On the decidability of model checking for several μ-calculi and Petri nets

Javier Esparza
Laboratory for Foundations of Computer Science
University of Edinburgh
King's Buildings
Edinburgh EH9 3JZ
e-mail: je@dcs.ed.ac.uk

Abstract

The decidability of the model checking problem for several μ-calculi and Petri nets is analysed. The linear time μ-calculus is decidable; if simple atomic sentences are added, it becomes undecidable. A very simple subset of the modal μ-calculus is undecidable.

1 Introduction

Research on decidability issues for Petri nets (or vector addition systems, a closely related model) has a long tradition. Two milestones are the introduction by Karp and Miller of coverability graphs [9] – which can be used to prove, among other properties, the decidability of boundedness – and the proof by Mayr and Kosaraju of the decidability of the reachability problem [12, 10]. A natural next step is to examine the decidability of the model checking problem for temporal logics able to express a large class of properties. This problem was first studied by Howell and Rosier in [6]; they observed that a linear time temporal logic interpreted on the infinite occurrence sequences of the net is undecidable even for conflict-free Petri nets, a fairly small class. In a subsequent paper [7], Howell, Rosier and Yen showed that the model checking problem for a fragment of this logic can be reduced to the reachability problem, and is thus decidable. This fragment consists of a set of atomic sentences, which can be combined using boolean connectives, and the operator **F** (eventually). The atomic sentences are of type $ge(p, c)$ (with intended meaning 'the number of tokens on place p is greater than or equal to c') or $fi(t)$ (with intended meaning 'transition t is the next one in the sequence')[1]. Jančar showed in [8] that for the logic with these basic

[1]Actually, the fragment is somewhat larger, but a detailed description is out of the scope of this paper.

predicates and the operator **GF** (always eventually), the model checking problem is decidable as well.
We analyse in this paper the decidability of several μ-calculi, logics with fixpoint operators [15]. Our three results are:

- The linear time μ-calculus is decidable.
- The linear time μ-calculus extended with atomic sentences of the form $A = 0$ – meaning 'the place A contains 0 tokens' – is undecidable, even for formulas with one single fixpoint and a very simple class of Petri nets in which every transition has at most one input and at most one output place.
- The modal μ-calculus is undecidable, even for formulas with one single fixpoint and the same class of Petri nets.

The third result (which is a small modification of Theorem 5.4 in [1]) is not very surprising, because the modal μ-calculus is known to be a extremely powerful logic, which properly contains many branching time logics such as PDL, CTL or CTL*. Since the decidability of branching time logics seems not to have been explored so far, we complement the paper with a fourth result, unfortunately negative: a rather weak branching time subset of the modal μ-calculus, which extends propositional logic with possibility operators, is undecidable.
The note is organised as follows: sections 2 and 3 contain basic definitions about Petri nets and the linear time μ-calculus, respectively. Sections 4 and 5 prove the first and second results above. Section 6 proves the other two.

2 Petri nets

A labelled net N is a fourtuple (S, T, F, l), where

- S and T are two disjoint, finite sets,
- F is a relation on $S \cup T$ such that $F \cap (S \times S) = F \cap (T \times T) = \emptyset$, and
- l is a surjective mapping $T \rightarrow Act$, where Act is a set of actions (surjectivity is assumed for convenience).

The elements of S and T are called places and transitions, respectively. Places and transitions are generically called nodes.
Given a node x of N, ${}^\bullet x = \{y \mid (y, x) \in F\}$ is the preset of x and $x^\bullet = \{y \mid (x, y) \in F\}$ is the postset of x.
Given a set of nodes X of N, we define ${}^\bullet X = \bigcup_{x \in X} {}^\bullet x$ and $X^\bullet = \bigcup_{x \in X} x^\bullet$.

A marking of a net $N = (S, T, F)$ is a mapping $M: S \to I\!N$. A marking M enables a transition t if it marks every place in ${}^\bullet t$. If t is enabled at M, then its occurrence leads to the successor marking M' which is defined for every place s by

$$M'(s) = \begin{cases} M(s) & \text{if} \quad s \notin {}^\bullet t \text{ and } s \notin t^\bullet \quad \text{or} \quad s \in {}^\bullet t \text{ and } s \in t^\bullet \\ M(s) - 1 & \text{if} \quad s \in {}^\bullet t \text{ and } s \notin t^\bullet \\ M(s) + 1 & \text{if} \quad s \notin {}^\bullet t \text{ and } s \in t^\bullet \end{cases}$$

(a token is removed from each place in the pre-set of t and a token is added to each place in the post-set of t).

A marking M is called dead if it enables no transition of N.

A labelled Petri net is a pair $\Sigma = (N, M_0)$ where N is a labelled net and M_0 is a marking of N. $M_1 \xrightarrow{a} M_2$, where M_1, M_2 are markings of N, denotes that M_1 enables some transition t labelled by a, and that the marking reached by the occurrence of t is M_2. A sequence $M_0 \xrightarrow{a_1} M_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} M_n$ is a finite occurrence sequence leading from M to M_n and we write $M_0 \xrightarrow{a_1 \dots a_n} M_n$. The empty sequence ϵ is an occurrence sequence: we have $M \xrightarrow{\epsilon} M$ for every marking M.
We assume that if two transitions have the same label, then they do not have the same preset and postset. Under this assumption, every label of an occurrence sequence can be uniquely mapped to its underlying transition. Making implicit use of this property, we sometimes speak about the transitions that occur in an occurrence sequence.

