Mapping Petri Net-Based Temporal Knowledge Representation Scheme into CP-Net Model

Slobodan Ribarić, Jan Šnajder
Faculty of Electrical Engineering and Computing, University of Zagreb
Unska 3, 10000 Zagreb, Croatia
E-mail: {slobodan.ribaric, jan.snajder}@fer.hr

***Abstract*— This paper presents a mapping of a knowledge representation scheme based on Petri Net with Time Tokens (PNTT) into a scheme based on Coloured Petri Net (CP-net). Basic module of the scheme uses Allen's interval-based temporal logic for time dependent event-driven reasoning. Mapping of the scheme into a CP-net model allows for the use of common tools such as *Design/CPN* for model construction and analysis. An example of a model mapped into a CP-net model is given and the reasoning process is illustrated.**

I. Introduction

The problem of knowledge representation and reasoning in temporally rich domains is one of the central problems in the field of artificial intelligence. Petri nets, due to their well-defined semantics and inherent concurrency, are well suited for this purpose, and various extensions for handling time have been developed [6], [10], [12]. The focus of this paper are the high-level Petri nets called Petri Nets with Time Tokens (PNTT), and the knowledge representation scheme based on Petri nets with time tokens (KRPTT) [11]. We describe the mapping of this scheme into a Coloured Petri Net (CP-net) [4], another high-level Petri net which, however, is not directly related to the concept of time.

In Section II we give formal definitions of the PNTT and the CP-net. The KRPTT scheme is described in Section III. Section IV describes the mapping of this scheme into a CP-net model. In Section V an example of the mapping is given and the reasoning process is illustrated. Concluding remarks are found in Section VI.

II. Petri net with time tokens vs. coloured Petri net

A. Petri net with time tokens

In this section we briefly describe the theoretical background needed for the understanding of the proposed knowledge representation scheme.

The PNTT is a 9-tuple $\mathrm{PNTT} = (P, T, I, O, \tau, M, \nu, \delta, \Omega)$, where P, T, I and O are components of the Place/Transition-net (PT-net) [9]. Function $\tau : P \to Q^* \cup \{\infty\}$ associates to each place a time duration of some activity or state. Set $M = \{m_1, m_2, \ldots, m_r\},\ r \geq 0$, is the set of *time tokens*.

Let $TD = \{\tau(p_1), \tau(p_2), \ldots, \tau(p_n)\},\ P = \{p_1, \ldots, p_n\}$, be the set of all time delays. Function $\nu : M \to \prod_{\infty > i \geq 1}(P \times TD)_i$, with Π denoting Cartesian product, maps each time token into a *time track*. A time track $\nu(m_i),\ m_i \in M$, is a sequence of places and time delays assigned to the time token m_i during the execution of the PNTT. Function $\delta : M \to Q^* \cup \{\infty\}$ maps each time token to the current amount of detaining time called *time accumulation*, also determined during execution of the PNTT. An initial time accumulation δ_0 can be assigned to a time token participating in the initial marking. It may be interpreted as an initial extension of the time interval of the corresponding place. By means of functions ν and δ, the structure of a time token is defined. It carries information about initial time accumulation, time of detainment at the places along its path and total accumulated time. The latter two are used for temporal reasoning.

An injective function $\Omega : P \to \wp(M)$, such that $\forall p_i, p_j \in P : i \neq j \Longrightarrow \Omega(p_i) \cap \Omega(p_j) = \emptyset$, is a *marking* of the PNTT. By Ω_0 we denote the initial marking, i.e. the initial assignment of time tokens to the places of the PNTT.

In PNTT, a transition is enabled if each of its input places contains at least as many time tokens as there are arcs from the place to the transition *and* if time of detainment of these tokens in places has elapsed. Such tokens are called *movable* time tokens. The firing of an enabled transition in the PNTT is performed automatically and immediately after the transition is enabled. By firing a transition in the PNTT, the tokens are distributed to its output places. The firing also changes the information of the corresponding time token (according to the time accumulation function ν). The number of tokens at input and output places of the transition is changed in accordance with the basic definition for PT-nets [9].

