Nonprimitive recursive complexity and undecidability for Petri net equivalences *

Petr Jančar

Dept. of Computer Science, Technical University of Ostrava, 17. listopadu 15, CZ-708 33 Ostrava, Czech Republic

Abstract

The aim of this note is twofold. Firstly, it shows that the undecidability result for bisimilarity in [6] can be immediately extended for the whole range of equivalences (and preorders) on labelled Petri nets. Secondly, it shows that restricting our attention to nets with finite reachable space, the respective (decidable) problems are nonprimitive recursive; this approach also applies to Mayr and Meyer's result [10] for the reachability set equality, yielding a more direct proof.

Key words: Petri-nets, decidability, complexity

1 Introduction

For the verification of systems, the extent to which it can be done automatically, i.e. algorithmically, is of great importance. This fact motivates the research to explore the decidability/undecidability border for behavioural equivalences and preorders ('implementations') on various classes of infinite-state systems (cf. e.g. [8]).

(Labelled) place/transition Petri nets comprise one of the well-studied models of such systems, with interesting (un)decidability results (cf. [2] for a survey). While the undecidability of reachability set equality, as well as of language equivalence, has been known since the seventies (cf. [4], [5] by Hack, who used Rabin's technique for the containment problem), the similar question for bisimilarity, which was recognized as a central behavioural equivalence during the eighties (cf. [9]), had been open for some time. Its undecidability is shown in [6]; in addition, a shorter and simpler proof for the reachability

* Supported by the Grant Agency of the Czech Republic, Grant No. 201/97/0456

set equality problem is given there as well. The simplification is mainly due to the fact that a straightforward reduction from the halting problem (for Minsky counter machines) was used while Rabin's proof relies on Hilbert's 10th problem and the reduction is technically more complicated.

In fact, the technique of the proof for bisimilarity implies undecidability for all reasonable action-based behavioural equivalences. This important consequence was not derived in [6] and it is done here [1].

For finite-state systems, the verification problems are mostly easily seen to be decidable, and the efficiency of the respective algorithms is of real interest. Here we will consider the equivalence questions for labelled Petri nets with finite reachability sets, *r-finite nets* for short; we could use the equivalent term of *bounded nets* but without an explicitly given bound. In this case Mayr and Meyer [10] show that the reachability set equality problem, being obviously decidable, is not primitive recursive. They claim it to be the first example of an uncontrived problem with such complexity. We will show here that, restricted on r-finite nets, the problem for any reasonable behavioural equivalence is not primitive recursive either.

The proof will rely on a bounded version of the halting problem. Moreover, the technique can again be applied to the reachability set equality problem thus yielding a simpler proof of Mayr and Meyer's result (who used a bounded version of Hilbert's 10th problem).

Section 2 provides formal definitions and states the results. Section 3 explains the undecidability proof while Section 4 the nonprimitive recursivity proof. Some additional remarks are contained in Section 5.

2 Definitions and Results

$\mathcal{N}$ denotes the set of nonnegative integers, A^* the set of finite sequences of elements of A.

We are interested in behavioural equivalences based on actions which the systems under consideration can exhibit. In what follows, $\mathcal{A}$ stands for a fixed (countable) set of all possible *actions* or *action names*.

Now we provide necessary Petri net definitions; notice that actions from $\mathcal{A}$ are used to label transitions.

A *labelled place/transition marked net*, a *net* for short, is a tuple $N = (P, T, F, L, M_0)$

[1] Based on the author's note in *Bulletin of EATCS* **56**, June 1995

where P and T are finite disjoint sets of *places* and *transitions* respectively, $F : (P \times T) \cup (T \times P) \to \{0,1\}$ is a *flow function* ($F(x,y) = 1$ means that there is an *arc* from x to y), $L : T \to \mathcal{A}$ is a *labelling* and $M_0 : P \to \mathcal{N}$ is an *initial marking.* A *marking* M is a function $M : P \to \mathcal{N}$; it attaches a number of *tokens* to each place.

