

Model Checking for Linear Time μ -calculus and Extended Petri Nets *

Presented at 5th International Workshop on Symbolic and Numeric Algorithms for Scientific Computing SYNACS03, Timișoara, Romania

Ferucio Laurențiu Țiplea and Olivia Oanea

Abstract. Finite jumping and \mathcal{L}_3 -conditional Petri nets, where \mathcal{L}_3 denotes the family of regular languages, are two proper extensions of Petri nets. In this paper we show that the linear time μ -calculus model checking problem for such Petri net extensions is decidable. Complexity issues are also discussed.

Key words: Petri nets, μ -calculus, model checking, complexity

AMS Subject Classification: 68Q60

1. Introduction Model checking is a method for formally verifying finite-state concurrent systems. Specifications about the system are expressed as temporal logic formulas, and efficient symbolic algorithms are used to traverse the model defined by the system and check if the specification holds or not. In recent years, researchers have tried to extend the applicability of model-checking to infinite state systems like pushdown systems, Petri nets, basic parallel processes etc. (Esparza, 1994, Esparza, 1997, Haddad and Poitrenaud, 2001, Latvala, 2001, Mayr, 1998, Wolper and Boigelot, 1998).

Much work has been done on model checking Petri nets for various logics. In (Esparza, 1994), Esparza has proved that the model checking problem for linear time μ -calculus and Petri nets is decidable. The technique is adapted from (Vardi and Wolper, 1986), and roughly speaking it consists of reducing the model checking problem to an emptiness problem. The same technique has been used by Latvala in order to model check

*The research reported in this paper was partially supported by CNCSIS grant 632/2004.

colored Petri nets against linear time logic (Latvala, 2001), and by Haddad and Poitrenaud in order to model check sequential recursive Petri nets against linear time μ -calculus (Haddad and Poitrenaud, 2001).

In this paper we show that the linear time μ -calculus model checking problem for finite jumping and \mathcal{L}_3 -conditional Petri nets is decidable, where \mathcal{L}_3 denotes the family of regular languages. In order to do that we follow the same line as the authors mentioned above (i.e., adapted from (Vardi and Wolper, 1986)), but with specific results for these classes of Petri nets. As both finite jumping and \mathcal{L}_3 -conditional Petri nets extend properly classical Petri nets, Esparza’s result regarding decidability of linear time μ -calculus for Petri nets becomes a particular case of our results.

Finite jumping Petri nets are classical Petri nets Σ equipped with a finite binary relation R on the markings of Σ . If $(M, M') \in R$, then the Petri net Σ may “spontaneously jump” from the marking M to M' (this is similar to λ -moves in automata theory). Clearly, Petri nets are particular cases of jumping Petri nets (the case of an empty relation R). Moreover, finite jumping Petri nets are strictly more expressive than Petri nets at least from the interleaving semantics point of view (Tiplea and Mäkinen, 1997). When modelling systems by Petri nets, the extension to jumps is useful for several reasons (Tiplea and Jucan, 1994, Tiplea and Desel, 1999, Tiplea and Tiplea, 1999):

- irrelevant parts of the behavior may be hidden;
- exception handling and recovery mechanisms can be added;
- recursive calls of procedures can be nicely modelled by jumps.

Conditional Petri nets (Tiplea, 1991, Tiplea, 1994, Tiplea and Bădăraău, 2000) have a different policy than jumping Petri nets. In such Petri nets, a language is associated to each transition. Then, a transition t can fire at a marking M only if there is a transition sequence leading to M which is a member of the language associated to t . In other words, the firing of t is “conditioned” by the previous transition sequence. \mathcal{L}_3 -conditional Petri nets extend strictly the power of Petri nets, as it has been shown in (Tiplea, 1994).

The paper is organized as follows. In section 2 we recall basic definitions for (jumping, conditional) Petri nets, and linear time μ -calculus. In section 3 we prove that the model checking problem for finite jumping and \mathcal{L}_3 -conditional Petri Nets against linear μ -calculus is decidable. Finally, the complexity of model-checking is discussed.

2. Preliminaries The aim of this section is to establish the basic terminology, notations, and results concerning Petri nets and linear time logics in order to give the reader the necessary prerequisites for the understanding of this paper (for details the reader is referred to (Reisig, 1985, Tiplea and Mäkinen, 1997, Dam, 1994)).

The set of nonnegative integers is denoted by \mathbf{N} . For a (finite) alphabet V , V^* (V^ω , respectively) denotes the set of finite (infinite, resp.) words over V . The empty word is λ , and V^∞ stands for $V^* \cup V^\omega$. The family of regular languages (Hopcroft, 1979) is denoted by \mathcal{L}_3 .

Petri Nets. A *Petri net* is a 4-tuple $\Sigma = (S, T, F, W)$, where S and T are two finite non-empty sets (of *places* and *transitions*, respectively), $S \cap T = \emptyset$, $F \subseteq (S \times T) \cup (T \times S)$ is the *flow relation*, and $W : (S \times T) \cup (T \times S) \rightarrow \mathbf{N}$ is the *weight function* of Σ verifying $W(x, y) = 0$ iff $(x, y) \notin F$.

A *marking* of a Petri net Σ is a function $M : S \rightarrow \mathbf{N}$. A *marked Petri net* is a pair $\gamma = (\Sigma, M_0)$, where Σ is a Petri net and M_0 , the *initial marking* of γ , is a marking of Σ . A *labeled marked Petri net* is a 3-tuple $\gamma = (\Sigma, M_0, l)$, where the first two components form a marked Petri net and l , the *labeling function* of γ , assigns to each transition a letter (label). In the sequel we often use the term “Petri net” whenever we refer to one of the concepts introduced above.