A sequence $M_0 \xrightarrow{a_1} M_1 \xrightarrow{a_2} \cdots$ is an infinite occurrence sequence. We write $M \xrightarrow{a_1\, a_2 \dots}$.
A sequence of actions σ is enabled at a marking M if $M \xrightarrow{\sigma} M'$ for some marking M' (if σ is finite) or $M \xrightarrow{\sigma}$ (if σ is infinite).

An occurrence sequence is maximal if either it is infinite or it leads to a dead marking. The language of Σ, denoted by $L(\Sigma)$, is the set of words obtained dropping the intermediate markings in the maximal occurrence sequences of Σ.

3 The linear time μ-calculus

The linear time μ-calculus has the following syntax:

$$\phi ::= Z \mid \neg\phi \mid \phi \wedge \phi \mid O_a\phi \mid \nu Z.\phi$$

where a ranges over a set Act of a actions, and Z over propositional variables.

Formulas are built out of this grammar, subject to the monotonicity condition that all free occurrences of X lie in the scope of an even number of negations.

Let $Act^\star$, Act^ω be the set of finite and infinite words on Act, and let $Act^\infty = Act^\star \cup Act^\omega$. A valuation $\mathcal{V}$ of the logic assigns to each variable X a set of words $\mathcal{V}(X)$ in Act^∞. We denote by $\mathcal{V}[A/Z]$ the valuation $\mathcal{V}'$ which agrees with $\mathcal{V}$ except on Z where $\mathcal{V}'(Z) = A$. Given a word $\sigma = a_1\, a_2 \dots$ on Act^∞, $\sigma(1)$ denotes the first action of σ, i.e., a_1, and σ^1 is the word $a_2\, a_3 \ldots$. With these notations, the denotation $\|\phi\|_{\mathcal{V}}$ of a formula ϕ is the set of words of Act^∞ inductively defined by the following rules:

$$
\begin{array}{rcl}
\|Z\|_{\mathcal{V}} & = & \mathcal{V}(Z) \\
\|\neg\phi\|_{\mathcal{V}} & = & Act^\infty - \|\phi\|_{\mathcal{V}} \\
\|\phi \wedge \psi\|_{\mathcal{V}} & = & \|\phi\|_{\mathcal{V}} \cap \|\psi\|_{\mathcal{V}} \\
\|O_a\phi\|_{\mathcal{V}} & = & \{\sigma \in Act^\infty \mid \sigma(1) = a \wedge \sigma^1 \in \|\phi\|_{\mathcal{V}}\} \\
\|\nu Z.\phi\|_{\mathcal{V}} & = & \cup\{A \subseteq Act^\infty \mid A \subseteq \|\phi\|_{\mathcal{V}[A/Z]}\}
\end{array}
$$

Observe that the denotation of a closed formula ϕ is independent of the valuation; we then use the symbol $\|\phi\|$. Let Σ be a labelled Petri net and ϕ a closed formula of the linear time μ-calculus. We say that Σ satisfies ϕ if $L(\Sigma) \subseteq \|\phi\|$. Notice that, with this definition of satisfaction, it can be the case that Σ satisfies neither a formula nor its negation.

4 Decidability of the linear time μ-calculus

The model checking problem for the linear time μ-calculus consists in deciding, given a Petri net Σ and a closed formula ϕ, if Σ satisfies ϕ or not. We prove in this section that this problem is decidable. The decision procedure is based on an automata-theoretic characterisation of the logic [16]. We show that there exist two automata $A_{\neg\phi}$, $B_{\neg\phi}$ that accept the finite and infinite words of $\|\neg\phi\|$, respectively. It follows that Σ satisfies ϕ if and only if $L(\Sigma) \cap L(A_{\neg\phi}) = \emptyset$ and $L(\Sigma) \cap L(B_{\neg\phi}) = \emptyset$. To decide these two properties we construct Petri net representations of $A_{\neg\phi}$ and $B_{\neg\phi}$, and combine them with Σ in a way similar to the product of automata; it is then easy to show that the two properties are equivalent to two decidable properties of Petri nets.

An automaton over an alphabet Act is a fourtuple $(Q, q_0, \delta, \mathcal{F})$, where Q is the set of states, q_0 the initial state, $\delta\colon Q \times Act \to Q$ the transition function and $\mathcal{F}$ the set of final states. Finite and Büchi automata are automata with different acceptance conditions: a finite automaton A accepts a word w in $Act^\star$ if the computation of A on w ends in some final state; a Büchi automaton B accepts a word w in Act^ω if the computation of B on w passes infinitely often through some final state.

Dam provides in [3] a procedure to construct, given a formula ϕ of a linear time μ-calculus different from ours, a Büchi automaton whose language is the denotation of the formula. Dam's version of the linear time μ-calculus has one single next operator O, instead of an operator O_a for every action a. Given a finite set V of propositional variables, the denotation of a formula with free variables in V is a set of words over the alphabet 2^V. The rules defining the denotation of a formula are like the ones defined above for $\neg$, $\wedge$ and greatest fixpoints, plus the following rule for the new next operator:

$$\|O\phi\|_{\mathcal{V}} \;=\; \{\sigma \in Act^\infty \mid \sigma^1 \in \|\phi\|_{\mathcal{V}}\}$$