A transition t is *enabled* at a marking M, denoted by $M \xrightarrow{t}$, 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 $M \xrightarrow{t} M'$, where $M'(p) = M(p) - F(p,t) + F(t,p)$ for all $p \in P$. For any $a \in \mathcal{A}$, by $M \xrightarrow{a}$ ($M \xrightarrow{a} M'$) we mean that $M \xrightarrow{t}$ ($M \xrightarrow{t} M'$) for some t with $L(t) = a$. In the natural way, the definitions can be extended for finite sequences of transitions $\sigma \in T^*$ and finite sequences of actions $w \in \mathcal{A}^*$.

The *reachability tree* of a net $N = (P, T, F, L, M_0)$ is a (usually infinite) unordered tree whose vertices are labelled by markings, edges by transitions; the root is labelled by M_0, and any vertex labelled by M has precisely one immediate successor for each t, $M \xrightarrow{t}$; the corresponding edge is labelled by t and the successor by M' where $M \xrightarrow{t} M'$.

The *reachability set* of a net N is the set of all *reachable markings*, i.e. the markings which appear in the reachability tree.

A place p in a net N is *bounded* iff there is $k \in \mathcal{N}$ s.t. $M(p) \leq k$ for any reachable marking M; otherwise it is *unbounded.*

A *net* is *r-finite* iff its reachability set is finite.

We will also need the following notion taken from [3]. The *strict branching structure* of (the behaviour of) a net is the (unordered) tree arising from the reachability tree by relabelling each edge with $L(t)$ instead of t and omitting (forgetting) all vertex labellings.

We also refer to the notion of sequential nets (as defined e.g. in [12]). Two transitions t_1, t_2 can *fire concurrently* in M iff $M(p) \geq F(p,t_1) + F(p,t_2)$ for every $p \in P$. A *net* N is *sequential* iff no two transitions can fire concurrently in any reachable marking.

We use the term of a reasonable (action-based behavioural) equivalence, or more generally of a reasonable preorder, on the set of all nets. The following technical notion does not aim to capture the intuitive concept precisely, it just should comprise all intuitively reasonable equivalences and preorders (like bisimulation equivalence, simulation preorder, trace set inclusion etc.). In other words, the next two conditions could be viewed as axioms which any intuitively reasonable preorder should satisfy.

Definition 1 *A preorder R on the set of all nets is* reasonable *iff the following two conditions hold:*

(1) $(N_1, N_2) \in R$ for any two sequential nets N_1, N_2 with the same (i.e. isomorphic) strict branching structures.
(2) If a sequence $w \in \mathcal{A}^*$ is enabled in N_1 and not enabled in N_2 then $(N_1, N_2) \notin R$.

Remark 2 *We could easily define a 'true concurrency' semantics whose (intuitively reasonable) associated equivalence does not satisfy Condition 1 when omitting the word 'sequential'. But for sequential nets, 'true concurrency' semantics coincides with 'interleaving' semantics.*

The aim of this paper is to prove the next two theorems.

Theorem 3 *Any reasonable preorder is undecidable for labelled Petri nets.*

Theorem 4 *Any reasonable preorder is not primitive recursive for r-finite labelled Petri nets.*

3 Undecidability Proof

Here we explain a proof for Theorem 3. It is based on a reduction from the halting problem for Minsky counter machines.