The sequential behavior of a Petri net γ is given by the *transition rule*, which is defined as follows. A transition t is *enabled* at a marking M (in γ), abbreviated $M[t]_\gamma$, if $W(s, t) \leq M(s)$ for all $s \in S$. If $M[t]_\gamma$, then t may *occur* yielding a new marking M' , abbreviated $M[t]_\gamma M'$, defined by $M'(s) = M(s) + W(t, s) - W(s, t)$ for all $s \in S$. The transition rule is usually extended to finite and infinite sequences of transitions.

A *finite computation (from M_0) of γ* is any sequence of the form

$$M_0[t_1]_\gamma M_1[t_2]_\gamma \cdots M_{n-1}[t_n]_\gamma M_n;$$

the sequence $t_1 \cdots t_n$ is called a *finite transition sequence (from M_0) of γ* , and M_n is called *reachable (from M_0) in γ* . *Infinite computations* and *infinite transition sequences* of γ are defined similarly. A *computation* of γ is any finite or infinite computation of γ .

A marking M of a Petri net γ is called a *dead marking* if no transition is enabled at M . A computation of γ is *maximal* if it is either infinite or, if it is finite then its last marking is a dead marking.

By extending homomorphically the labeling function of Petri nets to sequences of transitions we define the *language of γ* , denoted $L(\gamma)$, as being the set of all finite or infinite words $l(w)$ such that w induces a maximal computation of γ . Words $l(w) \in L(\gamma)$ are called *runs* of γ . We remark that a (finite or infinite) sequence of transitions of γ induces at most a computation of γ .

Extensions of Petri Nets. Many systems of interest cannot be adequately modeled by Petri nets. From this reason, several extensions of Petri nets have been proposed. Two of them are that of *jumping Petri nets* and *conditional Petri nets*.

A *jumping Petri net* (Tiplea and Jucan, 1994, Tiplea and Mäkinen, 1997) is a pair $\gamma = (\Sigma, R)$, where Σ is a Petri net and R , the *set of (spontaneous) jumps* of γ , is a binary relation on the set of markings of Σ . Pairs $(M, M') \in R$ are referred to as *jumps* of γ . If γ has finitely many jumps then we say that γ is a *finite jumping Petri net*.

The concepts of a marked jumping Petri net and labeled marked jumping Petri net are introduced in a similar way as for Petri nets. For example, $\gamma = (\Sigma, R, M_0, l)$ denotes a labeled marked jumping Petri net. We shall generally refer to these structures as “jumping Petri nets” or “finite jumping Petri nets”.

The *transition rule* of jumping Petri nets is as follows. A transition t is *enabled* at a marking M (in γ), abbreviated $M[t]_\gamma$, if there is a marking M_1 such that $MR^*M_1[t]_\Sigma$ (Σ is the underlying net of γ and R^* is the reflexive and transitive closure of R). The marking M' is *produced* by the occurrence of t at M , abbreviated $M[t]_\gamma M'$, if there exist markings M_1 and M_2 such that $MR^*M_1[t]_\Sigma M_2R^*M'$.

The concepts of a finite/infinite computation, finite/infinite transition sequence, reachable marking, dead marking, maximal computation, language, and run are similarly introduced for jumping Petri nets as for Petri nets. As opposite to Petri nets, a sequence of transitions of a jumping Petri net may induce more than a computation (depending on its set of jumps).

Conditional Petri nets (Tiplea, 1991, Tiplea, 1994) have a different policy than jumping Petri nets. The main idea is to associate to each transition t a language and to extend a transition sequence w by t only if w is in the language associated to t . Formally, let \mathcal{L} be an arbitrary family of languages. An \mathcal{L} -*conditional Petri net* is a pair $\gamma = (\Sigma, \varphi)$, where Σ is a Petri net, and φ is a function from T into $\mathcal{P}(T^*) \cap \mathcal{L}$. *Marked* and *labeled marked \mathcal{L} -conditional Petri nets* are defined as in the case of jumping Petri

nets. We shall generally refer to these structures as *conditional Petri nets*.

The *transition rule* of conditional Petri nets is defined as follows. Let M be a marking and $u \in T^*$. Then, a transition t is *enabled* at (M, u) , abbreviated $(M, u)[t]_\gamma$, iff $M[t]_\Sigma$ and $u \in \varphi(t)$. If $(M, u)[t]_\gamma$, then t may *occur* yielding a pair (M', v) , abbreviated $(M, u)[t]_\gamma(M', v)$, where $M[t]_\Sigma M'$ and $v = ut$.

A *finite computation (from M_0) of γ* is any sequence of the form

$$(M_0, \lambda)[t_1]_\gamma(M_1, t_1)[t_2]_\gamma \cdots (M_{n-1}, t_1 \cdots t_{n-1})[t_n]_\gamma(M_n, t_1 \cdots t_{n-1}t_n);$$

the sequence $t_1 \cdots t_n$ is called a *finite transition sequence (from M_0) of γ* , and the pair $(M_n, t_1 \cdots t_{n-1}t_n)$ is called a *reachable configuration (from M_0) in γ* . *Infinite computations* and *transition sequences* of γ are defined similarly. A *computation* of γ is any finite or infinite computation of γ .

A configuration (M, u) of a conditional Petri net γ is called a *dead configuration* if no transition is enabled at it. A computation of γ is *maximal* if it is either infinite or, if it is finite then its last marking is a dead marking. We want to point out that a dead configuration (M, u) does not mean that M is a dead marking. The marking M may be reachable by another transition sequence, e.g. v , and the configuration (M, v) be not dead.

The concepts of language and run are similarly introduced for conditional Petri nets as for Petri nets. A (finite or infinite) sequence of transitions of a conditional Petri net γ induces at most a computation of γ .