We briefly discuss how to adapt Dam's construction to our case. Dam's automaton has as alphabet $\mathcal{P}(V) \times \mathcal{P}(V)$, where $\mathcal{P}(V)$ is the powerset of V; the automaton is constructed in a compositional way. A word w is accepted by the automaton if and only if there is an accepting run

$$q_0 \xrightarrow{(\alpha_1^+,\alpha_1^-)} q_1 \xrightarrow{(\alpha_2^+,\alpha_2^-)} \ldots$$

such that at the ith point of w all variables in α_i^+ hold and no variable in α_i^- holds. For our version of the linear time μ-calculus, we enrich the alphabet of the automaton to $\mathcal{P}(V) \times \mathcal{P}(V) \times (Act \cup \{\tau\})$, where $\tau \notin Act$. Dam's construction can now be easily modified to take this third component of the alphabet into account. A word w is accepted by the new automaton if and only if there is an accepting run

$$q_0 \xrightarrow{(\alpha_1^+,\alpha_1^-,a_1)} q_1 \xrightarrow{(\alpha_2^+,\alpha_2^-,a_2)} \ldots$$

such that at the ith point of w all variables in α_i^+ hold, no variable in α_i^- holds, and the third component of w at this point is a_i.
If ϕ is a closed formula, we choose $V = \emptyset$. In this case, the automaton we obtain has just $Act \cup \{\tau\}$ as alphabet. The rôle of the action τ is the following: in our case, the Büchi automaton does not accept exactly $\|\phi\|$, because $\|\phi\|$ may contain finite words (for instance, $a \in \|O_a tt\|$), while a Büchi automaton only accepts infinite ones. If $\|\phi\|$ contains a finite word w, then the automaton accepts $w\,\tau^\omega$. τ only appears in transitions of the form (q, τ, q).

Proposition 4.1

Let ϕ be a closed formula of the linear time μ-calculus over a a set of actions Act. There exist a Büchi automaton B with alphabet $Act \cup \{\tau\}$ which accepts a word w iff $w \in \|\phi\| \cap Act^\omega$ or $w = w'\tau^\omega$ and $w' \in \|\phi\| \cap Act^\star$.

Proof:
Use Dam's construction with the modifications discussed above. ■

It is convenient for our analysis to split the automaton B of Proposition 4.1 into a finite automaton which accepts the finite words of $\|\phi\|$ and a Büchi automaton which accepts the infinite ones.

Proposition 4.2

Let ϕ be a closed formula of the linear time μ-calculus over a set of actions Act. There exist a finite automaton A_ϕ and a Büchi automaton B_ϕ, both with alphabet Act, such that $L(A_\phi) = \|\phi\| \cap Act^\star$, and $L(B_\phi) = \|\phi\| \cap Act^\omega$.

Proof:
A_ϕ and B_ϕ are the same automaton (i.e., they differ only in the acceptance condition), obtained by just removing the transitions of the form (q, τ, q) from δ. ■

We now assign to a given automaton a Petri net. Let $A = (Q, q_0, \delta, \mathcal{F})$ be an automaton over an alphabet Act. The labeled Petri net $\Sigma^A = (S, T, F, M_0, l)$ – with labels on Act – is defined as follows:

$$
\begin{aligned}
S &= Q \\
T &= \delta \\
F &= \{(q, (q, a, q')), ((q, a, q'), q') \mid (q, a, q') \in \delta\} \\
M_0(q) &= \begin{cases} 1 & \text{if } q = q_0 \\ 0 & \text{otherwise} \end{cases} \\
l((q, a, q')) &= a
\end{aligned}
$$

It follows immediately from this construction that the finite automaton A_ϕ accepts a word σ if and only if σ is an occurrence sequence of the Petri net Σ^{A_ϕ} leading to a marking in which some final state is marked (recall that the places of Σ^{A_ϕ} are the states of A). Similarly, B_ϕ accepts a word σ if and only if σ is an infinite occurrence sequence such that infinitely many intermediate markings along the sequence put one token in some of the final states of B_ϕ. If $T_\mathcal{F}$ is the set of input transitions of these states, then an equivalent condition is that σ contains infinitely many occurrences of some transition of $T_\mathcal{F}$.

Let $\Sigma_1 = (S_1, T_1, F_1, l_1, M_{01})$, $\Sigma_2 = (S_2, T_2, F_2, l_2, M_{02})$ be two labelled Petri nets with disjoint sets of nodes. The labelled Petri net $\Sigma_1 \times \Sigma_2 = (S, T, F, l, M_0)$ is defined as follows:

$$
\begin{aligned}
S &= S_1 \cup S_2 \\
T &= \{\{t_1, t_2\} \mid t_1 \in T_1 \wedge t_2 \in T_2 \wedge l_1(t_1) = l_2(t_2)\} \\
F &= (S \times T) \cap \{(s, \{t_1, t_2\}) \mid (s, t_1) \in F_1 \vee (s, t_2) \in F_2\} \\
&\cup\ (S \times T) \cap \{(\{t_1, t_2\}, s) \mid (t_1, s) \in F_1 \vee (t_2, s) \in F_2\} \\
M_0(s) &= \begin{cases} M_{01}(s) & \text{if } s \in S_1 \\ M_{02}(s) & \text{if } s \in S_2 \end{cases} \\
l(\{t_1, t_2\}) &= l_1(t_1)
\end{aligned}
$$

Observe that this product operation is very similar to the usual product of automata. Every marking M of $\Sigma_1 \times \Sigma_2$ projects onto a marking M_1 of Σ_1 and a marking M_2 of Σ_2; moreover, these two projections determine M. Making use of this property, we denote $M = (M_1, M_2)$. So, in particular, $M_0 = (M_{01}, M_{02})$.