A *2-counter machine C*, with nonnegative counters c_1, c_2, is a sequence of commands

$$\mathtt{1} : \mathtt{comm_1};\ \mathtt{2} : \mathtt{comm_2};\ \ldots\ldots;\ \mathtt{n} : \mathtt{comm_n}$$

where $\mathtt{comm_n}$ is a $\mathtt{HALT}$-command and $\mathtt{comm_i}$ $(\mathtt{i} = 1, 2, ..., \mathtt{n} - 1)$ are commands of the following two types (assuming $1 \le \mathtt{k}, \mathtt{k_1}, \mathtt{k_2} \le \mathtt{n}$, $1 \le \mathtt{j} \le 2$)

(1) $\mathtt{c_j} := \mathtt{c_j} + 1;\ \mathtt{goto}\ \mathtt{k}$,
(2) $\mathtt{if}\ \mathtt{c_j} = 0\ \mathtt{then}\ \mathtt{goto}\ \mathtt{k_1}\ \mathtt{else}\ (\mathtt{c_j} := \mathtt{c_j} - 1;\ \mathtt{goto}\ \mathtt{k_2})$.

We use the following well-known fact.

Proposition 5 *It is undecidable if a given 2-counter machine does halt on the zero input.*

The undecidability proof here relies on the next proposition.

Proposition 6 *There is an algorithm which, given any 2-counter machine C, constructs two sequential nets N_1^C, N_2^C (with at most two unbounded places) s.t. the following two conditions hold:*

(1) *If C halts on the zero input then there is a sequence of actions which is enabled in N_1^C and not in N_2^C,*
(2) *If C does not halt on the zero input then the strict branching structures of N_1^C, N_2^C are the same.*

Observe that Theorem 3 immediately follows from Propositions 5 and 6.

Remark 7 *We could more generally say that the set of all pairs (N_1, N_2) of sequential nets with the same strict branching structures (and with at most 2 unbounded places) is recursively inseparable from the set of all pairs (N_1, N_2) of sequential nets (with at most 2 unbounded places) s.t. there is a sequence w of actions which is enabled in N_1 and not in N_2.*

In fact, the construction proving Proposition 6 is contained in [6]. For completeness, we now provide its concise description (first informally and then more precisely). The construction uses four transition labels: **i** (increasing), **d** (decreasing), **z** (zero), **h** (halt).

We start with a straightforward construction of a net 'weakly' simulating the given machine C (the net can not test for zero, so the transitions labelled **z** can 'cheat').

Then we add special places p, p' and two copies of each **z**-transition; the copies can fire only 'cheatingly' (if the relevant counter is nonzero) and, in addition, one copy also moves a token from p to p' and the other copy from p' to p. There will be also a special 'halting transition' which is enabled iff there is a token in the 'halting place' and a token in p.

Finally we provide two nets, one starting with a token in p, the other with a token in p'.

Construction of N_1^C and N_2^C

(1) First we construct a net N^C. We take c_1, c_2 (the counter part), $s_1, s_2, ..., s_n$ (the state part, n being the number of instructions of C) and additional p, p' as its places.
(2) For $i = 1, 2, ..., n-1$ we add transitions and arcs depending on the type of $\mathtt{comm_i}$:
   (a) if $\mathtt{comm_i}$ is $(\mathtt{c_j} := \mathtt{c_j} + 1;\ \mathtt{goto\ k})$ then we add a transition t_i labelled with **i** (increasing), and arcs (s_i, t_i), (t_i, c_j), (t_i, s_k),
   (b) if $\mathtt{comm_i}$ is $(\mathtt{if\ c_j = 0\ then\ goto\ k_1\ else\ (c_j := c_j - 1;\ goto\ k_2)})$ then add t_i^{NZ} (for the 'non-zero case') labelled with **d** (decreasing), and arcs (s_i, t_i^{NZ}), (c_j, t_i^{NZ}), (t_i^{NZ}, s_{k_2}), and three transitions t_i^Z, t_i', t_i'' labelled with **z** (zero), and the arcs (s_i, t), (t, s_{k_1}) for each $t \in \{t_i^Z, t_i', t_i''\}$; we also add the arcs (c_j, t_i'), (t_i', c_j), (c_j, t_i''), (t_i'', c_j), and the arcs (p, t_i'), (t_i', p'), (p', t_i''), (t_i'', p).

