which coincides with* M *everywhere except on the place* p *where we have* $M[p \mapsto v](p) = v$*.)*

Proof:

1. This follows from the trivial observation that $M \preceq \mathrm{Rem}_p(M)$, which follows from noting that the collection of all such pairs constitutes a simulation relation.
2. If $\mathrm{Rem}_p(M) \not\preceq s$, then we may compute a value q such that $\mathrm{Rem}_p(M) \not\preceq_q s$. (That is, Player 1 may force a win within q moves.) Taking $v = q \cdot w$, where w is the maximum multiplicity of any arc emanating from p, would then give us our desired result: $M[p \mapsto q \cdot w] \not\preceq s$. (Player 1 can simply use the same strategy for forcing a win within q moves.) We then get $M'[p \mapsto v] \not\preceq s$ for $M' \geq M$ from the fact that $M[p \mapsto q \cdot w] \preceq M'[p \mapsto q \cdot w]$. □

Theorem 3.5 $N \preceq R$ *is decidable.*

Proof: We proceed by induction on the number k of places of N. If N has no places ($k = 0$) then it clearly defines a finite-state (in fact, a one-state) behaviour, so the problem reduces to the (decidable) problem of comparing finite-state behaviours.

Suppose now that N has $k > 0$ places and that we can decide simulation preorder between a net with $k - 1$ places and a bounded net.

Define a tree with root node $\{ \langle M_N, s_R \rangle \}$ and such that the children of a node are precisely the collection of simulation expansions of the node with the following modifications: we shall omit any pair $\langle M, s \rangle$ such that either some ancestor node contains a pair $\langle M', s \rangle$ with $M' \geq M$ (in which case $M' \preceq s$ imples $M \preceq s$), or such that $\mathrm{Rem}_p(M) \preceq s$ for some place p. (This latter condition can be decided due to the inductive hypothesis). A node is deemed to be a leaf of the tree if it corresponds to the empty set, or if it contains a pair $\langle M, s \rangle$ such that, for some place p, $\mathrm{Rem}_p(M) \not\preceq s$ and $M(p) \geq v$, where v is as given by Lemma 3.4(2). Again from Dickson's Lemma 2.5 and Kŏnig's Lemma it follows that this tree must be finite.

If this tree has a branch terminated with a leaf labelled by the empty set, then it is straightforward to demonstrate that the set

$$\Big\{ \langle M, s \rangle \ : \ \mathrm{Rem}_p(M) \preceq s \text{ for some } p, \quad or \\ \langle M', s \rangle \text{ appears in this branch for some } M' \geq M \Big\}$$

is a simulation, thus demonstrating $N \preceq R$.

Conversely, if $N \preceq R$ then there must be a branch terminated with the empty set, as some branch must contain only subsets of a simulation relation containing the root node. □

4 Trace Set Inclusion

In this section we again suppose R to be a bounded net with states (markings) ranged over by s and s' and initial state s_R; and N to be an arbitrary net with markings ranged over by M and M' and initial marking M_N. We shall demonstrate that it is decidable both whether or not $\mathcal{T}(R) \subseteq \mathcal{T}(N)$ and whether or not $\mathcal{T}(N) \subseteq \mathcal{T}(R)$. We can assume now though that R is a deterministic automaton: for each state s of R and each $a \in \Sigma$, there is at most one state s' of R such that $s \xrightarrow{a} s'$.

We begin with the easy direction.