The following lemma is an immediate consequence of the definitions and the occurrence rule for Petri nets.

Lemma 4.3

$(M_1, M_2) \xrightarrow{\sigma} (M_1', M_2')$ iff $M_1 \xrightarrow{\sigma} M_1'$ and $M_2 \xrightarrow{\sigma} M_2'$. (where M_1, M_1' are markings of Σ_1 and M_2, M_2' markings of σ_2)

Proof:
($\Rightarrow$): Let $\{t_1, u_1\}\ \{t_2, u_2\} \ldots$ be the sequence of transitions of $\Sigma_1 \times \Sigma_2$ underlying σ. Then, $t_1\ t_2 \ldots$ and $u_1\ u_2 \ldots$ are sequences of transitions underlying the occurrence sequence σ in Σ_1 and Σ_2, respectively.
($\supseteq$): Similar to $\subseteq$. ■

We can now reduce $L(\Sigma) \cap L(B_\phi) \neq \emptyset$ to a property of the Petri net $\Sigma \times \Sigma^{B_\phi}$.

Lemma 4.4

$L(\Sigma) \cap L(B_\phi) \neq \emptyset$ iff $\Sigma \times \Sigma^{B_\phi}$ has an occurrence sequence which contains infinitely many occurrences of some transition $\{t, u\}$, where $u \in T_{\mathcal{F}}$.

Proof:
($\Rightarrow$): Let σ be a word of $L(\Sigma) \cap L(B_\phi)$. Then σ is the sequence of labels of two occurrence sequences in Σ and Σ^{B_ϕ}; moreover, σ contains infinitely many occurrences of some transition u of $T_{\mathcal{F}}$. By Lemma 4.3, σ is the sequence of labels of an occurrence sequence in $\Sigma \times \Sigma^{B_\phi}$, in which transitions of the form $\{t, u\}$ occur infinitely often.
($\Leftarrow$): Let $(M_{01}, M_{02}) \xrightarrow{\sigma}$ be an infinite occurrence sequence of $\Sigma \times \Sigma^{B_\phi}$ which contains infinitely many occurrences of of some transition $\{t, u\}$, where $u \in T_{\mathcal{F}}$. By Lemma 4.3, $M_{02} \xrightarrow{\sigma}$ is an occurrence sequence of Σ^{B_ϕ}. Let $M_{02} \xrightarrow{a_1} M_1 \xrightarrow{a_2} M_2 \xrightarrow{a_3} \ldots$ be the full representation of $M_{02} \xrightarrow{\sigma}$. By the definition of Σ^{B_ϕ}, each marking M_i puts one token in exactly one place of Σ^{B_ϕ}, and no tokens in the rest. So each marking M_i is unequivocally associated to a state of B_ϕ, and the whole sequence to a computation. The state associated to the markings succeeding an occurrence of $\{t, u\}$ is a final state of B_ϕ; so B_ϕ accepts σ. ■

The condition on $\Sigma \times \Sigma^{B_\phi}$ of Lemma 4.4 was shown to be decidable by Jantzen and Valk [4]. Yen shows in [17] that it is decidable within exponential space.

Lemma 4.5 *[17]*

Let Σ be a Petri net with n nodes, and let T be a set of transitions of Σ. It can be decided in $O(2^{c \cdot n \cdot logn})$ space, for some constant c, if some infinite occurrence sequence of Σ contains infinitely many occurrences of some transition of T. ■

Proof:
There exists such an infinite occurrence sequence iff there exists a finite occurrence sequence $M_0 \xrightarrow{\sigma_1} M_1 \xrightarrow{\sigma_2} M_2$ such that $M_1 \leq M_2$ and σ_2 contains some occurrence of transition of T (then M_1 enables the sequence σ_2^ω). This can be expressed as a formula on paths of Petri nets which belongs to the class of formulas defined by Yen in [17]. Yen shows that any of these formulas can be decided using the space indicated in the lemma. ■

$L(\Sigma) \cap L(A_\phi) \neq \emptyset$ can also be reduced to a suitable property of $\Sigma \times \Sigma^{A_\phi}$.

Lemma 4.6

$L(\Sigma) \cap L(A_\phi) \neq \emptyset$ iff there exists a reachable dead marking of $\Sigma \times \Sigma^{A_\phi}$ which puts one token in some final state of A_ϕ.

Proof:
($\Rightarrow$): Let σ be a word of $L(\Sigma) \cap L(A_\phi)$. Then, $M_{01} \xrightarrow{\sigma} M_1$ in Σ and $M_{02} \xrightarrow{\sigma} M_2$ in Σ^{A_ϕ}; moreover, M_1 is a dead marking of Σ, and M_2 puts a token in some final state of A_ϕ. By Lemma 4.3, $(M_{01}, M_{02}) \xrightarrow{\sigma} (M_1, M_2)$ is an occurrence sequence of $\Sigma \times \Sigma^{A_\phi}$. Since M_1 is a dead marking of Σ, (M_1, M_2) is a dead marking of $\Sigma \times \Sigma^{A_\phi}$.
($\Leftarrow$): Let $(M_{01}, M_{02}) \xrightarrow{\sigma} (M_1, M_2)$ be an occurrence sequence of Σ such that (M_1, M_2) is a dead marking, and M_2 puts a token in some final state of A_ϕ. By Lemma 4.3, $M_{02} \xrightarrow{\sigma} M_2$ is an occurrence sequence of Σ^{A_ϕ}, and therefore $\sigma \in L(A_\phi)$.
It remains to prove that σ belongs to the language of Σ, i.e., that $M_{01} \xrightarrow{\sigma} M_1$ is a maximal occurrence sequence of Σ.
Assume this is not the case. Then there exists an occurrence sequence $M_{01} \xrightarrow{\sigma} M_1 \xrightarrow{a} M_1'$. Let t be the transition underlying the occurrence of a.
By the definition of A_ϕ, a belongs to the alphabet of A_ϕ, and therefore to the alphabet of Σ^{A_ϕ} as well. So Σ^{A_ϕ} contains a transition (q, a, q'), where q is the final state marked at M_2. This transition is enabled at M_2. By Lemma 4.3, $\{t, (q, a, q')\}$ is enabled at (M_1, M_2), which contradicts that (M_1, M_2) is a dead marking. ■

