On Temporal Properties of Nested Petri Nets

Leonid Dvoryansky
Department of Software Engineering
National Research University Higher School of Economics
Moscow, Russia
leo@mathtech.ru

Daniil Frumin
Department of Software Engineering
National Research University Higher School of Economics
Moscow, Russia
difrumin@edu.hse.ru

Abstract—Nested Petri nets is an extension of Petri net formalism with net tokens for modelling multi-agent distributed systems with complex structure. Temporal logics, such as CTL, are used to state requirements of software systems behaviour. However, in the case of nested Petri nets models, CTL is not expressive enough for specification of system behaviour. In this paper we propose an extension of CTL with a new modality for specifying agents behavior. We define syntax and formal semantics for our logic, and give small examples of its usage.

Index Terms—Petri nets, nested Petri nets, temporal logic, CTL

I. INTRODUCTION

Petri nets is a popular formalism for modelling concurrent systems. Different extensions of Petri nets are extensively studied in the literature. The most popular are coloured Petri nets [5]. Nested Petri nets [6] is a formalism for modelling hierarchical multi-agent systems. There is a variety of temporal logics for specifying behavioural properties of discrete systems, such as HML, CTL, LTL, μ-calculi [4], [1], [3]. However, they are not convenient for expressing some natural properties of nested Petri nets.

The paper is organized as follows. To start with, we give some necessary foundations of labelled transition systems and Petri nets. Then we describe nested the Petri nets formalism. We propose an extension of CTL with a new modality for specifying agents behavior. We define syntax and formal semantics for our logic, and give small examples of its usage.

II. BACKGROUND

Definition 1. A Labelled Transition System (LTS) is a tuple $(S, q_0, R, Act)$ where

- $S$ — a set of states (worlds);
- $Act$ — a set of actions;
- $q_0 \in S$ is an initial state;
- $R \subseteq S \times Act \times S$ is a transition relation.

For convenience we write $s \xrightarrow{a} s'$ instead of $(s, a, s') \in R$.

Definition 2. A Petri net (P/T-net) is a 4-tuple $(P, T, F, W)$ where

- $P$ and $T$ are disjoint finite sets of places and transitions, respectively;
- $F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs;
- $W : F \rightarrow N \setminus 0$ — an arc multiplicity function, that is, a function which assigns every arc a positive integer called an arc multiplicity.

We denote by $\tilde{W}$ the extension of $W$ by zero

$$\tilde{W}(x, y) = \begin{cases} n, & xFy \land W(x, y) = n \\ 0, & \neg xFy \end{cases}$$

A marking of a Petri net $(P, T, F, W)$ is a multiset over $P$, i.e. a mapping $M : P \rightarrow N$. By $M(N)$ we denote a set of all markings of a P/T-net $N$.

We say that the transition $t$ in a P/T-net $N = (P, T, F, W)$ is active in the marking $M$ iff for every $p \in\{p \mid (p, t) \in F\}$: $M(p) \geq \tilde{W}(p, t)$. An active transition may fire, resulting in a marking $M'$, such as for all $p \in P$: $M'(p) = M(p) - \tilde{W}(p, t) + \tilde{W}(t, p)$.

III. NESTED PETRI NETS

In this section we define nested Petri nets (NP-nets) [6]. For simplicity we consider here only two-level NP-nets, where net tokens are usual Petri nets.

Definition 3. A nested Petri net is a tuple $(\text{Atom}, \text{Expr}, \text{Lab}, \text{SN}, (EN_1, \ldots, EN_k))$ where

- Atom = Var $\cup$ Con $-$ a set of atoms;
- Lab is a set of transition labels;
- $(EN_1, \ldots, EN_k)$, where $k \geq 1$ — a finite collection of P/T-nets, called element nets;
- $\text{SN} = (P_{SN}, T_{SN}, F_{SN}, \nu, W, \Lambda)$ is a high-level Petri net where
  - $P_{SN}$ and $T_{SN}$ are disjoint finite sets of system places and system transitions respectively;
  - $F_{SN} \subseteq (P_{SN} \times T_{SN}) \cup (T_{SN} \times P_{SN})$ is a set of system arcs;
  - $\nu : P_{SN} \rightarrow \{EN_1, \ldots, EN_k\} \cup \{\bullet\}$ is a place typing function;
  - $W : F_{SN} \rightarrow \text{Expr}$ is an arc labelling function;
  - $\Lambda : T_{SN} \rightarrow \text{Lab} \cup \{\tau\}$ is a transition labelling function, $\tau$ is the special “silent” label;