Theorem 4.1 *$\mathcal{T}(N) \subseteq \mathcal{T}(R)$ is decidable.*

Proof: This follows from Theorem 3.5 (using the assumption that R is a deterministic automaton) once we note that the set

$$\Big\{ \langle M, s \rangle \ : \ \mathcal{T}(M) \subseteq \mathcal{T}(s) \Big\}$$

is a simulation; demonstrating this is straightforward. □

Theorem 4.2 *$\mathcal{T}(R) \subseteq \mathcal{T}(N)$ is decidable.*

Proof: Define a tree whose nodes are labelled by elements of $\mathcal{R}(R) \times 2^{\mathcal{R}(N)}$ with the root node labelled by $\langle s_R, \{M_N\} \rangle$, and whose edges are labelled by actions from Σ. For any transition $s \xrightarrow{a} s'$ the node $\langle s, A \rangle$ has a successor $\langle s', A' \rangle$ along an edge labelled by a, where A' contains the maximal (and hence pairwise incomparable) M' such that $M \xrightarrow{a} M'$ for some $M \in A$. The tree is thus constructed successively in such a way that for any node $\langle s, A \rangle$ reached from the root following a path labelled by w we have $s_R \xrightarrow{w} s$, and that A is the (finite) set of all maximal markings M such that $M_N \xrightarrow{w} M$; $A = \emptyset$ means that no marking M can be reached from M_N by performing the action sequence w.

The node $\langle s, \emptyset \rangle$ is deemed to be an unsuccessful leaf; and the node $\langle s, A \rangle$ (for $A \neq \emptyset$) is deemed to be a successful leaf if either there exists no transition from s, or if the node has an ancestor labelled by $\langle s, B \rangle$ such that for every $M \in B$ there is

some $M' \in A$ such that $M \leq M'$. Due to Higman's Theorem 2.4 all branches must be finite, and therefore the constructed tree is finite. (For the application of Higman's Theorem, we view elements of $2^{\mathcal{R}(N)}$ as strings of markings with an arbitrarily-chosen ordering.)

If an unsuccessful node occurs then we can easily demonstrate that $\mathcal{T}(R) \not\subseteq \mathcal{T}(N)$; the label of the path to this unsuccessful node defines a trace of R which is not a trace of N. Conversely, if all nodes are successful, then we can easily demonstrate that $\mathcal{T}(R) \subseteq \mathcal{T}(N)$; the crucial fact is that $\mathcal{T}(M) \subseteq \mathcal{T}(M')$ when $M \leq M'$. □

5 Bisimulation Equivalence

In this section we again suppose R to be a bounded net with states (markings) ranged over by s and s' and initial state s_R; and N to be an arbitrary net with markings ranged over by M and M' and initial marking M_N. We shall demonstrate that it is decidable whether or not $R \sim N$. We can assume now that R is minimal with respect to bisimilarity: $s \not\sim s'$ whenever $s \neq s'$. In particular, this implies that $s \not\sim_{n-1} s'$ whenever $s \neq s'$, where R consists of n states. (This latter result is easily seen to be true by noting that the sequence of approximation equivalences $\sim_0, \sim_1, \ldots$ must stabilise within n steps.)

To establish our result, we use some known results concerning semilinear sets. A set $V \subseteq \mathcal{N}^k$ of vectors is *linear* if there are vectors $\vec{y}, \vec{x}_1, \vec{x}_2, \ldots, \vec{x}_p \in \mathcal{N}^k$ such that

$$V = \left\{ \vec{y} + c_1\vec{x}_1 + c_2\vec{x}_2 + \ldots + c_p\vec{x}_p \ : \ c_i \in \mathcal{N} \right\}.$$

V is *semilinear* if it is a finite union of linear sets.

Lemma 5.1 *It is decidable, given any net N and any semilinear set A, whether or not $\mathcal{R}(N) \subseteq A$.*

Proof: The complement $\overline{A}$ of the semilinear set A is an effectively computable semilinear set [4]. In a straightforward way, a net N' can be constructed such that $\mathcal{R}(N')\lceil P' = \overline{A}$ for some subset P' of places of N'. Using standard techniques we can then construct a net N'' from N and N' such that the zero marking is reachable in N'' if and only if there is some $M \in \mathcal{R}(N) \cap \overline{A}$. The result then follows from the decidability of the reachability problem [14]. □

Definition 5.2 *For our given nets R and N, let*

- $A_i = \left\{ \langle s, M \rangle \ : \ s \sim_i M \right\}$ *for each $i \in \mathcal{N}$; and*
- $B = \left\{ M \ : \ s \sim_n M \ \text{ for some } s \right\}$.