Linear Time μ -calculus. There are two main classes of temporal and modal logics: *branching time logics* and *linear time logics*. The difference between these two logics consists in how they are interpreted. The first one is interpreted over computation trees, while the second one is interpreted over sets of runs. In the most general sense, temporal logic formulae are interpreted over computations of processes given as arbitrary labeled transition systems. They can be given a *state-based* or an *action-based* semantics, or a combination of the two. In state-based semantics, formulae are built out of atomic propositions and interpreted according to a valuation that assigns to each atomic proposition a set of states in the transition system (the states that satisfy this proposition). In this case, the labels are not important at all and, therefore, they can be ignored. In action-based semantics, *true* is the only atomic sentence, and the information carried by the states is ignored. In this semantics, logics have relativised next opera-

tors, one for each possible label (which is an atomic action). In this paper we use action-based semantics.

Linear time μ -calculus, abbreviated $LT\mu C$, is one of the most expressive linear time logics (see, e.g., (Dam, 1994)). Its syntax is given by

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

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

Free and *bound* occurrences of variables in $LT\mu C$ -formulas are defined as usual. An $LT\mu C$ -formula is *closed* if no variable has free occurrences in it.

In order to define the semantics of linear time μ -calculus, a *syntactic monotonicity condition* must be imposed (Dam, 1994). It states that for the $LT\mu C$ -formula $\nu X.\phi$ to be well-formed any occurrence of X in ϕ must be within the scope of an even number of negations. Then, extend well-formedness to arbitrary $LT\mu C$ -formulas ψ by requiring all subformulas of ψ of the form $\nu X.\phi$ to be well-formed. Therefore, in what follows, $LT\mu C$ -formulas are considered always well-formed.

A *valuation* \mathcal{V} of the logic assigns to each variable X a set of words $\mathcal{V}(X) \subseteq Act^\infty$. We denote by $\mathcal{V}[X/A]$ the valuation given by

$$\mathcal{V}[X/A](X') = \begin{cases} \mathcal{V}(X'), & \text{if } X' \neq X \\ A, & \text{otherwise} \end{cases}$$

for all variables X' and $A \subseteq Act^\infty$.

Given a (finite or infinite) word $\sigma = a_1a_2a_3\cdots$ on Act , $\sigma(1)$ stands for the first action of σ , i.e. a_1 , and σ^1 stands for the word $a_2a_3\cdots$.

The *denotation* $\|\phi\|_{\mathcal{V}}$ of an $LT\mu C$ -formula ϕ w.r.t. a valuation \mathcal{V} is given by:

$$\begin{aligned} \|X\|_{\mathcal{V}} &= \mathcal{V}(X) \\ \|\neg\phi\|_{\mathcal{V}} &= Act^\infty - \|\phi\|_{\mathcal{V}} \\ \|\phi_1 \wedge \phi_2\|_{\mathcal{V}} &= \|\phi_1\|_{\mathcal{V}} \cap \|\phi_2\|_{\mathcal{V}} \\ \|O_a\phi\|_{\mathcal{V}} &= \{\sigma \in Act^\infty \mid \sigma(1) = a \wedge \sigma^1 \in \|\phi\|_{\mathcal{V}}\} \\ \|\nu X.\phi\|_{\mathcal{V}} &= \bigcup \{A \subseteq Act^\infty \mid A \subseteq \|\phi\|_{\mathcal{V}[X/A]}\} \end{aligned}$$

The *satisfaction* relation is then defined by

$$\sigma \text{ satisfies } \phi \text{ w.r.t. } \mathcal{V} \quad \text{iff} \quad \sigma \in \|\phi\|_{\mathcal{V}},$$

for all $\sigma \in Act^\infty$, LT μ C-formula ϕ , and valuation \mathcal{V} .

We remark that the denotation of a closed LT μ C-formula is independent of the valuation. Therefore, we shall write $\|\phi\|$ instead of $\|\phi\|_{\mathcal{V}}$.

3. Petri Nets and Linear Time μ -Calculus A (jumping, conditional) Petri net γ *satisfies* an LT μ C-formula ϕ if σ satisfies ϕ , for all $\sigma \in L(\gamma)$. In other words, γ satisfies ϕ if $L(\gamma) \subseteq \|\phi\|$.

The *model-checking problem for LT μ C and (jumping, conditional) Petri nets* consists in deciding whether or not a given (jumping, conditional) Petri net satisfies a given LT μ C-formula.

One of the methods to get a decision procedure for a problem like the above one is to use an automata-theoretic characterization of the logic. This idea has been introduced by Vardi and Wolper (Vardi and Wolper, 1986) and used intensively since then. It consists mainly in the following:

- assume that $M \in \mathcal{M}$ is a model of a (finite- or infinite-state) system, and ϕ is a formula (of some logic), where \mathcal{M} is a class of models (e.g., automata, Petri nets, pushdown processes, basic parallel processes etc.);
- model-checking M against ϕ means deciding whether or not $L(M) \subseteq \|\phi\|$ or, equivalently, $L(M) \cap \overline{\|\phi\|} = \emptyset$, where $L(M)$ is the language of M , defined in some suitable way (see, for example, the language of a Petri net in Section 2), and $\overline{\|\phi\|}$ is the complement of $\|\phi\|$;
- assume that an automaton $A_{\neg\phi}$ can be effectively constructed, such that $L(A_{\neg\phi}) = \overline{\|\phi\|}$. Therefore, the above problem reduces to deciding whether or not $L(M) \cap L(A_{\neg\phi}) = \emptyset$;
- assume again that a new model M' can be effectively constructed, such that $L(M') = L(M) \cap L(A_{\neg\phi})$. In this case, the model checking problem can be reduced to the emptiness problem for the class \mathcal{M} of models.

The above method, and its efficiency, depends highly on the class \mathcal{M} of models. It has been applied to Petri nets by Esparza (Esparza, 1994), to colored Petri nets by Latvala (Latvala, 2001), to sequential recursive Petri nets by Haddad and Poitrenaud (Haddad and Poitrenaud, 2001) etc.

We shall show that the same technique can be applied to finite jumping and \mathcal{L}_3 -conditional Petri nets, which are proper extensions of Petri nets. From this point of view, our result extends Esparza's result regarding linear time μ -calculus model checking for Petri nets (Esparza, 1994).