B. Coloured Petri net

A coloured Petri net (CP-net) [4], [5] extends the formalism of the PT-net by equipping each token with an attached value called the *token colour*. A CP-net allows for a more compact representation of a behaviourally equivalent PT-net. The structure of the colour may be arbitrary complex, resulting in different levels of reduction, i.e. mapping a PT-net into a CP-net is a one-to-many mapping.

The so-called *Timed CP-net* [5] further extends the CP-net formalism with the concept of time. For various reasons, not elaborated here, this extension is not adequate for the kind of time reasoning described in this paper.

The formal definition of the CP-net as given in [4] uses an expression representation and therefore presupposes that certain syntax exists together with a well-defined semantics.

The semantic categories are that of a *type*, the type of a variable v (denoted by $Type(v)$), the type of an expression expr (denoted by $Type(\mathsf{expr})$) and the set of variables in an expression expr (denoted by $Var(\mathsf{expr})$). A *binding* of a set variables V associates with each variable $v \in V$ an element $b(v) \in Type(v)$. An *evaluation* of an expression expr in a binding b (denoted by $\mathsf{expr} < b >$) is performed by substituting for each variable $v \in Var(\mathsf{expr})$ the value $b(v) \in Type(v)$ determined by the binding. An expression without variables is said to be a *closed* expression; it can be evaluated in all bindings, yielding the same value. For convenience, we use $\mathsf{expr}(v_1, \ldots, v_n, \underline{v_1}, \ldots, \underline{v_m})$ to denote an expression with variables $v_1, \ldots, v_n$ and parameters $\underline{v_1}, \ldots, \underline{v_m}$ (possibly intermixed). Parameters can be thought of as variables set to constant value and therefore not participating in a binding. Expression $\mathsf{v}(v)$ stands for the variable v itself.

In what follows we use S_{MS} to denote the set of all multi-sets over a non-empty set S, and $m(s) \in \mathbb{N}$ to denote the number of appearances of the element s in the multi-set $m \in S_{MS}$. The multi-set m is represented by a formal sum $\sum_{s\in S} m(s)`s$. Analogously, we evaluate an expression (either closed or not) of the form $\mathsf{m(s_1)`s_1} + \cdots + \mathsf{m(s_n)`s_n}$ to obtain a corresponding multi-set. A closed expression empty evaluates to an empty multi-set. By $\mathbb{B}$ we denote the Boolean type containing the elements $\{\top, \bot\}$. A closed expression true evaluates to $\top \in \mathbb{B}$.

A coloured Petri net is a 9-tuple $CPN = (\Sigma, P, T, A, N, C, G, E, I)$ satisfying the following requirements:

(i) Σ is a finite set of non-empty types, called *colour sets*.
(ii) P is a finite set of *places*.
(iii) T is a finite set of *transitions*.
(iv) A is a finite set of *arcs* such that $P \cap T = P \cap A = T \cap A = \emptyset$.
(v) $N : A \to P \times T \cup T \times P$ is a *node* function. It connects places to transitions and vice versa.
(vi) $C : P \to \Sigma$ is a *colour* function associating to each place a single colour set.
(vii) G is a *guard* function defined from T into expressions such that $\forall t \in T : [Type(G(t)) = \mathbb{B} \wedge Type(Var(G(t))) \subseteq \Sigma]$.
(viii) E is an *arc expression* function. It is defined from A into expressions such that $\forall a \in A : [Type(E(a)) = C(p_a(a))_{MS} \wedge Type(Var(E(a))) \subseteq \Sigma]$, where $p_a : A \to P$ maps each arc a to the component of $N(a)$ which is a place.
(ix) I is an *initialization* function. It is defined from P into closed expressions such that $\forall p \in P : [Type(I(p)) = C(p)_{MS}]$.