The arc expression language Expr is defined as follows. Let $Con$ be a set of constants interpreted over $A = A_{net} \cup \{\bullet\}$ and $A_{net} = \{EN_i, m \mid \exists i = 1, \ldots, k : EN = EN_i, m \in M(EN_i)\}$, i.e. $A_{net}$ is a set of marked element nets. Let
Let \( \text{Var} \) be a set of variables. Then an expression in \( \text{Expr} \) is a multiset over \( \text{Con} \cup \text{Var} \). The arc labeling function \( \text{W} \) is restricted in such way that constants or multiple instances of the same variable are not allowed in input arc expressions of the transition, constants and variables in the output arc expressions should correspond to the types of output places, and each variable in an output arc expression of the transition should occur in one of the input arc expressions of the transition.

A marking \( M \) in an NP-net \( \text{NPN} \) is a function mapping each \( p \in \text{P}_{SN} \) to some (possibly empty) multiset \( M(p) \) over \( A \). By abuse of notation, a set of all markings in a NP-net \( \text{NPN} \) is denoted by \( \mathcal{M}(\text{NPN}) \).

A behavior of an NP-net consists of three kinds of steps. A system-autonomous step (resp. element-autonomous step) is a firing of a transition, labeled with \( \tau \), in the system net (resp. in one of the element nets).

An element-autonomous step is a transition firing according to the standard firing rules for P/T-nets.

To describe a system-autonomous step we need the concept of binding.

**Definition 4.** Let \( \text{Vars}(c) \) denote a set of variables in an expression \( e \in \text{Expr} \). For each \( t \in \text{T}_{SN} \) we define \( \text{W}(t) = \{ \text{W}(x,y) \mid (x,y) \in \text{F}_{SN} \wedge (x = t \vee y = t) \} \) – all expressions labelling arcs incident to \( t \).

A binding \( b \) of a transition \( t \) is a function \( b : \text{Vars}(\text{W}(t)) \rightarrow A \), mapping every variable in the \( t \)-incident arc expression to some value.

We say that a transition \( t \) is active w.r.t. a binding \( b \) iff \( \forall p \in \{ p \mid (p,t) \in \text{F}_{SN} \} \}: b(\text{W}(p,t)) \subseteq M(p) \). An active transition may fire (denote \( M \xrightarrow{\{b\}} M' \) yielding a new marking \( M'(p) = M(p) - b(\text{W}(p,t)) + b(\text{W}(t,p)) \) for each \( p \in \text{P}_{SN} \). An autonomous step in a net token changes only this token inner marking. An autonomous step in a system net can move, copy, generate, or remove tokens involved in the step, but doesn’t change their inner markings.

A (vertical) synchronization step is a simultaneous firing of a transition, labeled with some \( \lambda \in \text{Lab} \), in a system net together with firings of transitions, also labeled with \( \lambda \), in all net tokens involved in (i.e. consumed by) this system net transition firing. For further details see [6]. Note, however, that here we consider a typed variant of NP-nets, when a type of an element net is instantiated to each place.

**IV. Temporal Logics**

In this section we describe CTL – computational tree logic, which is widely used for specifying temporal properties of reactive systems.

A CTL formula is defined by the following grammar:

\[
\Phi ::= \text{true} \mid (\neg \Phi) \mid (\Phi \lor \Phi_2) \mid \text{EU}(\Phi_1, \Phi_2) \mid \text{AU}(\Phi_1, \Phi_2) \mid \text{EX}(\Phi) \mid p
\]

where \( p \) is an atomic proposition.

A CTL formula is interpreted over Kripke structures. Kripke structure is a labelled transition system (c.f. definition 1) where \( \text{Act} \) is a singleton \( \{ \tau \} \).