3.1. The Automata Associated to an LT μ C-formula Let ϕ be a closed formula of the linear time μ -calculus. By a slight modification of a construction given by Dam (Dam, 1992), Esparza showed in (Esparza, 1994) how to construct a finite automaton A_ϕ and a Büchi automaton B_ϕ such that $L(A_\phi) = \|\phi\| \cap Act^*$ and $L(B_\phi) = \|\phi\| \cap Act^\omega$.

We shall make use of these automata without presenting explicitly their construction.

3.2. The Equivalent Petri Net Associated to an Automaton A well-known construction associates to an automaton $A = (Q, Act, q_0, \delta, Q_f)$ ² an “equivalent” Petri net $\gamma_A = (\Sigma, M_0, l)$. We recall here the construction using our notation:

- $S = Q$ and $T = \delta$;
- $F = \{(q, (q, a, q')), ((q, a, q'), q') \mid (q, a, q') \in \delta\}$;
- $W((x, y)) = 1$, for all $(x, y) \in F$;
- $M_0(q_0) = 1$ and $M_0(q) = 0$, for all $q \in Q - \{q_0\}$;
- $l((q, a, q')) = a$, for all $(q, a, q') \in \delta$.

Given a state q of A , denote by M_q the marking of γ_A given by $M(q) = 1$ and $M(q') = 0$, for all $q' \in Q - \{q\}$. We say that M_q *corresponds* to q . We remark that γ_A has finitely many reachable markings, each of them corresponding to some state of A .

By this construction applied to A_ϕ and B_ϕ in Section 3.1 we get:

- if $q_0, a_1, q_1, a_2, q_2, \dots, q_{n-1}, a_n, q_n$ is a path in A_ϕ (i.e., $(q_i, a_{i+1}, q_{i+1}) \in \delta$ for all $0 \leq i \leq n-1$), then

$$M_0[(q_0, a_1, q_1)]_{\gamma_{A_\phi}} M_{q_1}[(q_1, a_2, q_2)]_{\gamma_{A_\phi}} M_{q_2} \cdots \\ \cdots M_{q_{n-1}}[(q_{n-1}, a_n, q_n)]_{\gamma_{A_\phi}} M_{q_n}$$

is a computation in γ_{A_ϕ} , and conversely. Therefore, A_ϕ accepts a word σ iff there is a transition sequence w of γ_{A_ϕ} such that $l(w) = \sigma$ and w leads to a marking corresponding to a final state of A_ϕ ;

- By a similar reasoning to the above one we obtain that B_ϕ accepts a word σ iff there is an infinite transition sequence w of γ_{B_ϕ} such that $l(w) = \sigma$ and infinitely many intermediate markings in the computation induced by w correspond to final states in B_ϕ .

² Q is a finite set of states, Act is a finite alphabet, q_0 is the initial state, δ is the transition relation, and $Q_f \subseteq Q$ is the set of final states.

3.3. Action Based Composition of Petri Nets The next step is to define an *action based composition* between a (jumping, conditional) Petri net and the Petri net associated to an automaton.

Let $\gamma_1 = (\Sigma_1, R_1, M_0^1, l_1)$ be a jumping Petri net and $\gamma_2 = (\Sigma_2, M_0^2, l_2)$ be a Petri net such that γ_1 and γ_2 are *disjoint* (i.e., $(S_1 \cup T_1) \cap (S_2 \cup T_2) = \emptyset$). For any two markings M_1 of γ_1 and M_2 of γ_2 denote by (M_1, M_2) the marking $M : S_1 \cup S_2 \rightarrow \mathbf{N}$ given by $M(s) = M_1(s)$ for all $s \in S_1$, and $M(s) = M_2(s)$ for all $s \in S_2$. Now, define a new jumping Petri net $\gamma_1 \times \gamma_2 = (\Sigma, R, M_0, l)$ as follows:

- $S = S_1 \cup S_2$;
- $T = \{(t_1, t_2) \mid t_1 \in T_1 \wedge t_2 \in T_2 \wedge l_1(t_1) = l_2(t_2)\}$;
- $F = (\{(s, (t_1, t_2)) \mid (s, t_1) \in F_1 \vee (s, t_2) \in F_2\} \cap (S \times T)) \cup (\{(t_1, t_2), s) \mid (t_1, s) \in F_1 \vee (t_2, s) \in F_2\} \cap (T \times S))$;
- the weight function W agrees with W_1 on arcs $(s, (t_1, t_2)) \in F$ if $(s, t_1) \in F_1$, or on arcs $((t_1, t_2), s) \in F$ if $(t_1, s) \in F_1$, and with W_2 in all the other cases;
- $R = \{((M_1, M), (M_2, M)) \mid (M_1, M_2) \in R_1 \wedge M \text{ reachable in } \gamma_2\}$;
- $M_0 = (M_0^1, M_0^2)$;
- $l((t_1, t_2)) = l_1(t_1)$, for all $(t_1, t_2) \in T$.

A similar construction can be applied to conditional Petri nets. More precisely, if $\gamma_1 = (\Sigma_1, \varphi_1, M_0^1, l_1)$ is a conditional Petri net, then $\gamma_1 \times \gamma_2 = (\Sigma, \varphi, M_0, l)$ is a conditional Petri net, where Σ , M_0 , and l are defined as above, and

- $\varphi((t_1, t_2)) = \{u \in T^* \mid pr_1(u) \in \varphi_1(t_1)\}$, for any transition $(t_1, t_2) \in T$,

where pr_1 is the first projection function on 2-dimensional vectors extended to words of such vectors (e.g., $pr_1((t_1, t_2)(t_3, t_4)) = t_1 t_3$). Moreover, we shall consider $\lambda \in \varphi((t_1, t_2))$ whenever $\lambda \in \varphi_1(t_1)$.

The following two lemmata give some basic properties of the product $\gamma_1 \times \gamma_2$.