The existence of the dead marking of lemma 4.6 can be decided by solving an exponential number of instances of the *submarking reachability problem* [5]. Given a net with a set of places S, a *submarking* of the net is a *partial* mapping from S onto $I\!N$. The submarking reachability problem is the problem of deciding, given a Petri net $\Sigma = (N, M_0)$ and a submarking P of N, if some reachable marking coincides with P on all the places where P is defined. The submarking reachability problem is reducible (in polynomial time) to the reachability problem [5].

Lemma 4.7

Let Σ be a Petri net, and let S be a subset of places of Σ. It is decidable if some dead marking of Σ puts a token on some place of S.

Proof:
If M is a dead marking, then for every transition t some input place of t is unmarked at M. The set of submarkings which specify that, for every transition t, some input place of t is unmarked, and moreover that some place of S contains one token, is finite: it contains at most $2^{|T|} \cdot |S|$ elements, where T is the set of transitions of the net. The property of the lemma can then be decided by solving the submarking reachability problem for these sets. ■

The complexity of the reachability problem is still open. The most efficient algorithm is not primitive recursive, while the best known lower bound is exponential space [11]. If exponential space suffices, then the complexities of deciding $L(\Sigma) \cap L(B_\phi) = \emptyset$ and $L(\Sigma) \cap L(A_\phi) = \emptyset$ are similar. Otherwise, deciding the second property is more involved.

Theorem 4.8

Let Σ be a labelled Petri net, and let ϕ be a closed formula of the linear time μ-calculus. It is decidable if Σ satisfies ϕ.

Proof:
Σ satisfies ϕ iff $L(\Sigma) \subseteq \|\phi\|$, or, by the semantics of negation, iff $L(\Sigma) \cap \|\neg\phi\| = \emptyset$. By Proposition 4.2, $L(\Sigma) \cap \|\neg\phi\| = \emptyset$ iff $L(\Sigma) \cap L(A_{\neg\phi}) = \emptyset$ and $L(\Sigma) \cap L(B_{\neg\phi}) = \emptyset$. Use then Lemmas 4.4, 4.5, 4.6 and 4.7. ■

5 Adding atomic sentences

It is well known that Petri nets have less computing power than Turing machines. In particular, it is impossible to construct a Petri net model of a counter. Crudely speaking, the reason why is that Petri nets cannot 'test for zero', meaning that there is no way to enable a transition exactly when a place is *not* marked. If Petri nets are enriched with inhibitor arcs, which add this functionality, they become Turing powerful [14].

Instead of enriching Petri nets, we can enrich the linear time μ-calculus, and allow it to 'test for zero': it suffices to supplement the logic with atomic sentences of the form $A = 0$, meaning 'the place A contains no tokens at the current marking'. We show that this addition makes the model checking problem undecidable, even for formulas with one single fixpoint and a very small class of unlabelled Petri nets. The proof is by reduction from the halting problem for register machines; it is a modification of Bradfield's construction in [1].

Let us start by giving the semantics of the extended logic. We interpret the sentence $A = 0$ on Petri nets $\Sigma = (N, M_0)$ having a place A. We then extend valuations in the following way:

$$\mathcal{V}(A = 0) = \{\sigma \in L((N, M)) \mid M(A) = 0\}$$

A register machine $\mathcal{R}$ is a tuple

$$(\{q_0, \ldots, q_{n+1}\}, \{R_1, \ldots, R_m\}, \{\delta_0, \ldots, \delta_{n-1}\})$$

where R_i are the registers, q_i are the states with q_0 being the initial state and q_{n+1} the unique halting state, and δ_i is the transition rule for state q_i ($0 \leq i \leq n$): δ_i is either (i) '$R_j := R_j + 1$; goto q_k' for some j, k, or (ii) 'if $R_j = 0$' then goto q_k else ($R_j := R_j - 1$; goto $q_{k'}$)' for some j, k, k'.

The halting problem for register machines is defined thus: given a register machine $\mathcal{R}$ and a set $v_1, \ldots, v_m$ of nonnegative integers, to decide if the computation of $\mathcal{R}$ with the registers initialised to $v_1, \ldots, v_m$ ever reaches the halting state. This problem is known to be undecidable [13].