(3) We add a further transition t^H labelled with **h** (halt) and the arcs (s_n, t^H), (p, t^H); thus the construction of N^C is finished.
(4) Finally, we take two copies of N^C. In one copy we put 1 token in s_1 and 1 token in p (0 elsewhere), which yields N_1^C; in the other copy we put 1 token in s_1 and 1 token in p' (0 elsewhere), which yields N_2^C.

Notice that only c_1, c_2 are (possibly) unbounded. The nets are obviously sequential: in all reachable markings, there is at most 1 token in the set of places $\{s_1, s_2, \ldots, s_n\}$ and any transition has an s_i among its input places; hence no two transitions can fire concurrently.

It remains to verify the conditions 1 and 2 in Proposition 6.

The sequence of actions mentioned in 1 is the label sequence corresponding to the transition sequence which correctly simulates C and finishes by t^H (see [6] for more details).

For the condition 2, [6] only claims (the weaker condition of) bisimilarity of N_1^C and N_2^C. Nevertheless the stronger condition (equality of the strict branching structures) is easily verifiable:

Suppose that trees $\mathcal{T}_1, \mathcal{T}_2$ are the strict branching structures of N_1^C, N_2^C respectively. We define the isomorphism $f : \mathcal{T}_1 \to \mathcal{T}_2$ as follows:

First, in both $\mathcal{T}_1$ and $\mathcal{T}_2$ there is exactly one infinite path corresponding to the correct simulation of C (recall that C does not halt); let f be the straightforward bijection on these infinite paths.

Now take any two vertices v_1, v_2 s.t. $f(v_1) = v_2$ (v_1 being on the 'correct path' in $\mathcal{T}_1$, v_2 on the 'correct path' in $\mathcal{T}_2$). Observe that the markings M_1, M_2 'underlying' v_1, v_2 (in the respective reachability tree) differ just on p, p' and they have no token in s_n. Surely, v_1 and v_2 have the same numbers of successors, either one or three. In the latter case, there is one **d**-successor (this is the 'correct one') and two **z**-successors. Obviously, the set of the two markings underlying the **z**-successors of v_1 is the same as the set of the two markings underlying the **z**-successors of v_2. Therefore, the extension of f is straightforward.

Remark 8 *For proving undecidability of reachability set equality (as well as containment) problem, [6] uses a modification of N_1^C, N_2^C where a certain 'coding subnet' is added, yielding nets with 5 unbounded places.*

4 Nonprimitive Recursivity Proof

Here we explain a proof for Theorem 4.

Let us consider the family of functions $A_i : \mathcal{N} \to \mathcal{N}$, $i = 0, 1, 2, \ldots$, and the function $A : \mathcal{N} \to \mathcal{N}$ defined as follows:

- $A_0(x) = 2x + 1$
- $A_{n+1}(0) = 1$
- $A_{n+1}(x + 1) = A_n(A_{n+1}(x))$
- $A(n) = A_n(2)$

It is not difficult to show that there is a constant $c \in \mathcal{N}$ s.t. for any n a sequential net WC_n (a 'weak computer') can be constructed, in time polynomial in n, so that

- WC_n has less than $c \cdot n$ places, transitions and arcs, and just 1 token in a ('starting') place,
- WC_n has two special places *off* and *out*: any computation (path in the reachability tree) is finite, finishing exactly when putting a token in *off*; the set of possible values of *out* in such final markings is precisely the set $\{0, 1, 2, \ldots, A(n)\}$.

This can be found e.g. in [10], as well as the reference to the fact that the function A dominates any primitive recursive function (i.e. for any primitive recursive function f there is n_0 s.t. $A(n) > f(n)$ for all $n \geq n_0$).

We will use the following bounded version of the halting problem; its nonprimitive recursivity can be shown by standard methods of computability theory.

Proposition 9 *The problem to decide, given a 2-counter machine C and $n \in \mathcal{N}$, if C halts on the zero input in $A(n)$ steps is nonprimitive recursive.*

Theorem 4 is an immediate consequence of Proposition 9 and the next proposition 10.