Set $X = P \cup T$ denotes the set of all nodes. Let a function $In : X \to \wp(X)$ map each node x to the set of its input nodes, $In(x) = \{x' \in X \mid \exists a \in A : \ N(a) = (x', x)\}$ and a function $Out : X \to \wp(X)$ map each node x to the set of its output nodes, $Out(x) = \{x' \in X \mid \exists a \in A : \ N(a) = (x, x')\}$.

In order for transition t to be enabled, there must exist a binding in which guard expression $G(t)$ evaluates to $\top \in \mathbb{B}$. In the same binding, the multi-sets of tokens of appropriate colours, as specified by the input arcs expressions, must be present in all input places of transition t. When enabled, transition may fire. Firing of transition t generates multi-sets of tokens in output places of t, and consumes multi-set of tokens from input places of t, as specified by the arc expressions. See [4] for a detailed descriptions and formal definitions of the CP-net enablement and firing rules.

III. Knowledge representation scheme based on PNTT

A. Knowledge representation

The PNTT is used as the main component of the KRPTT scheme. The scheme is defined as a five-tuple $\mathrm{KRPTT} = (\mathrm{PNTT}, \mathrm{TLM}, \alpha, \beta, F)$. PNTT is a Petri net with time tokens, and TLM is a temporal logic module based on Allen's temporal logic [1].

A bijective function $\alpha : P \to D \cup C_p$ is defined from set of places P to set of activities/states D and set of control places C_p. A function $\beta : T \to \Sigma \cup C_t$ is defined from set of transitions T to set of primitive events Σ and set of control transitions C_t. Functions α and β give a semantic interpretation of the PNTT.

A set F is a set of *flags*. A flag $f \in F$ is a 4-tuple $f = (p_a, p_b, tr, P_o)$. Elements $p_a, p_b \in P$ are places with time tokens to be tested by the TLM according to temporal relation tr. The tr is from a set of thirteen possible time interval relations defined in [1] and reproduced in Table I. We also allow tr to be a compound relation of the form $tr = tr_1 \vee tr_2 \vee \cdots \vee tr_n$. The set P_o is a set of output places in which new tokens will be generated by the TLM whenever tr holds for time tokens in places p_a and p_b. The generated tokens are called *control tokens* and can be treated as time tokens with zero accumulation time. The actual testing of relation tr is performed by inspecting information carried by time tokens in places p_a and p_b as soon as it becomes available.

The time reasoning in the KRPTT, which enables an agent to infer about temporal intervals and achieve time dependent goals, is based on the dynamic properties manifested by the *execution* of the PNTT. The execution of the PNTT is influenced by the TLM in such a way that the TLM puts control tokens in places defined according to the flag $f \in F$.

B. Time reasoning

The process of reasoning in the KRPTT scheme is called *time dependent event-driven reasoning*. It includes the ability to reason about time relationships among actions and states in the temporally rich domains, but it also includes the ability to determine the subsequent activities (states) based on initial or current activities (states) and temporal information associated to each of them. The reasoning process can be described as follows: The input is the initial marking of the KRPTT representation of the world. It determines current activities (states) and initial temporal relationships among the activities (states). By firing of enabled transitions time tokens are moved

TABLE I

ALLEN'S INTERVAL RELATIONS

Relation	Symbol	Symbol for Inverse	Pictorial Example
X before Y	$<$	$>$	xxx yyy
X equal Y	$=$	$=$	xxx yyy
X meets Y	m	mi	xxxyyy
X overlaps Y	o	oi	xxx ..yyy
X during Y	d	di	xxx yyyyyy
X starts Y	s	si	xxx yyyyy
X finishes Y	f	fi	xxx yyyyy

through the net. Depending on the path and the time duration of some activity or state, each time token has a "history" of execution represented by its time track. The PNTT execution is additionally controlled by the TLM, which tests and evaluates temporal relations between activities (states) specified by the flags. According to the results of the evaluation, tokens are put in control places also specified by the flags. This in turn directly influences the firing sequence, leading to a restricted set of possible firing sequences.