Lemma 1. *Let γ_1 be a jumping Petri net and γ_2 be a Petri net such that γ_1 and γ_2 are disjoint. Then, the following are true:*

- (1) *If γ_1 has finitely many jumps and γ_2 has finitely many reachable markings, then $\gamma_1 \times \gamma_2$ is a finite jumping Petri net.*
- (2) *$(M_1, M_2)[(t_1, t_2)]_{\gamma_1 \times \gamma_2} (M'_1, M'_2)$ iff $M_1[t_1]_{\gamma_1} M'_1$ and $M_2[t_2]_{\gamma_2} M'_2$, for all markings M_1, M'_1 of γ_1 and M_2, M'_2 of γ_2 , and for any*

transition (t_1, t_2) of $\gamma_1 \times \gamma_2$.

Proof. (1) follows directly from the construction of $\gamma_1 \times \gamma_2$.

In order to prove (2) assume that $(M_1, M_2)[(t_1, t_2)]_{\gamma_1 \times \gamma_2}(M'_1, M'_2)$. Then, there are markings M_3, M_4, M_5 , and M_6 such that, using the notation above, the following holds:

$$(M_1, M_2)R^*(M_3, M_4)[(t_1, t_2)]_{\Sigma}(M_5, M_6)R^*(M'_1, M'_2).$$

It is easy to see that $M_2 = M_4$ and $M_6 = M'_2$. Therefore,

$$M_1R^*_1M_3[t_1]_{\Sigma_1}M_5R^*_1M'_1,$$

and $M_2[t_2]_{\gamma_2}M'_2$. Equivalently, $M_1[t_2]_{\gamma_1}M'_1$ and $M_2[t_2]_{\gamma_2}M'_2$, which proves the first implication.

The proof goes similarly for the converse implication.

Lemma 2. *Let γ_1 be an \mathcal{L} -conditional Petri net and γ_2 be a Petri net such that γ_1 and γ_2 are disjoint. Then, the following are true:*

- (1) *If $\mathcal{L} = \mathcal{L}_3$, then $\gamma_1 \times \gamma_2$ is an \mathcal{L}_3 -conditional Petri net.*
- (2) *$((M_1, M_2), u)[(t_1, t_2)]_{\gamma_1 \times \gamma_2}((M'_1, M'_2), u(t_1, t_2))$ iff*
 - $(M_1, pr_1(u))[t_1]_{\gamma_1}(M'_1, pr_1(u)t_1)$, and
 - $M_2[t_2]_{\gamma_2}M'_2$,

for all markings M_1, M'_1 of γ_1 and M_2, M'_2 of γ_2 , and for any sequence of transitions u and transition (t_1, t_2) of $\gamma_1 \times \gamma_2$.

Proof. (1) The substitution $\sigma : T_1 \rightarrow 2^{T_1 \times T_2}$ given by

$$\sigma(t_1) = \{(t_1, t_2) | t_2 \in T_2 \text{ and } l_1(t_1) = l_2(t_2)\},$$

for all $t_1 \in T_1$, is finite and satisfies $\varphi((t_1, t_2)) = \sigma(\varphi_1(t_1))$, for all transitions (t_1, t_2) of $\gamma_1 \times \gamma_2$. This shows that $\varphi((t_1, t_2))$ is a regular language because $\varphi_1(t_1)$ is regular and \mathcal{L}_3 is closed under finite substitutions.

(2) Assume first that $((M_1, M_2), u)[(t_1, t_2)]_{\gamma_1 \times \gamma_2}((M'_1, M'_2), u(t_1, t_2))$. Then, $M_2[t_2]_{\gamma_2}M'_2$ and $u \in \varphi((t_1, t_2))$. The definition of φ leads to $pr_1(u) \in \varphi_1(t_1)$, which shows that $(M_1, pr_1(u))[t_1]_{\gamma_1}(M'_1, pr_1(u)t_1)$.

The converse implication can be proved in a similar way.

3.4. Deciding Emptiness Now, we come to the problem of deciding whether or not $L(\gamma) \cap L(A_\phi) = \emptyset$ and $L(\gamma) \cap L(B_\phi) = \emptyset$, where γ is a finite

jumping or an \mathcal{L}_3 -conditional Petri net, ϕ is a LT μ C-formula, and A_ϕ and B_ϕ are the two automata in Section 3.1.

Lemma 3. *Let ϕ be a closed LT μ C-formula.*

- (1) *If γ is a finite jumping Petri net, then $L(\gamma) \cap L(A_\phi) \neq \emptyset$ iff there is a reachable dead marking (M, M_q) of $\gamma \times \gamma_{A_\phi}$ such that M_q corresponds to a final marking q of A_ϕ .*
- (2) *If γ is an \mathcal{L}_3 -conditional Petri net, then $L(\gamma) \cap L(A_\phi) \neq \emptyset$ iff there is a reachable dead configuration $((M, M_q), u)$ of $\gamma \times \gamma_{A_\phi}$ such that M_q corresponds to a final marking q of A_ϕ .*

Proof. (1) Assume first that there is $\sigma \in L(\gamma) \cap L(A_\phi)$.

Since $\sigma \in L(A_\phi)$, σ induces a computation in γ_{A_ϕ} such that the last marking of this computation is of the form M_q , where q is a final state of A_ϕ . Similarly, since $\sigma \in L(\gamma)$, σ induces a computation in γ such that the last marking of this computation is a dead marking M .

By Lemma 1, from these two computations we can find a transition sequence w of $\gamma \times \gamma_{A_\phi}$ such that $l(w) = \sigma$ and (M, M_q) is the last marking of the computation induced by w , where l is the labeling function of $\gamma \times \gamma_{A_\phi}$.

The converse implication can be proved similarly.

(2) Almost the same proof as for (1) can be used in the case of \mathcal{L}_3 -conditional Petri nets. The only difference is that a dead configuration $((M, M_q), w)$ is reached (using the notation above).