We can recursively define interpretation of a given CTL formula \( \phi \) over a Kripke structure \( K \) and a current state \( s \). We suppose some fixed interpretation of atomic propositions \( I : S \times A\text{P} \rightarrow \{ \text{true}, \text{false} \} \).

- \( (K,s) \models \text{true} \);
- \( (K,s) \models p \iff I(p,s) = \text{true} \);
- \( (K,s) \models (\phi_1 \lor \phi_2) \iff (K,s) \models \phi_1 \) or \( (K,s) \models \phi_2 \);
- \( (K,s) \models (\neg \phi) \iff (K,s) \not\models \phi \);
- \( (K,s) \models \text{EX}(\phi) \iff \exists s', \phi \implies \exists s'' \in R.(K,s'') \models \phi \);
- \( (K,s) \models \text{EU}(\phi_1, \phi_2) \iff \text{there exists a path } s_1 s_2 \ldots \text{ in } K \) (that is \( s_1 \xrightarrow{\tau} s_2, s_2 \xrightarrow{\tau} s_3, \ldots \) such that: \( s_1 = s \) and \( \exists n, (\forall j \in 0, n - 1. (K,s_j) \models \phi_1) \land (K,s_n) \models \phi_2 \));
- \( (K,s) \models \text{AU}(\phi_1, \phi_2) \iff \text{for every path } s_1 s_2 \ldots \text{ in } K \) the following holds: \( s_1 = s \) and \( \exists n, (\forall j \in 0, n - 1. (K,s_j) \models \phi_1) \land (K,s_n) \models \phi_2 \).

We can also define additional useful operator weak until: \( \text{AW}(\phi, \psi) = \neg \text{EU}(\phi \land \neg \psi, \psi \land \neg \psi) \), \( \text{EW}(\phi, \psi) = \neg \text{AU}(\phi \land \neg \psi, \neg \phi \land \neg \psi) \). Intuitively, if model satisfies \( \text{AW}(\phi, \psi) \) (resp. \( \text{EW}(\phi, \psi) \)) then for all paths (resp. there exists a path) in which either \( \phi \) is true until we encounters \( \psi \) or \( \phi \) is always true. The difference between \( \text{AW}(\psi, \phi) \) and \( \text{AU}(\psi, \phi) \) is that in the former it’s not necessary that \( \phi \) is reached.

**V. Temporal Properties of NP-nets**

Let’s consider the following example (Fig.1). Here the left net is a system net with the net token \( \alpha \) residing in \( p_1 \), and \( \alpha \) depicted in the right part of Fig.1.

![Fig. 1. NP-net NPN1](image)

LTS representing the behaviour of this NP-net is shown in Fig.2.

Imagine, we want to check that in every net token transitions \( t_1 \) and \( t_2 \) fire by turns. We could try the formula \( \text{AXAW}(\neg t_1, t_2)) \land (t_2 \implies \text{AXAW}(\neg t_2, t_1)) \). Although this approach might look attractive at the first sight, it does not work in many cases.

To show why our approach does not work here let’s take a closer look at the LTS (Fig.4) corresponding to \( \text{NPN}_2 \) (Fig.4).

Since \( t_1 \) appears “before” \( t_2 \) in LTS \( \Sigma_2 \), we can conclude that \( \text{NPN}_2 \) satisfies our formula. However, actually in the second net token \( t_2 \) fired before \( t_1 \). The problem is that transition firings in different net tokens are indistinguishable in our model: \( t_2 \) in figure 4 refers to the firing of \( t_2 \) in the net token which originally resided in \( q_3 \), but \( \Sigma_2 \) does not contain any information about that.

To handle such properties we introduce a new modality in the next section.
VI. nCTL

In the following section we present a solution to the problem described in the previous passage by introducing additional modality.

A. Syntax

To specify properties concerning states (markings), system transitions and transitions in element nets we introduce a logic with three categories of formulae. Just like in CTL we make use of path quantifiers in both state and transition formulae.

We define the syntax of nCTL with a fixed nested Petri net NPN in mind.