We define a labelled net $N_{\mathcal{R}}$ as follows: the places of $N_{\mathcal{R}}$ are $q_0, \ldots, q_n, R_1, \ldots, R_m$. For every register R_i the net contains two transitions inc_i, dec_i, such that ${}^{\bullet}inc_i =$

$dec_i = \emptyset$ and $inc_i^\bullet = {}^\bullet dec^\bullet = \{R_i\}$. The rest of the transitions and the flow relation are determined by the δ_i: if δ_i is of the form (i), then there is a transition δ_i^+ with ${}^\bullet\delta_i^+ = \{q_i\}$ and $\delta_i^{+\bullet} = \{q_k\}$; and if δ_i is of form (ii), then there are transitions δ_i^0 and δ_i^- such that ${}^\bullet\delta_i^0 = {}^\bullet\delta_i^- = \{q_i\}$, $\delta_i^{0\bullet} = \{q_k\}$ and $\delta_i^{-\bullet} = \{q_{k'}\}$. Finally, there is a transition $halt$, such that ${}^\bullet halt = \{q_n\}$ and $halt^\bullet = \emptyset$.
Observe that every transition of $N_{\mathcal{R}}$ has at most one input place and at most one output place.

Define the formula $halt(\mathcal{R})$ as:

$$\begin{aligned}\mu Z.\quad & O_{halt}\, tt \\ & \bigvee_{\substack{i=0\ldots n \\ j=1\ldots m}} \quad (O_{\delta_i^+} O_{inc_j} Z \ \vee \ O_{\delta_i^-} O_{dec_j} Z) \\ & \bigvee_{j=1\ldots m} \quad (R_j = 0 \ \wedge \ O_{\delta_j^0} Z)\end{aligned}$$

Theorem 5.1

Let $\mathcal{R}$ be a register machine, and let $v_1, \ldots, v_m$ be initial values for the registers of $\mathcal{R}$. Let $\Sigma_{\mathcal{R}} = (N_{\mathcal{R}}, M)$ be a Petri net, where M is the marking that puts v_i tokens on the place R_i. $\mathcal{R}$ halts for these initial values iff $\Sigma_{\mathcal{R}}$ does not satisfy $\neg halt(\mathcal{R})$.

Proof:
Σ_R satisfies $\neg halt(\mathcal{R})$ iff $L(\Sigma_{\mathcal{R}}) \subseteq \|\neg halt(\mathcal{R})\|$ iff $L(\Sigma_{\mathcal{R}}) \cap \|halt(\mathcal{R})\| = \emptyset$. Let σ be a sequence of $\|halt(\mathcal{R})\|$. It follows from the semantics of the linear time μ-calculus that $\sigma = \sigma_1 \ldots \sigma_k\, halt$, where every σ_i is of the form $\delta_i^+\ inc_j$, $\delta_i^-\ dec_j$ or δ_i^0. Moreover, whenever $M_1 \xrightarrow{\delta_i^0} M_2$ along the occurrence sequence corresponding to σ, $M_1(A) = 0$. Such an occurrence sequence simulates a halting computation of the register machine. Conversely, from a halting computation of the register machine we can easily obtain a sequence of $halt(\mathcal{R})$. So $L(\Sigma_{\mathcal{R}}) \cap \|halt(\mathcal{R})\| \neq \emptyset$ iff $\mathcal{R}$ halts with input $v_1, \ldots, v_m$. ∎

As mentioned above, the transitions of the Petri nets derived from register machines have at most one input place and at most one output place. In particular, they are Petri net representations of Basic Parallel Processes [2].
It seems difficult to find an interesting class of Petri nets with infinite state spaces, and properly included in the class derived from register machines. Since the formula $halt(\mathcal{R})$ is also rather simple, this undecidability result seems to determine the decidability border rather neatly.

6 Branching time logics

The modal μ-calculus, has the same syntax as the linear time μ-calculus (although $O_a\phi$ is usually replaced by $\langle a\rangle\phi$), but is interpreted on labelled transition systems. Let

$\mathcal{T} = (\mathcal{S}, \xrightarrow{a}_{a \in Act})$ be a labelled transition system. Given a valuation $\mathcal{V}$ that assigns to each variable a subset of $\mathcal{S}$, the denotation of a formula is defined by the following rules:

$$\begin{array}{rcl} \|Z\|_{\mathcal{V}} & = & \mathcal{V}(Z) \\ \|\neg\phi\|_{\mathcal{V}} & = & \mathcal{S} - \|\phi\|_{\mathcal{V}} \\ \|\phi_1 \wedge \phi_2\|_{\mathcal{V}} & = & \|\phi_1\|_{\mathcal{V}} \cap \|\phi_2\|_{\mathcal{V}} \\ \|\langle a \rangle \phi\|_{\mathcal{V}} & = & \{s \in \mathcal{S} \mid \exists s' \in \mathcal{S}. s \xrightarrow{a} s' \wedge s' \in \|\phi\|_{\mathcal{V}}\} \\ \|\nu Z.\phi\|_{\mathcal{V}} & = & \bigcup\{A \subseteq \mathcal{S} \mid A \subseteq \|\phi\|_{\mathcal{V}[A/Z]}\} \end{array}$$

As in the case of the linear time μ-calculus, the denotation of a closed formula is independent of the valuation.

Given a labelled Petri net $\Sigma = (N, M_0)$, the transition system associated to Σ, denoted by $\mathcal{T}(\Sigma)$, has the set of reachable markings as states; there is a transition labelled by a between two markings M_1, M_2 if $M_1 \xrightarrow{a} M_2$. We say that Σ satisfies a formula ϕ of the μ-calculus if $M_0 \in \|\phi\|_{\mathcal{V}}$.

It is undecidable if a Petri net satisfies a formula. It suffices to proceed as for the linear time μ-calculus, but change $R_j = 0$ into $\neg\langle dec_j \rangle tt$, where tt is true of every state. To prove it, observe that $\|\langle dec_j \rangle tt\|$ are the markings which enable the transition dec_j, which are exactly the markings that put no tokens on R_j. So, in fact, in the modal μ-calculus we can 'test for zero' without having to add atomic sentences.