By Proposition 3, the problem “ $L(\gamma) \cap L(A_\phi) \neq \emptyset$ ” can be reduced to the problem of deciding whether or not finite jumping Petri nets (\mathcal{L}_3 -conditional Petri nets) have dead markings (dead configurations).

Lemma 4. *It is decidable whether or not a finite jumping Petri net (an \mathcal{L}_3 -conditional Petri net) has dead markings (dead configurations).*

Proof. We consider first the case of finite jumping Petri nets. Let $\gamma = (\Sigma, R, M_0, l)$ be such a Petri net. Consider the set Γ of Petri nets given by

$$\Gamma = \{\gamma_0 = (\Sigma, M_0, l)\} \cup \{\gamma_{(M, M')} = (\Sigma, M', l) \mid (M, M') \in R \wedge M \text{ is reachable in } \gamma\}.$$

As the reachability problem for finite jumping Petri nets is decidable (Tiplea and Jucan, 1994) and R is finite, the set Γ can be effectively constructed.

Now, it is easy to see that γ has dead markings iff γ' has dead markings, for some $\gamma' \in \Gamma$. But, for Petri nets this problem is decidable (Esparza, 1994). Therefore, it is decidable for finite jumping Petri nets.

The proof for \mathcal{L}_3 -conditional Petri nets is heavily based on the construction in (Tiplea and Bădăraș, 2000) used to prove reachability for such nets.

Let γ be an \mathcal{L}_3 -conditional Petri net, $T = \{t_1, \dots, t_n\}$ its set of transitions, $A_i = (Q_i, T, \delta, q_0^i, Q_f^i)$ be a finite deterministic automaton accepting the regular language $\varphi(t_i)$, for all $1 \leq i \leq n$. We may assume that:

- $Q_i \cap Q_j = \emptyset$, for all $i \neq j$, and
- $(S \cup T) \cap \bigcup_{i=1}^n Q_i = \emptyset$.

We transform γ into a marked Petri net $\gamma' = (\Sigma', M'_0)$ by adding to the set S all the sets $S_i = \{p_q | q \in Q_i\}$, and by replacing each transition t_i by some “labeled copies” as follows:

- for each sequence of states $q_1, q'_1 \in Q_1, \dots, q_n, q'_n \in Q_n$ such that $q_i \in Q_f^i$ and $\delta_1(q_1, t_i) = q'_1, \dots, \delta_n(q_n, t_i) = q'_n$, consider a transition $t_{q_1, q'_1, \dots, q_n, q'_n}^i$, which will be connected to places in S as t_i is, and for all $1 \leq j \leq n$,

$$W(p_{q_j}, t_{q_1, q'_1, \dots, q_n, q'_n}^i) = W(t_{q_1, q'_1, \dots, q_n, q'_n}^i, p_{q'_j}) = 1.$$

We assume, without loss of generality, that at least a transition t is enabled at M_0 (in γ) so that T' is non-empty. Let M'_0 be the marking given by:

- $M'(p) = M_0(p)$, for all $p \in S$;
- $M'(p_{q_0^i}) = 1$, for all $1 \leq i \leq n$;
- $M'(p_q) = 0$, for all states $q \in \bigcup_{i=1}^n S_i - \{q_0^i | 1 \leq i \leq n\}$.

Consider next the homomorphism $h : (T')^* \rightarrow T^*$ given by

$$h(t_{q_1, q'_1, \dots, q_n, q'_n}^i) = t_i,$$

for every transition $t_{q_1, q'_1, \dots, q_n, q'_n}^i$ defined as above (see Figure 1 for a standard pictorial view of the Petri net γ').

For any $w \in T^*$ and any marking M of γ we have $(M_0, \lambda)[w]_\gamma(M, w)$ iff there is $w' \in (T')^*$ and a marking M' of γ' such that $h(w') = w$ and $M'_0[w']_{\gamma'} M'$ and $M = M'|_S$.

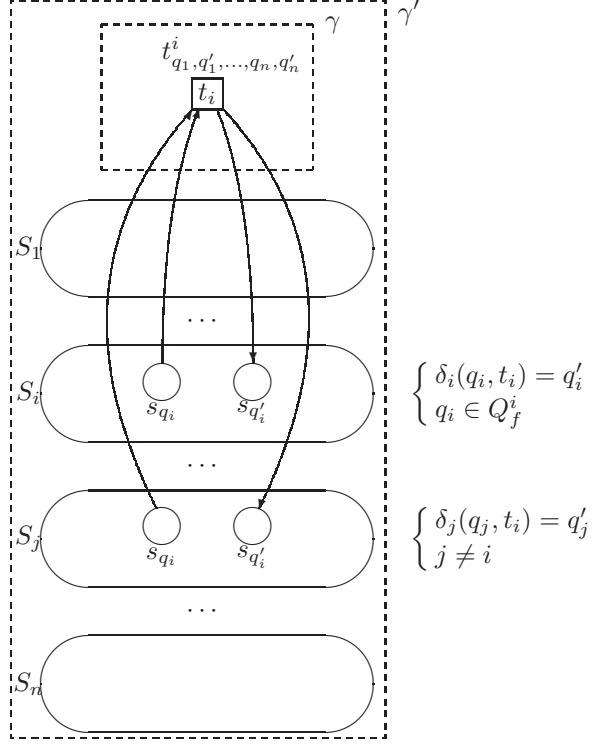


Fig. 1. The Petri net γ'

Now it is easy to check that γ has a reachable dead configuration (M, u) iff γ' has a reachable dead marking M' such that the restriction of M' to S is M . Therefore, γ has reachable dead configurations iff γ' has reachable dead markings, which shows that the dead marking problem for \mathcal{L}_3 -conditional Petri nets is decidable.