Proposition 10 *There is a polynomial-time algorithm which, given any 2-counter machine C and $n \in \mathcal{N}$, constructs two sequential r-finite nets $N_1^{C,n}$, $N_2^{C,n}$ s.t. the following two conditions hold:*

(1) If C halts on the zero input within $A(n)$ steps then there is a sequence of actions which is enabled in $N_1^{C,n}$ and not in $N_2^{C,n}$,
(2) If C does not halt on the zero input within $A(n)$ steps then the strict branching structures of $N_1^{C,n}$, $N_2^{C,n}$ are the same.

Given C and n, we construct the disjoint union of the net N^C (from Section 3) and the net WC_n; all transitions of WC_n will be labelled by $\mathbf{b}$ (beginning).

We add a new transition t^S, labelled by $\mathbf{b}$ as well, and the arcs (off, t^S), (t^S, s_1) (t^S starts N^C after WC_n has finished its computing).

Then for any transition t in N^C, we add the arc (out, t) (thus N^C cannot fire a sequence longer than $A(n)$).

Finally, we take two copies of the resulting net. In one we put a token in p, thus getting $N_1^{C,n}$, and in the other we put a token in p', thus getting $N_2^{C,n}$.

Verification of Proposition 10 is now straightforward.

Remark 11 *The same idea can be applied for the reachability set equality (or containment) problem; then only the 'basic' transitions, not that belonging to the 'coding part' (cf. Remark 8), take a token from the place 'out'. Thus a more direct proof of the result in [10] can be provided.*

5 Additional Remarks

Recall that for the undecidability of behavioural equivalences we needed nets with two unbounded places.

For the case of nets with one unbounded place, e.g. bisimulation equivalence is decidable [7] as well as simulation preorder [1]. Another decidable subclass is obtained by considering deterministic nets only—e.g. those with one-to-one transition labellings (this can be also found in [6]).

As also mentioned in [10], the problem of r-finiteness of a given net can be decided in exponential space [11] and therefore lies 'deeply' inside the class of primitive recursive problems.

References

[1] P. Abdulla and K. Čerāns, Simulation is decidable for one-counter nets, in Proc. Concur'98, *Lecture Notes in Computer Science* **1466** (Springer 1998) 253–268.

[2] J. Esparza und M. Nielsen, Decidability Issues for Petri Nets – a survey, *Journal of Information Processing and Cybernetics* **30**(3) (1994) 143–160.

[3] R.J. van Glabbeek, What is branching time semantics and why to use it ?, *Bull. of EATCS*, **53** (June 1994) 191–198.

[4] M. Hack, Decision problems for Petri nets and vector addition systems, MAC Techn. Memo 53, MIT, 1975.

[5] M. Hack, The equality problem for vector addition systems is undecidable, *Theoretical Computer Science* **2** (1976) 77–95.

[6] P. Jančar, Undecidability of bisimilarity for Petri nets and related problems, *Theoretical Computer Science* **148** (1995) 281–301.

[7] P. Jančar, Bisimulation equivalence is decidable for one-counter processes, in Proc. ICALP'97, *Lecture Notes in Computer Science* **1256** (Springer 1997) 549–559 (to appear in Information and Computation).

[8] F. Moller, Infinite results, in Proc. Concur'96, *Lecture Notes in Computer Science* **1119** (Springer 1996) 195–216.

[9] R. Milner, *Communication and Concurrency* (Prentice Hall, 1989).

[10] E.W. Mayr and A.R. Meyer, Finite containment problem for Petri nets, *Journal of the ACM* **28** (1981) 561–576.

[11] C. Rackoff, The covering and boundedness problems for vector addition systems, *Theoretical Computer Science* **6** (1978) 223–231.

[12] W. Vogler, *Modular Construction and Partial Order Semantics of Petri Nets*, Lecture Notes in Computer Science **625** (Springer, 1992).