State formulae: $A ::= true \mid \neg A \mid (A_1 \lor A_2) \mid (\exists) \mid EU(A_1,A_2) \mid AU(A_1,A_2)$

Transition formulae: $B ::= true \mid \chi \mid \neg B \mid (B_1 \lor B_2) \mid (\forall) \mid EU(B_1,B_2) \mid AU(B_1,B_2)$

Element transition formulae: $C ::= true \mid \gamma \mid \neg C \mid (C_1 \lor C_2) \mid [A] \mid EU(C_1,C_2) \mid AU(C_1,C_2) \mid AX C$

Here true is a boolean constant, $\omega$ is a function, called a marking predicate, with the type $M \rightarrow \{true, false\}$, i.e., a predicate on the set of all markings. A function $\chi$ maps transitions $T$ of $SN$ to booleans. $\gamma$ is a predicate on set of all transitions of all element nets (we do not need a predicate on markings of element nets, since every marking of an element net can be characterized by a subset of $M$). $EU$ and $AU$ are familiar from the conventional CTL.

A nCTL formula is a well-formed state formula.

B. Examples

The idea of using both state and event modalities were first developed in ASK-CTL library for coloured Petri nets [2]. We extend this idea to NP-nets.

Intuitively, when we encounter an element transition sub-formula, we switch our interpreting context to an LTS of an element net. Nesting of the modalities allow us to switch back and forth between contexts. $[\phi]$ means that there exists a path in the LTS of the NP-net along which $\phi$ holds for every element net.

Now we can express the “switching” property for the NPN$_2$ (Fig.3): $[AU(\neg t_2,t_1) \land AG(t_1) \implies AX AU(\neg t_1,t_2)) \land AG(t_1) \implies AX AU(\neg t_1,t_2)]$. The $AU(\neg t_2,t_1)$ part is necessary to check whether $t_1$ is the first transition to be fired.

In order to properly verify whether the LTS, corresponding to NPN$_2$, is a model for our formula, we should change the way we construct the LTS. Firstly, we introduce a set $N = \{n_0, n_1, \ldots\}$ every member of which represents a single element net token (note that $N$ is different from $A_{net}$, since the latter contains only types of element nets together with their markings, while the former also distinguishes between individual tokens). Now we mark arcs in a LTS with tuples of the form $(\tau, n_i)$, where $\tau$ a name of transition in an element token $n_i \in N$. From here on in we use notation $\tau[n_i]$ to denote $(\tau, n_i)$.

The new LTS corresponding to the NPN$_2$ is shown in Fig.5. The element net tokens residing in $q_1$ (resp. $q_2$) is denoted as $n_0$ (resp. $n_1$). If we check the only path generated by transitions $\tau = (T_1, t_1[n_0], T_2, t_2[n_1], t_2[n_0])$ – we see that it does not satisfy our formula.

It is worth mentioning that our approach is not equivalent to model checking of element transition formulae on LTSs corresponding to element nets. This is caused by the fact...
that the LTS corresponding to an element net should be considered only w.r.t. transitions in the system net due to vertical synchronization. In addition, nCTL provides an ability to switch back to system context, based on the properties of an element net. Consider this example: $[(A \cdot t_1 \implies [\lambda_1]) \lor (A \cdot t_2 \implies [\lambda_2])]$. NP-net $N'$ models that formula if either $t_1$ fires in the next step in the element net and $N'$ reaches marking $M_1$ or $t_2$ fires in the next step in the element net and $T_2$ is enabled in the system net after that.

We need to construct LTS with respect to information about specific element nets. Let us consider some concrete problems connected with that.

Fig. 5. New LTS of $NPN_2$

![LTS diagram](image1)

Fig. 6. NP-net $NPN_3$

![NP-net diagram](image2)

Fig. 7. Net token $\kappa$ in $NPN_3$

![Token diagram](image3)

Fig. 8. LTS corresponding to $NPN_3$

![LTS diagram](image4)

VII. nCTL semantics

nCTL formulae are interpreted over a pair $(L_{NPN}, m)$, where $L_{NPN}$ is a LTS corresponding to $NPN$ and $m$ is a reachable marking of the latter. We say that $NPN$ satisfies formula $\phi$ iff $(L_{NPN}, m_0) \models \phi$, where $m_0$ is the initial marking of $NPN$. State-transition modalities allow us to switch between different types of formulae.