In spite of its simplicity, this undecidability result has a consequence which may be a little bit surprising. For the class of Basic Parallel Processes (BPPs), bisimulation equivalence has been shown to be decidable [2]; moreover, since BPPs are image-finite, two Basic Parallel Processes are bisimilar if and only if they satisfy the same properties of the modal μ-calculus. So it is decidable if two BPPs satisfy the same properties. However, the model checking problem is undecidable.

The μ-calculus is known to be an extremely powerful logic. The question arises if some weaker but interesting branching time logic could be decidable. Unfortunately, there is little hope of obtaining such a result: we now show that a very weak branching time logic is undecidable. The logic extends propositional logic with an operator $\Diamond_a$ for each action a of the set Act. The syntax is thus:

$$\phi ::= tt \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \Diamond_a \phi$$

Formulas are interpreted on a transition system $\mathcal{T} = (\mathcal{S}, \xrightarrow{a}_{a \in Act})$. Define $\overset{a}{\Longrightarrow}$ as the relation $(\bigcup_{b \in Act - \{a\}} \xrightarrow{b})^\star \xrightarrow{a}$. The denotation of a formula ϕ is a set of states $\|\phi\|$ defined according to the following rules:

$$\begin{array}{rcl} \|tt\| & = & \mathcal{S} \\ \|\neg\phi\| & = & \mathcal{S} - \|\phi\| \\ \|\phi_1 \wedge \phi_2\| & = & \|\phi_1\| \cap \|\phi_2\|_{\mathcal{V}} \\ \|\Diamond_a \phi\| & = & \{s \in \mathcal{S} \mid \exists s' \in \mathcal{S}. s \overset{a}{\Longrightarrow} s'\} \end{array}$$

So $\Diamond_a \phi$ means that it is possible to do an action a, and the state reached after that satisfies ϕ.

We prove undecidability by a reduction from the *containment problem*. An instance of the problem consists of two Petri nets Σ_1, Σ_2 with the same number of places, and a bijection f between the sets of places of Σ_1 and Σ_2. f can be extended to a bijection between markings in the obvious way. The question to decide is if for every reachable marking M of Σ_1, $f(M)$ is a reachable marking of Σ_2. Rabin showed that this problem is undecidable; the proof can be found in [5].

Let Σ_1, Σ_2 and f be an instance of the containment problem; assume without loss of generality that the sets of nodes of Σ_1 and Σ_2 are disjoint. We construct an unlabelled Petri net $\Sigma = (N, M_0)$ in several stages. At each stage we add some places and transitions. For transitions we use the notation $X \to Y$, meaning that the transition t has the sets X and Y of places as preset and postset, respectively; thus, the flow relation is specified at the same time as the set of transitions. If we want to give the transition the name t, we write $t\colon X \to Y$.

- Add $\Sigma_1 \cup \Sigma_2$.
- Add three places A, B, C, and two transitions $t_{AB}\colon \{A\} \cup S_1 \to \{B\} \cup S_1$ and $t_{BC}\colon \{B\} \cup S_2 \to \{C\} \cup S_2$. Put one token on A, and no tokens on B and C.
- For every place s of Σ_1, add a transition $\{s, f(s), C\} \to \{C\}$.
- For every place s added so far, with the exception of C, add a transition $\{s\} \to \{s\}$.

Every marking M of Σ projects onto a marking M_1 of Σ_1, a marking M_2 of Σ_2, and a marking M_3 of the three places A, B and C; moreover, these three projections determine M. Making use of this property, we write $M = (M_1, M_2, M_3)$; also, we write M_3 as a string of three numbers representing the number of tokens in A, B, and C. Finally, the null marking, i.e., the marking that puts no token in any place, is denoted by 0.

The following lemma follows easily from the definition of Σ.

Lemma 6.1

If $(M_{01}, M_{02}, 100) \xrightarrow{\sigma} (M_1, M_2, 001)$ *is an occurrence sequence of* Σ*, then there exist sequences* $\sigma_1, \sigma_2, \sigma_3$ *such that* $\sigma = \sigma_1\, t_{AB} \sigma_2\, t_{BC}\, \sigma_3$. ∎

We can now characterise the 'yes' instances of the containment problem in terms of occurrence sequences of Σ.

Lemma 6.2

Σ_1, Σ_2, f is a 'yes' instance of the containment problem iff every occurrence sequence $(M_{01}, M_{02}, 100) \xrightarrow{\sigma\, t_{AB}} (M_1, M_{02}, 010)$ can be extended to a maximal occurrence sequence $(M_{01}, M_{02}, 100) \xrightarrow{\sigma\, t_{AB}\, \tau} (0, 0, 001)$.

Proof:

($\Rightarrow$): M_1 is a reachable marking of Σ_1. Since M_1 is reachable in Σ_2 as well, there exists a sequence $\sigma_2\, t_{BC}$ such that

$$(M_1, M_{02}, 010) \xrightarrow{\sigma_2\, t_{BC}} (M_1, M_1, 001)$$

Let σ_3 be a sequence which contains each transition of the form $\{s, f(s), C\} \to \{C\}$ $M(s)$ times, and no occurrence of any other transition. We have $(M_1, M_1, 001) \xrightarrow{\sigma_3} (0, 0, 001)$. By the definition of Σ, no transition is enabled at the marking $(0, 0, 001)$, and therefore

$$(M_0, M_0, 100) \xrightarrow{\sigma\, t_{AB}\, \sigma_2\, t_{BC}\, \sigma_3} (0, 0, 001)$$

is maximal.