If some places are denoted as goal states, the scheme can conclude whether these time dependent goals may be achieved and detect sequence(s) of activities (states) which leads to these goals.

IV. MAPPING OF THE KRPTT INTO A CP-NET MODEL

We now describe the mapping of a complete KRPTT scheme into a functionally equivalent CP-net, with no extensions or modifications to the CP-net formalism. The main motivation behind this approach is to make it possible to use common CP-net tools such as *Design/CPN* [2] or *CPN Tools* for model construction and analysis. Also, the CP-net implementation of the KRPTT is preferred if the scheme is used as a part of a larger CP-net model (e.g. as part of a knowledge representation module of an intelligent agent), especially if it is to be analyzed and executed equally (at the same level of abstraction) as the rest of the model.

We begin by mapping the PNTT structure directly into a CP-net which we denote by $\mathrm{PNTT_{CPN}}$. Since the PNTT structure equals that of a PT-net, the mapping is straightforward. In order for the $\mathrm{PNTT_{CPN}}$ to become behaviourally equivalent to the PNTT net controlled by the TLM, we have to deal with the concept of (i) time tokens, (ii) the TLM, and (iii) the flags in an explicit manner.

A. Time and control tokens

Time tokens differ from ordinary tokens of the PT-net in that they carry additional information about time of detainment in places so far visited. The attachment of value to tokens comes natural in CP-nets, and for that purpose we introduce a new colour set $\sigma_{TT} \in \Sigma$. The colour set σ_{TT} is analogous to $\prod_i (P \times TD)_i$, i.e. it is a list of ordered pairs $P \times TD$ representing a time track. As described in Section II-A, firing of a transition changes the time information of the token involved in firing. In $\mathrm{PNTT_{CPN}}$, the time track is updated by means of an output arc expression, denoted $\mathsf{delay}(tt, \underline{\tau}, \underline{p})$. When evaluated in a binding, it generates a time token whose time track is extended in order to reflect time of detainment $\tau = \tau(p)$ in place p. More formally, for binding b:

$$< tt = \big((p_1, \tau(p_1)), \ldots, (p_i, \tau(p_i))\big) >$$

evaluation $\mathsf{delay}(tt, \underline{\tau(p_{i+1})}, \underline{p_{i+1}}) < b >$ gives:

$$\big((p_1, \tau(p_1)), \ldots, (p_i, \tau(p_i)), (p_{i+1}, \tau(p_{i+1}))\big) \in (\sigma_{TT})_{MS}.$$

We also introduce the colour set $\sigma_T \in \Sigma$ with a unique value denoted by the expression $()$. Colour set σ_T will be used for control tokens, which in fact correspond to time tokens with zero accumulated time.

B. The TLM and the flags

Let $p_{test} : F \to P$ and $p_{aff} : F \to P$ be projections:

$$\begin{array}{ll} \forall f = (p_a, p_b, tr, P_o) \in F : & p_{test}(f) = \{p_a, p_b\} \\ \forall f = (p_a, p_b, tr, P_o) \in F : & p_{aff}(f) = P_o \end{array}$$

which we call a set of *testing* and a set of *affecting* places of flag f, respectively.

In order to implement the functionality of the TLM in a CP-net, modifications and extensions to the structure of the TLM must be made. First, we introduce a blackboard, which we model with a single place denoted by $p_{bb} \in P$. Second, we connect the blackboard to each testing place p of the $\mathrm{PNTT_{CPN}}$, using one transition (denoted by $t^p_{cpy} \in T$) for each connection. The role of the transition t^p_{cpy} is to generate copies of time tokens in the blackboard place p_{bb}. We then introduce another place, denoted by $p_F \in P$, for storing flags from the KRPTT scheme. For the purpose of evaluation of temporal relations as specified by the flags, we create a transition $t_{test} \in T$. The input places of t_{test} are the blackboard p_{bb} (containing copies of tokens to be tested) and the flag storage p_F (containing the flags which specify the tests). Finally, we connect the transition t_{test} to each affecting place of the $\mathrm{PNTT_{CPN}}$. An example of a CP-net model representing a TLM is given in Fig. 4.