Note that for any M there is at most one s such that $s \sim_{n-1} M$. Hence for any $M \in B$, $s \sim_{n-1} M$ implies $s \sim_n M$.

Lemma 5.3 *A_i for each $i \in \mathcal{N}$, and B, are effectively computable semilinear sets.*

Proof: A_0 is clearly effectively semilinear. Suppose now that A_i is effectively semilinear. Note that $A_{i+1} = \{ \langle s, M \rangle$: any move $s \xrightarrow{a} s'$ is matched by a move $M \xrightarrow{a} M'$ such that $\langle s', M' \rangle \in A_i$ and vice versa $\}$. Since A_i is semilinear, we can easily characterise A_{i+1} by a formula in Presburger arithmetic (theory of addition); it is known [4] that any such formula can be effectively transformed into a semilinear set of its true values. Hence A_{i+1} is also effectively semilinear and we have completed the induction argument.

B is effectively semilinear as it is a restriction of the effectively semilinear set A_n. □

Theorem 5.4 *$R \sim N$ is decidable.*

Proof: We shall show that $R \sim N$ if and only if $R \sim_n N$ and $\mathcal{R}(N) \subseteq B$, which can be checked due to Lemma 5.1.

If $R \sim N$, then $R \sim_n N$, and for any $M \in \mathcal{R}(N)$ there is $s \in \mathcal{R}(R)$ such that $s \sim M$. But then $s \sim_n M$, and hence $M \in B$.

Now suppose $R \sim_n N$ and $\mathcal{R}(N) \subseteq B$. Then

$$\left\{ \langle s, M \rangle \ : \ s \sim_n M \text{ and } M \in \mathcal{R}(N) \right\}$$

can be easily seen to be a bisimulation containing the pair $\langle s_R, M_N \rangle$ thus demonstrating that $R \sim N$. □

6 Undecidability of Regularity Testing

In this section we demonstrate that it is undecidable whether or not a given net is either trace equivalent, or simulation equivalent, to some (unspecified) bounded net. Our trace equivalence result is in contrast to the decidability result of [20] in the case of a net with an injective labelling, that is, where the transitions are all uniquely labelled. To do this, we rely on the undecidability of the halting problem for Minsky (counter) machines [17]. In particular, given an arbitrary Minsky machine C, we construct a net N_C (as in [10] with a modification inspired by [7]) for which we can demonstrate the following.

1. If the machine C halts, then the net N_C is simulation equivalent to some finite-state behaviour R. As a corollary, N_C and R are trace equivalent as well.

2. If the machine C does not halt, then the net N_C is not trace equivalent to any finite-state behaviour. Hence it cannot be simulation equivalent to any finite-state behaviour, either.

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

$$\begin{array}{ll} \mathtt{X}_1 & : \mathrm{comm}_1 \\ \mathtt{X}_2 & : \mathrm{comm}_2 \\ & \cdots \\ \mathtt{X}_{n-1} & : \mathrm{comm}_{n-1} \\ \mathtt{X}_n & : \mathtt{halt} \end{array}$$

representing a simple program which uses counters c_1, c_2, ..., c_m, where each of the first $n-1$ instructions is either of the form

X : $\mathtt{c}_j\mathtt{:=c}_j\mathtt{+1;\ goto}\ X'$

or of the form

X : $\mathtt{if\ c}_j\mathtt{=0\ then\ goto}\ X'$
$\mathtt{else\ c}_j\mathtt{:=c}_j\mathtt{-1;\ goto}\ X''$

A Minsky machine C starts executing with the value 0 in each of the counters and the control at the label $\mathtt{X}_0$. When the control is at label $\mathtt{X}_k$ ($1 \leq k < n$), the machine executes instruction 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 `halt` instruction at label $\mathtt{X}_n$. We recall now the fact that the halting problem for Minsky Machines is undecidable: there is no algorithm which decides whether or not a given Minsky machine halts.