**Interpretation for state formulae:**

- $(L_{NPN}, m) \models \phi$ where $\phi = \ltrue$.
- $(L_{NPN}, m) \models \phi \iff \omega(m)$.
- $(L_{NPN}, m) \models \neg \phi \iff (L_{NPN}, m) \not\models \phi$.
- $(L_{NPN}, m) \models \phi_1 \lor \phi_2 \iff (L_{NPN}, m) \models \phi_1 \lor (L_{NPN}, m) \models \phi_2$.
- $(L_{NPN}, m) \models \phi \iff \exists a. m \xrightarrow{a} m' \land (L_{NPN}, (m, a, m')) \models \phi$.
- $(L_{NPN}, m) \models \phi \iff \exists a. m \xrightarrow{a} m' \land (L_{NPN}, (m, a, m')) \models \phi$.
- $(L_{NPN}, m) \models \phi \iff (L_{NPN}, m) \models A \phi$.
- $(L_{NPN}, m) \models \phi \iff (L_{NPN}, m) \models B \phi$.
- $(L_{NPN}, m) \models \phi \iff (L_{NPN}, m) \models C \phi$.

**Interpretation for transition formulae:**

- $(L_{NPN}, (m, a, m')) \models B \true$.
- $(L_{NPN}, (m, a, m')) \models B \chi \iff \forall \sigma = a_1a_2\cdots \in P_m \cdot \exists n < |\sigma| \cdot m \xrightarrow{a_1} m_1 \xrightarrow{a_2} m_2 \cdots m_{n-1} \xrightarrow{a_n} m_n \cdot (\forall i \in 0, n - 1 . (L_{NPN}, m_i) \models \phi_1) \land (L_{NPN}, m_n) \models \phi_2$.
- $(L_{NPN}, (m, a, m')) \models A \phi_1 \iff \exists \sigma = a_1a_2\cdots \in P_m \cdot \exists n \leq |\sigma| \cdot m_0 \xrightarrow{a_1} m_1 \xrightarrow{a_2} m_2 \cdots m_{n-1} \xrightarrow{a_n} m_n \cdot (\forall i \in 0, n - 1 . (L_{NPN}, m_i) \models \phi_1)$.
- $(L_{NPN}, (m, a, m')) \models B \phi_1 \iff (L_{NPN}, (m, a, m')) \models B \phi_1$.
- $(L_{NPN}, (m, a, m')) \models A \phi \iff (L_{NPN}, (m, a, m')) \models A \phi$.
- $(L_{NPN}, (m, a, m')) \models B \phi \iff (L_{NPN}, (m, a, m')) \models B \phi$.
- $(L_{NPN}, (m, a, m')) \models C \phi \iff (L_{NPN}, (m, a, m')) \models C \phi$.
- $(L_{NPN}, (m, a, m')) \models A \phi \iff (L_{NPN}, (m, a, m')) \models A \phi$.
- $(L_{NPN}, (m, a, m')) \models B \phi \iff (L_{NPN}, (m, a, m')) \models B \phi$.
- $(L_{NPN}, (m, a, m')) \models C \phi \iff (L_{NPN}, (m, a, m')) \models C \phi$.
From a practical perspective, nCTL logic enables us to state properties of agents in multiagent systems.

VIII. CONCLUSION AND FUTURE RESEARCH

The temporal logic nCTL, described in this paper, is an extension of CTL for specifying semantic properties of nested Petri nets. Logic nCTL may be helpful for describing behavioural properties of multi-agent systems with complex structure. nCTL allows to express both system net and element nets properties directly. This gives a straightforward way of formalizing NP-net specific temporal properties.

Our next goal is to develop an algorithm for constructing LTS describing semantics of a given NP-net. We also plan to investigate the expressive power of nCTL and study the possibility of developing effective verification algorithm for nested Petri nets.

REFERENCES