The CP-net model of the TLM functions as follows: Whenever a time token tt of the colour set σ_{TT} arrives at a testing place p, its copy tt' will be generated in the blackboard place p_{bb} by firing of transition t^p_{cpy}. Token tt' corresponds to the snapshot of token tt taken after it has been detained in p for time $\tau(p)$, i.e. after the token tt becomes movable (as defined in the PNTT).

Another problem must be addressed here: Since each testing place p is the input place of *both* the transition t^p_{cpy} (which is a part of CP-net representation of the TLM module) and a transition $t \in Out(p)$ (which is a part of the $\mathrm{PNTT_{CPN}}$), both t^p_{cpy} and t will be enabled simultaneously, resulting in a conflict situation (Fig. 1a). Due to the non-deterministic firing of enabled transitions, one has no guarantee that t^p_{cpy} will

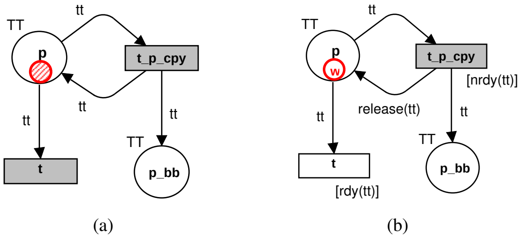

Fig. 1. A conflict situation (a) and its resolution (b). (Enabled transitions are shown shaded.)

fire prior to t, so it is possible for token tt to leave place p *before* the TLM has made a copy of it. To solve this problem, we extend the colour set σ_{TT} so that alone by the colour of tt one can say whether a copy of it has been made or not. Colour set σ_{TT} now corresponds to $(\prod_i (P \times TD)_i) \times S$. The first element is the time track, as described above, while the second $S = \{w, r\}$ indicates token status. Status w means token is waiting to be copied by the TLM, and status r means token has already been copied and is now ready to leave place p. We then equip both transitions t^p_{cpy} and t with logically complementary guards: To transition t^p_{cpy} we attach a guard expression $\mathsf{nrdy}(tt)$ which, in a binding, evaluates to $\top \in \mathbb{B}$ iff status of tt is w. To transition t we attach a guard expression $\mathsf{rdy}(tt)$ which evaluates to the logical complement of $\mathsf{nrdy}(tt)$. The firing of transition t^p_{cpy} changes the status of the time token tt from w to r, by means of an output arc expression denoted by $\mathsf{release}(tt)$. Fig. 1b shows these modifications. To ensure that the token status changes from r to w whenever token tt enters a testing place p, we provide another expression $\mathsf{delay}'(tt, \underline{\tau}, \underline{p})$ to be used on incoming arcs of place p. Expression $\mathsf{delay}'(tt, \underline{\tau}, \underline{p})$ is similar to the expression $\mathsf{delay}(tt, \underline{\tau}, \underline{p})$ introduced earlier, except its evaluation also changes the status of the time token tt from r to w.