A Minsky machine C as described above gives rise to a particular net N_C with transitions labelled by $\Sigma = \{ i, d, z \}$ as follows.

- The places of the net are given by $\{ c_1, c_2, \ldots, c_m, , X_1, X_2, \ldots, X_n, U \}$. The initial marking consists of just one token, located on the place X_1.
- For every instruction X of the form

  X : $\mathtt{c}_j\mathtt{:=c}_j\mathtt{+1;\ goto}\ X'$

  the net has a transition labelled by i (for "increment") with the single input place X and the two output places X' and c_j; see Figure 1(i). These arcs, as with all arcs in the net, have multiplicity 1.
- For every instruction X of the form

  X : $\mathtt{if\ c}_j\mathtt{=0\ then\ goto}\ X'$
  $\mathtt{else\ c}_j\mathtt{:=c}_j\mathtt{-1;\ goto}\ X''$

  the net has a transition labelled by d (for "decrement") with the two input places X and c_j, and the one output place X''; and two transitions labelled by z (for "zero"), the first with the single input place X and the single output place X', and the second with the two input places X and c_j, and the one output place U; see Figure 1(ii).

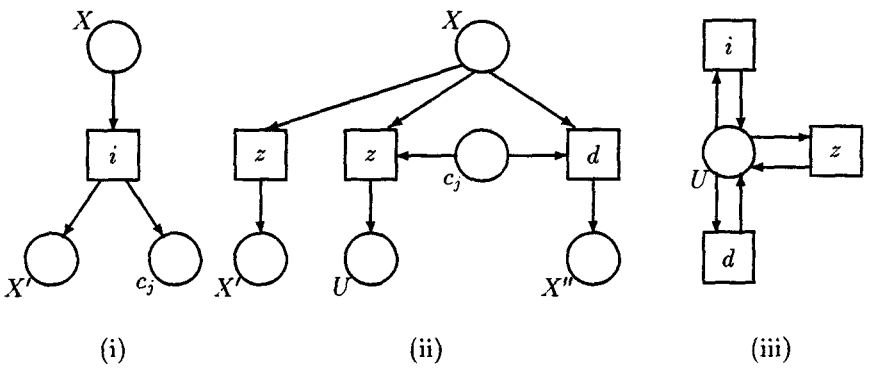


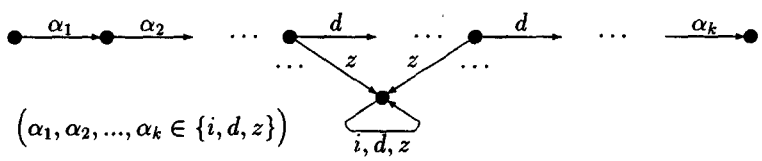
Figure 1: Constructions for N_C

- there are three further transitions associated with the place U (for "universal"). They each have U as both their single input place and their single output place, and they are labelled by i, d, and z, respectively; see Figure 1(iii).

The net N_C simulates the Minsky machine C in a weak sense: there is a unique computation of the net corresponding to the computation of the machine, and any incorrect ("dishonest") transition in the net can be made to lead to the universal state where any action is possible forevermore.

Lemma 6.1 *If C halts then N_C is simulation equivalent to a finite-state behaviour (i.e. a bounded net) R. Thus also $\mathcal{T}(N_C) = \mathcal{T}(R)$.*

Proof: The basis of the graph of R is the (finite) path corresponding to the (correct) computation of C (which halts by assumption). Outside of this path there is one further state which has three loops, labelled by i, d, and z respectively. From any state on the path which has an outgoing arc d, we add a further outgoing arc labelled by z and directed to this extra state. The finite-state behaviour can thus be pictured as follows.