($\Leftarrow$): Let M_1 be a reachable marking of Σ_1. Then, we have

$$(M_{01}, M_{02}, 100) \xrightarrow{\sigma_1} (M_1, M_{02}, 100) \xrightarrow{t_{AB}} (M_1, M_{02}, 010)$$

for some sequence σ_1. So there exists a sequence τ such that

$$(M_{01}, M_{02}, 010) \xrightarrow{\sigma\, t_{AB}\, \tau} (0, 0, 001)$$

is a maximal occurrence sequence. By Lemma 6.1, $\tau = \sigma_2\, t_{BC}\, \sigma_3$ for some sequences σ_2 and σ_3. Let

$$(M_1, M_{02}, 010) \xrightarrow{\sigma_2\, t_{BC}} (M_1', M_2, 010) \xrightarrow{\sigma_3} (0, 0, 100)$$

We show $M_1' = M_1 = M_2$, which proves that M_1 is reachable in Σ_2.

Since σ_2 occurs after t_{AB} has occurred, no transition of Σ_1 occurs in σ_2. So $M_1' = M_1$. Since σ_3 occurs after t_{BC}, only transitions of the form $\{s\} \to \{s\}$ for some place s can occur in σ_3. Since these transitions remove one token from s and one from $f(s)$, we have $M_1' - 0 = M_2 - 0$. So $M_1' = M_2$. ■

To finish the undecidability proof, it suffices to encode the characterisation of Lemma 6.2 into the temporal logic defined above.

Theorem 6.3

Σ_1, Σ_2, f is a 'yes' instance of the containment problem iff Σ satisfies the formula

$$\Box_{t_{AB}} \bigvee_{t \in T} (\Diamond_t \bigwedge_{u \in T} \neg \Diamond_u)$$

where T is the set of transitions of Σ.

Proof:
The formula states that every occurrence sequence $\sigma_1\ t_{AB}$ can be extended to an occurrence sequence $\sigma_1\ t_{AB}\ \tau$ leading to a marking at which no transition of Σ is enabled. By the definition of Σ, this marking can only be $(0, 0, 001)$. Apply then Lemma 6.2. ■

7 Conclusions

We have examined the decidability of the model checking problem for several μ-calculi and Petri nets. The decidability border turns out to be rather neat: if we consider only 'pure' μ-calculi, i.e., without atomic sentences, the whole linear time μ-calculus is decidable, while one of the weakest branching time subsets of the modal μ-calculus is undecidable. Also, we have seen that the addition of very simple atomic sentences makes the linear time μ-calculus undecidable, even for formulas with one fixpoint and Petri nets in which every transition has at most one input and at most one output place.

Acknowledgements

I thank Julian Bradfield, Mads Dam and Colin Stirling for very helpful discussions.

References

[1] J. C. Bradfield: Verifying Temporal Properties of Systems. Birkhäuser, Boston, Mass. ISBN 0-8176-3625-0 (1991).

[2] Soren Christensen, Yoram Hirshfeld and Faron Moller: Bisimulation Equivalence is Decidable for basic Parallel Processes. Proceedings of CONCUR'93, LNCS 715, 143–157 (1993).

[3] M. Dam: Fixpoints of Büchi automata. LFCS Report ECS-LFCS-92-224, University of Edinburgh (1992).

[4] M. Jantzen and R. Valk: The Residue of Vector Sets with Applications to Decidability Problems in Petri Nets. Acta Informatica 21, 643–674 (1985)

[5] M.H.T. Hack: Decidability questions for Petri nets. Ph. D. Thesis, MIT (1976).

[6] R.R. Howell and L.E. Rosier: Problems concerning fairness and temporal logic for conflict-free Petri nets. Theoretical Computer Science 64, 305–329 (1989).

[7] R.R.Howell, L.E. Rosier and H. Yen: A taxonomy of fairness and temporal logic problems for Petri nets. Theoretical Computer Science 82, 341–372 (1991).

[8] P. Jančar: Decidability of a temporal logic problem for Petri nets. Theoretical Computer Science 74, 71–93 (1990).

[9] R.M. Karp and R.E. Miller: Parallel Program Schemata. Journal of Computer and System Sciences 3, 147–195 (1969).

[10] S.R. Kosaraju: Decidability of reachability in vector addition systems. Proceedings of the 6th Annual ACM Symposium on the Theory of Computing, 267–281 (1982).

[11] R. Lipton: The Reachability Problem Requires Exponential Space. Technical Report 62, Yale University (1976).

[12] E.W. Mayr: An algorithm for the general Petri net reachability problem. SIAM Journal of Computing 13, 441-460 (1984).

[13] M. Minsky: Computation: Finite and Infinite Machines. Prentice-Hall (1967).

[14] J.L. Peterson: Petri Net Theory and the Modelling of Systems. Prentice-Hall (1981).

[15] C. Stirling: Modal and Temporal Logics. In S. Abramsky, D. Gabbay and T. Maibaun (eds.) Handbook of Logic in Computer Science. Oxford University Press (1991)

[16] M.Y. Vardi and P. Wolper: Automata Theoretic Techniques for Modal Logics of Programs. Journal of Computer and System Sciences 32, 183–221 (1986).

[17] H. Yen: A Unified Approach for Deciding the Existence of Certain Petri Net Paths. Information and Computation 96(1), 119–137 (1992).