C. Evaluation of temporal relations

To implement the testing of temporal relations as specified by the flags, a new colour set $\sigma_F \in \Sigma$ for flags is introduced. The structure of tokens from colour set σ_F corresponds to the 4-tuple $(p_a, p_b, \bigvee_i tr_i, P_o)$. Elements p_a, p_b and P_o were described in Section III-A. By $\bigvee_i tr_i$ we denote a compound temporal relation of the form $tr_1 \vee tr_2 \vee \cdots \vee tr_n$, where tr_i is any relation from Table I. For each flag $f \in F$ from the KRPTT scheme, a corresponding token ft from the colour set σ_F will initially be put in place p_F which is used as flag storage in our CP-net model. Testing of temporal relations is carried out by the transition t_{test}. The input places of t_{test} are the blackboard p_{bb} (from which tokens tt_a and tt_b are inspected), and the place p_F (from which flag ft will be consumed upon firing). For t_{test} to be enabled, tokens tt_a and tt_b must both be present in the blackboard place p_{bb} (i.e. information needed to evaluate the relation must be available), and the corresponding flag ft must exist in the flag storage. This correspondence is achieved by means of a guard expression, denoted by $\mathsf{canTest}(ft, tt_a, tt_b)$, and attached to transition t_{test}. It evaluates to $\top \in \mathbb{B}$ in a binding iff tokens tt_a and tt_b happen to be from testing places of the flag ft. If all these conditions are satisfied, the transition t_{test} is enabled and may fire. Set of output places of t_{test} is the set of all affecting places. An expression denoted by $\mathsf{test}(ft, tt_a, tt_b, \underline{p})$ is attached to each outgoing arc of the transition t_{test}. Its evaluation generates a single control token of the colour set σ_T iff relation tr specified by the flag ft holds between time intervals associated to tokens tt_a and tt_b, *and* if p is the affecting place of the flag ft. If not, evaluation of expression t_{test} does not generate a control token, but instead results in an empty multi-set.

D. Definition of the CP-net based KRPTT model

Mapping of the KRPTT elements into constructs of a CP-net as described above results in a CP-net model which we denote by $\mathrm{KRPTT_{CPN}}$. The $\mathrm{KRPTT_{CPN}}$ may be regarded as a composite CP-net consisting of two subnets: the $\mathrm{PNTT_{CPN}}$ and the CP-net representation of the TLM. The $\mathrm{KRPTT_{CPN}}$ is functionally equivalent to the KRPTT scheme. By this we mean that if the execution of the $\mathrm{KRPTT_{CPN}}$ is interpreted as a reasoning process, as described in Section III-B, its outcome will be identical to that of the original KRPTT scheme.

The formal definition of the structure of the $\mathrm{KRPTT_{CPN}}$ is the following: Let the scheme $\mathrm{KRPTT} = (\mathrm{PNTT}, \mathrm{TLM}, \alpha, \beta, F)$, where $\mathrm{PNTT} = (P, T, I, O, \tau, M, \nu, \delta, \Omega)$, be given. We define $\mathrm{KRPTT_{CPN}} = (\Sigma, P^*, T^*, A, N, C, G, E, I)$ as:

(i) $\Sigma = \{\sigma_T, \sigma_{TT}, \sigma_F\}$.

(ii) $P^* = P \cup \{p_{bb}, p_F\}$. For convenience we define: $P_{test} = \bigcup_{f \in F} p_{test}(f)$ and $P_{aff} = \bigcup_{f \in F} p_{aff}(f)$ as the set of all testing and the set of all affecting places, respectively.

(iii) $T^* = T \cup T_{cpy} \cup \{t_{test}\}$, where $T_{cpy} = \bigcup_{p \in P_{test}} t^p_{cpy}$.

(iv) $A = \{(p^*, t^*) \in (P \times T) \mid p^* \in I(t^*)\} \cup$
$\{(t^*, p^*) \in (T \times P) \mid p^* \in O(t^*)\} \cup$
$\{(p_{bb}, t_{test}), (t_{test}, p_{bb}), (p_F, t_{test})\} \cup$
$\bigcup_{p \in P_{test}} \{(p, t^p_{cpy}), (t^p_{cpy}, p), (t^p_{cpy}, p_{bb})\} \cup$
$\bigcup_{p \in P_{aff}} \{(t_{test}, p)\}$.

(v) $\forall a = (x_1, x_2) \in A : N(x_1, x_2) = (x_1, x_2)$.