It can be easily verified that N_C both simulates, and is simulated by, this R. □

For the opposite direction, we shall assume without loss of generality that in any infinite computation of C we can find for any $q \in \mathcal{N}$ a subcomputation during which some counter is decreased q times in succession. This is possible by including two

extra counters, along with a piece of code which gets executed after every original progam statement, which has the effect of incrementing the first new counter, setting the second new counter to the value of the first, and then decrementing it until it reaches zero. This straightforward transformation is clearly possible, and leads to longer and longer sequences of decrement actions, without changing the effect of the original program.

Lemma 6.2 *If C does not halt then $\mathcal{T}(N_C)$ is different from the trace set of any finite-state behaviour. Hence N_C is not trace equivalent, and thus also not simulation equivalent, to any finite-state behaviour.*

Proof: Suppose that $\mathcal{T}(N_C) = \mathcal{T}(R)$ for some finite-state behaviour R with, say, q states. Then R must accept the prefix of an honest computation sequence which includes a contiguous sequence of q decrement actions. Using the Pumping Lemma for finite-state behaviours [8], this means that R must be able to reach a state by following an honest computation sequence from which it can follow an arbitrary number of decrement actions, which clearly is not possible for N_C. Hence the trace sets of N_C and R cannot be equal. □

As described, these two results immediately yield our results, as summarized as follows.

Theorem 6.3

1. *It is undecidable whether or not a given net is simulation equivalent to some (unspecified) finite-state behaviour.*
2. *It is undecidable whether or not a given net is trace equivalent to some (unspecified) finite-state behaviour.*

As a final note, we leave the analogous question regarding bisimilarity open.

7 Conclusion

In this paper we studied the problem of deciding regular properties of Petri nets. In particular we demonstrated the following results:

- Given an arbitrary Petri net and a bounded Petri net,
  - it is decidable if the trace set of one is included in the trace set of the other, and thus it is decidable if they are trace equivalent;
  - it is decidable if one simulates the other, and thus it is decidable if they are simulation equivalent; and
  - it is decidable if they are bisimulation equivalent.

- Given an arbitrary Petri net,
  - it is undecidable if it is either simulation equivalent or trace equivalent to some (unspecified) bounded net.

In [20] Valk and Vidal-Naquet study similar questions, and arrive at results which contrast interestingly with ours. In particular, they demonstrate that language equivalence between a given net and a given bounded net is undecidable, while it is decidable if a given net with uniquely-labelled transitions is trace equivalent to some (unspecified) bounded net.

Mauw and Mulder [13] study a similar problem within the framework of context-free processes, so-called BPA-systems. Specifically, they demonstrate the decidability of the regularity of BPA-systems with respect to bisimulation equivalence. Within the framework of BPA, they are free to exploit the algebraic structure of terms allowed by the sequential composition operator of BPA. However in our instance we have no obvious such structure, and have left open the question of deciding if an arbitrary net is bisimilar to some (unspecified) bounded net.

In [11] the questions addressed here are extended to relations involving weak equivalences, that is, those which allow hidden internal events. In particular, the decidability results demonstrated here for trace set preorder and equivalence are extended in [11] to the weak versions of these relations; and the problem of deciding if a regular behaviour weakly simulates a net is decidable, whereas the opposite direction is only shown to be semidecidable. Finally it is shown that weak bisimulation equivalence is undecidable.

Our interest in the problems addressed in the paper was enhanced by inspiring discussions with Parosh Abdulla and Kārlis Čerāns on the problem of comparing two one-counter Petri nets, that is, nets with only one unbounded place. Partial results towards that study are described in [1] where we demonstrate the results presented in the present paper in the subcase of comparing one-counter nets vs bounded nets. The present paper thus extends all of the results presented in the earlier paper. However, both the decidability questions for one-counter nets, and the motivation given in the introduction of this paper—that is, that of implementing complex regular behaviours with simple general nets—are deserving of further exploration.