Lemma 5. *Let γ be a finite jumping or an \mathcal{L}_3 -conditional Petri net and ϕ be a closed $LT\mu C$ formula. Then, $L(\gamma) \cap L(B_\phi) \neq \emptyset$ iff $\gamma \times \gamma_{B_\phi}$ has an infinite transition sequence which contains infinitely many occurrences of some transition (t_1, t_2) , where t_2 is a transition whose application in γ_{B_ϕ} produces a marking corresponding to a final state of B_ϕ .*

Proof. The proof is very similar to the proof of Proposition 3, but with the difference that infinite computations are considered.

By Proposition 5, the problem “ $L(\gamma) \cap L(B_\phi) \neq \emptyset$ ” can be reduced to the problem of deciding whether or not finite jumping or \mathcal{L}_3 -conditional

Petri nets have infinite transition sequences which contain infinitely many occurrences of some transition.

Lemma 6. *It is decidable whether or not a finite jumping or an \mathcal{L}_3 -conditional Petri net has infinite transition sequences which contain infinitely many occurrences of some transition.*

Proof. Consider first the case of finite jumping Petri nets, and let $\gamma = (\Sigma, R, M_0, l)$ be such a net.

Let t be a distinguished transition of γ , and let Γ be the set defined in the proof of Lemma 4. Define a directed graph $G_\gamma = (V, E)$ as follows:

- $V = \{M_0\} \cup \{M \mid \exists M' : (M, M') \in R\} \cup \{M' \mid \exists M : (M, M') \in R\}$;
- $R \subseteq E$;
- $(M_2, M_3) \in E$ for all $(M_1, M_2), (M_3, M_4) \in R$ such that there is a transition sequence of $\gamma_{(M_1, M_2)} = (\Sigma, M_2, l)$ leading from M_2 to M_3 and containing at least an occurrence of t . Moreover, we mark these arcs (for example, by t).

The graph G_γ can be effectively constructed due to the fact that the reachability problem for both Petri nets and finite jumping Petri nets is decidable. Moreover, G_γ is finite.

Then, γ has the property in lemma w.r.t. t iff one of the following two properties holds true:

1. there is $\gamma' \in \Gamma$ with the property in lemma w.r.t. t , or
2. there is an infinite path from M_0 in G_γ containing a cycle with at least one occurrence of a marked arc.

Both these two properties are decidable. The first one was shown to be decidable by Jantzen and Valk (Jantzen, 1985) (see also the discussion in (Esparza, 1998)), and the last one can be easily decided by a simple inspection of the finite graph G_γ . The construction of the graph G_γ can be easily modified in order to decide whether or not γ has infinite transition sequences which contain infinitely many occurrences of some label.

Let us consider now the case of \mathcal{L}_3 -conditional Petri nets and let $\gamma = (\Sigma, \varphi, M_0)$ be such a net. Let $\gamma' = (\Sigma', M'_0)$ be the corresponding Petri net defined as in the proof of Lemma 4. For any $w \in T^*$ that contains infinitely many occurrences of some transition t_i and for any marking M of γ , we have $(M_0, \lambda)[w]_\gamma(M, w)$ iff there is $w' \in (T')^*$ and a marking M' of γ' such that $h(w') = w$ and $M'_0[w']_{\gamma'} M'$ and $M = M'|_S$ such that $t_{q_1, q'_1, \dots, q_n, q'_n}^i$ occurs infinitely many times in w' . Thus, the decision problem

in the lemma can be reduced to the similar decision problem for classical Petri nets.

3.5. Putting All Together The results we have established in the previous sections lead to the following.

Theorem 1. *Let γ be a finite jumping or an \mathcal{L}_3 -conditional Petri net and ϕ be a closed $LT\mu C$ formula. Then, it is decidable whether or not γ satisfies the formula ϕ .*

Proof. The following equivalences hold true:

$$\begin{aligned} \gamma \text{ satisfies } \phi &\Leftrightarrow L(\gamma) \subseteq \|\phi\| \\ &\Leftrightarrow L(\gamma) \cap \|\neg\phi\| = \emptyset \\ &\Leftrightarrow L(\gamma) \cap L(A_{\neg\phi}) = \emptyset \wedge L(\gamma) \cap L(B_{\neg\phi}) = \emptyset. \end{aligned}$$

Then, the theorem follows from Propositions 3 and 5, and Lemmata 4 and 6.

Petri nets can be regarded as jumping Petri nets with an empty set of jumps, or as \mathcal{L}_3 -conditional Petri nets where the languages associated to transitions contain all the possible sequences of transitions. Therefore, our results can be applied to (standard) Petri nets, getting that the linear time μ -calculus model checking problem for Petri nets is decidable. This is in fact the result obtained by Esparza in (Esparza, 1994).

4. Complexity Issues The space complexity of Esparza's algorithm in (Esparza, 1994) is exponential in the size of the Petri net and double exponential in the size of the formula. Habermehl (Habermehl, 1997) reduced the space complexity to polynomial space in the size of the formula, which is the same as for finite state systems. Moreover, he proved that the model checking problem for Petri nets and weak linear time μ -calculus (a version of the linear time μ -calculus interpreted only over infinite runs) is $PSPACE$ -complete in the size of the formula and $EXPSPACE$ -complete in the size of the Petri net.

Let us discuss the complexity of model checking (finite jumping, conditional) Petri nets against linear time μ -calculus.

The complexity of deciding whether or not a finite jumping Petri net $\gamma = (\Sigma, R, M_0, l)$ has dead markings does not exceed $(|R| + 1)\mathcal{D}_\Gamma$, where \mathcal{D}_Γ is the maximum complexity of deciding whether or not a Petri net in the set Γ has dead markings (using the notations in the proof of Lemma

4). As it was mentioned in (Esparza, 1998), finding a dead marking of a Petri net is as hard as the reachability problem.

The complexity of the problem in Lemma 6, for a given finite jumping Petri net $\gamma = (\Sigma, R, M_0, l)$, is $\max\{(|R| + 1)\mathcal{J}_\Gamma, |R|^2\}$, assuming that the graph G_γ in the proof of Lemma 6 is pre-computed. In this relation, \mathcal{J}_Γ is the maximum complexity of deciding whether the property in Lemma 6 holds true for some Petri net in the set Γ (Yen showed in (Yen, 1992) that this problem is decidable within exponential space). The complexity of finding a cycle in the graph G_γ with at least one occurrence of a given arc does not exceed $|R|^2$.