(vi) $C(p_{bb}) = \sigma_{TT}$, $C(p_F) = \sigma_F$,
$$\forall p^* \in P : \; C(p^*) = \begin{cases} \sigma_{TT} & \text{if } \alpha(p^*) \in D \\ \sigma_T & \text{otherwise.} \end{cases}$$

(vii) $G(t_{test}) = \mathsf{canTest}(ft, tt_a, tt_b)$,
$\forall t^* \in T_{cpy} : \; G(t^*) = \mathsf{nrdy}(tt)$,
$$\forall t^* \in T : \; G(t^*) = \begin{cases} \mathsf{rdy}(tt) & \text{if } In(t^*) \in P_{test} \\ \mathsf{true} & \text{otherwise.} \end{cases}$$

(viii) $E(p_{bb}, t_{test}) = E(t_{test}, p_{bb}) = 1`\mathsf{tt_a} + 1`\mathsf{tt_b}\,(tt_a, tt_b)$,
$E(p_F, t_{test}) = \mathsf{ft}(ft)$,
$\forall p^* \in P_{test} : \; E(p^*, t^p_{cpy}) = E(t^p_{cpy}, p_{bb}) = \mathsf{tt}(tt)$,
$\forall p^* \in P_{test} : \; E(t^p_{cpy}, p^*) = \mathsf{release}(tt)$,
$\forall p^* \in P_{aff} : \; E(t_{test}, p^*) = \mathsf{test}(ft, tt_a, tt_b, \underline{p^*})$,

$$\forall(p^*,t^*) \in A \cap (P \times T):$$
$$E(p^*,t^*) = \begin{cases} \mathsf{tt}(tt) & \text{if } \alpha(p^*) \in D \\ () & \text{otherwise.} \end{cases}$$
$$\forall(t^*,p^*) \in A \cap (T \times P):$$
$$E(t^*,p^*) = \begin{cases} \mathsf{delay}'(tt, \underline{\tau(p^*)}, \underline{p^*}) & \text{if } p^* \in P_{test} \\ \mathsf{delay}(tt, \underline{\tau(p^*)}, \underline{p^*}) & \text{otherwise.} \end{cases}$$

(ix) $I(p_{bb}) = \mathsf{empty}$,
$I(p_F) = 1`\mathsf{ft}_1 + \cdots + 1`\mathsf{ft}_\mathsf{n}(\underline{ft_1}, \ldots, \underline{ft_n})$,
$\forall p^* \in P:\ I(p^*) = 1`\mathsf{tt}_1 + \overline{\cdots} + 1`\mathsf{tt}_\mathsf{r}(\underline{tt_1}, \ldots, \underline{tt_r})$.
The colour of token ft_i corresponds to the value of flag $f_i \in F$. Multi-set of time tokens $1`tt_1 + \cdots + 1`tt_i + \cdots + 1`tt_r$ corresponds to a set of time tokens $\Omega_0(p^*) = \{m_1, \ldots, m_i, \ldots, m_r\}$. The colour of token tt_i corresponds to $\big((p^*, \tau(p^*) + \delta_0(m_i)), s\big)$. If $p^* \in P_{test}$ then status $s = w$, otherwise status $s = r$.

The verification of contraints from Section II-B is straightforward and thus it is omitted.

Throughout this Section, we used the abstract syntax introduced in Section II-B to define the arc, guard and initialization expressions. The semantics of these expressions was described only informally. An appropriate programming language can now be used to formally define the colour sets and the expression semantics. A language well suited for that purpose is the CPN ML [3], an extension to the functional programming language SML [7], [13].