References

[1] Abdulla, P., K. Čerāns, P. Jančar, and F. Moller. One counter Petri nets vs bounded Petri nets. Research Report, 1995.

[2] Christensen, S., Y. Hirshfeld, and F. Moller. Bisimulation is decidable for basic parallel processes. In E. Best (editor), *Proceedings of CONCUR'93: Concurrency Theory, Lecture Notes in Computer Science* **715**, pp143–157, Springer-Verlag, 1993.

[3] Dickson, L.E. Finiteness of the odd perfect and primitive abundant numbers with distinct factors. *American Journal of Mathematics* **35**, pp413–422, 1913.

[4] Ginsburg, S., and E. Spanier. Semigroups, Presburger formulas, and languages. *Pacific Journal of Mathematics* **16**, pp285–296, 1966.

[5] van Glabbeek, R.J. The linear time – branching time spectrum. In J.C.M. Baeten, and J.W. Klop (editors), *Proceedings of CONCUR'90: Concurrency Theory, Lecture Notes in Computer Science* **458**, pp278–297, Springer-Verlag, 1990.

[6] Higman, H. Ordering by divisibility in abstract algebras. *Proceedings of the London Mathematical Society* **3**(2), pp326–336, 1952.

[7] Hirshfeld, Y. Petri nets and the equivalence problem. In E. Börger, Y. Gurevich, and K. Meinke (editors), *Proceedings of CSL'93: Computer Science Logic, Lecture Notes in Computer Science* **832**, pp165–174, Springer-Verlag, 1994.

[8] Hopcroft, J.E., and J.D. Ullman. **Introduction to Automata Theory, Languages, and Computation.** Addison Wesley, 1979.

[9] Hüttel, H. Undecidable equivalences for basic parallel processes. In R.K. Shyamasundar (editor), *Proceedings of FSTTCS'93: Foundations of Software Technology and Theoretical Computer Science, Lecture Notes in Computer Science* **761**, Springer-Verlag, 1993.

[10] Jančar, P. Undecidability of bisimilarity for Petri nets and some related problems. *Journal of Theoretical Computer Science* (to appear). (A preliminary version appears in P. Enjalbert, E.W. Mayr and K.W. Wagner (editors), *Proceedings of STACS'94: Symposium on Theoretical Aspects of Computer Science, Lecture Notes in Computer Science* **775**, pp581–592, Springer-Verlag, 1994.)

[11] Jančar, P. Decidability questions for equivalences on Petri nets. Czech Habilitation Thesis. Masaryk University, Brno. (Submitted April 1995.)

[12] Karp, R., and R. Miller. Parallel program schemata. *Journal of Computer and System Sciences* **3**, pp167–195, 1969.

[13] Mauw, S., and H. Mulder. Regularity of BPA-systems is decidable. In B. Jonsson and J. Parrow (editors), *Proceedings of CONCUR'94: Concurrency Theory, Lecture Notes in Computer Science* **836**, pp34–47, Springer-Verlag, 1993.

[14] Mayr, E. An algorithm for the general Petri net reachability problem. *SIAM Journal of Computing* **13**, pp441–460, 1984.

[15] Mazurkiewicz, A. Trace theory. In W. Brauer, W. Reisig, and G. Rozenberg (editors), *Petri Nets: Applications and Relationships to Other Models of Concurrency, Lecture Notes in Computer Science* **255**, pp279–324, Springer-Verlag, 1987.

[16] Milner, R. **Communication and Concurrency**. Prentice Hall, 1989.

[17] Minsky, M. **Computation: Finite and Infinite Machines.** Prentice Hall, 1967.

[18] Peterson, J.L. **Petri Net Theory and the Modeling of Systems.** Prentice Hall, 1981.

[19] Stirling, C. Local model checking games. Department of Computer Science Research Report, University of Edinburgh, 1995.

[20] Valk, R., and G. Vidal-Naquet. Petri nets and regular languages. *Journal of Computer and System Sciences* **23**(3), pp299–325, 1981.