Therefore, the complexity of deciding whether or not a finite jumping Petri net γ satisfies a linear time μ -calculus formula ϕ depends on the cardinality of the relation R and on the complexity of deciding similar problems for some Petri nets associated to γ .

Let us take now into consideration the case of \mathcal{L}_3 -conditional Petri nets. For such nets, both decision problems (the dead marking problem and the problem in Lemma 6) can be reduced to similar decision problems for Petri nets. Therefore, we have to analyse the size of the Petri net γ' associated to an \mathcal{L}_3 -conditional Petri net γ as defined in the proof of Lemma 4.

Let $\delta_i^t = \{(q, t, q') \mid \delta_i(q, t) = q'\}$, for all i (we use the same notation as in the proof of Lemma 4). Then, $|S'| = |S| + \sum_{i=1}^n |S_i|$ and

$$|T'| \leq \sum_{t \in T} |\delta_1^t| \cdots |\delta_n^t| \leq \delta^{|T|},$$

where $\delta = \max\{|\delta_i| \mid 1 \leq i \leq n\}$. This shows that γ' may grow exponentially in the size of γ .

REFERENCES

- Dam, M. (1992). *Fixpoints of Büchi Automata*, LFCS Report ECS-LFCS-92-224, University of Edinburgh.
- Dam, M. (1994). *Temporal Logic, Automata, and Classical Theories. An Introduction*, Notes for the 6th European Summer School in Logic, Language, and Information, Copenhagen (URL: <ftp://ftp.sics.se/pub/fdt/mfd/tlact.ps.Z>).
- Esparza, J. (1994). *On the Decidability of Model Checking for Several μ -calculi and Petri Nets*, Proceedings of CAAP'94, LNCS 787, pp. 115-129.
- Esparza, J. (1997). *Decidability of Model Checking for Infinite-State Concurrent Systems*, Acta Informatica 34, pp. 85-107.
- Esparza, J. (1998). *Decidability and Complexity of Petri Net Problems - An Introduction*, Lectures on Petri Nets I: Basic Models (W. Reisig, G. Rozenberg, eds.), LNCS 1491, pp. 374-428.

- Habermehl, P. (1997). *On the Complexity of the Linear-Time Mu-Calculus for Petri Nets*, In P. Azema and G. Balbo, editors, Application and Theory of Petri Nets, LNCS 1248, pp. 102-116.
- Haddad, S. and D. Poitrenaud (2001). *Checking Linear Temporal Formulas on Sequential Recursive Petri Nets*, in Proc. of the 8th International Symposium on Temporal Representation and Reasoning TIME-01, Italy, pp. 198-205.
- Hopcroft, J.E. and J.D. Ullman (1979). *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley.
- Jantzen, M. and R. Valk (1985). *The Residue of Vector Sets with Applications to Decidability Problems in Petri Nets*, Acta Informatica 21, 1985, pp. 643-674.
- Latvala, T. (2001). *Model Checking LTL properties of High-Level Petri Nets with Fairness Constraints*, in Proc. of the 22nd International Conference on Application and Theory of Petri Nets ICATPN 2001, LNCS 2075, pp. 242-262.
- Mayr, R (1998). *Decidability and Complexity of Model Checking Problems for Infinite-State Systems*, Ph.D. Thesis, Technischen Universität München.
- Reisig, W. (1985). *Petri Nets. An Introduction*, EATCS Monographs on Theoret. Comput. Sci., Springer-Verlag.
- Țiplea, F.L. and T. Jucan and C. Masalagiu (1991). *Conditional Petri net languages*, Journal of Information Processing and Cybernetics EIK 27, pp. 55-66.
- Țiplea, F.L. (1994). *On Conditional Grammars and Conditional Petri Nets*, in Mathematical Aspects of Natural and Formal Languages (Gh. Păun, ed.), World Scientific Series in Computer Science vol. 43, Singapore, pp. 431-456.
- Țiplea, F.L. and T. Jucan (1994). *Jumping Petri Nets*, Foundations of Computing and Decision Sciences 19(4), pp. 319-332.
- Țiplea, F.L. and E. Mäkinen (1997). *Jumping Petri Nets. Specific Properties*, Fundamenta Informaticae 32(3+4), pp. 373-392.
- Țiplea, F.L. and J. Desel (1999). *Petri Net Process Decomposition with Application to Validation*, Proceedings of the 6th Workshop "Algorithmen und Werkzeuge für Petrinetze", Frankfurt am Main (Germany), pp. 61-68.
- Țiplea, F.L. and A. Țiplea (1999). *Petri Net Reactive Modules*, Technical Report TR 1999-7, Institut für Informatik, Universität Augsburg, Augsburg (Germany), 50 pp.
- Țiplea, F.L. and C. Bădărău (2000). *A Note on Decidability of Reachability for Conditional Petri Nets*, Acta Cybernetica 14, pp 455-459.
- Vardi, M. and P. Wolper (1986). *Automata Theoretic Technique for Modal Logics and Programs*, Journal of Computer and Systems Sciences 32, pp. 183-221.
- Wolper, P. and B. Boigelot (1998). *Verifying Systems with Infinite but Regular State Spaces*, in Proceedings of the 10th International Conference on Computer Aided Verification, Lecture Notes in Computer Science 1427, pp. 88-97.
- Yen, H. (1992). *A Unified Approach for Deciding the Existence of Certain Petri Net Paths*, Information and Computation 96(1), pp. 119-137.

Faculty of Computer Science
 "Al.I.Cuza" University of Iași
 6600 Iași, Romania
 E-mail: {fttiplea,olivia}@infoiasi.ro

Received April 2004