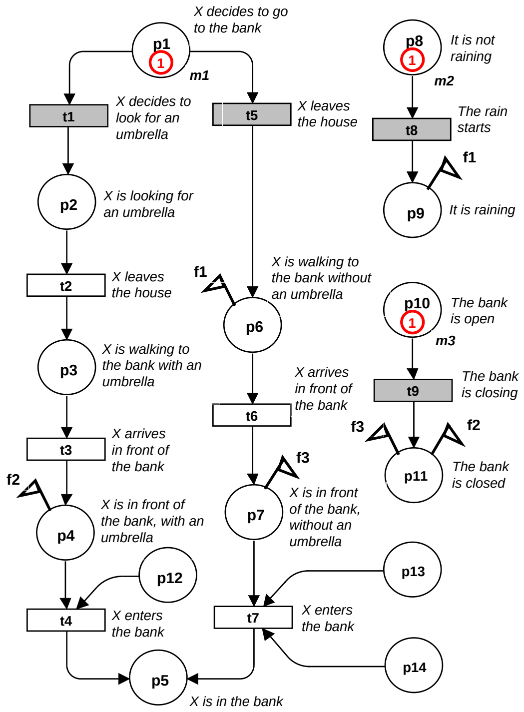

Fig. 2. A PNTT model of the described situation.

V. An Example

To illustrate the mapping we first represent the following situation from temporally rich domain in the KRPTT scheme. The example is taken from [8].

The beginning of the world is at 2:30 p.m. A person X wants to get to the bank before it closes. The bank is open until 3:00 p.m., and X needs 15 minutes to get to the bank. The forecast is rain, and X doesn't want to get wet. X can decide to take an umbrella or to go to the bank without it. If X decides to take an umbrella, finding it will take 5 minutes. At 2:39 it starts to rain.

The described situation can be represented by the KRPTT. The scheme will then be able to give answers to some questions, e.g. if X can get to the bank without getting wet and under what circumstances.

The first element of the KRPTT scheme is the PNTT. The PNTT corresponding to the situation above is shown in Fig. 2. It consists of three subnets representing the activities or states of the person (places $p_1, \ldots, p_7$), the rain (places p_8, p_9), and the bank (places p_{10}, p_{11}). Places p_{12}, p_{13}, p_{14} are control places. The definitions of T, I and O can be deduced from Fig. 2 and thus are omitted. Semantic interpretation of places and transitions (given by means of mappings α and β) can also be seen in Fig. 2. For control places we have $\{\alpha(p_{12}), \alpha(p_{13}), \alpha(p_{14})\} \subseteq C_p$. Time delays (representing minutes) are the following:

$$\tau(p_1) = \tau(p_4) = \tau(p_7) = \tau(p_{12}) = \tau(p_{13}) = \tau(p_{14}) = 0,$$
$$\tau(p_2) = 5, \tau(p_3) = \tau(p_6) = 15,$$
$$\tau(p_5) = \tau(p_9) = \tau(p_{11}) = \infty, \tau(p_8) = 9, \tau(p_{10}) = 30.$$

Timing constraints are enforced by a set of flags $F = \{f_1, f_2, f_3\}$, where $f_1 = (p_6, p_9, <, \{p_{13}\})$, $f_2 = (p_4, p_{11}, <, \{p_{12}\})$ and $f_3 = (p_7, p_{11}, <, \{p_{14}\})$. The "<" stands for the "before" time relation (see Table I). Set of time tokens is $M = \{m_1, m_2, m_3\}$.

Suppose that person X has decided to go to the bank 7 minutes after the situation at the beginning of the world (at 2:37 p.m.) We are now interested under what circumstances, if any, X can get to the bank before it closes but without getting wet. The query is formulated by considering place p_5 to be the goal state, and by defining the initial marking Ω_0 and the initial time accumulation δ_0 as follows:

$$\Omega_0(p_1) = \{m_1\},\ \Omega_0(p_8) = \{m_2\},\ \Omega_0(p_{10}) = \{m_3\},$$
$$\forall p \in P \setminus \{p_1, p_8, p_{10}\}:\ \Omega_0(p) = \emptyset;$$
$$\delta_0(m_1) = 7,\ \delta_0(m_2) = \delta_0(m_3) = 0.$$

This completes the definition of the KRPTT scheme representing the described situation. To obtain a functionally equivalent CP-net model, we define its structure according to definitions from Section IV-D. The resulting subnets, the $\mathrm{PNTT_{CPN}}$ and the CP-net representing the TLM, are shown in Fig. 3 and Fig. 4, respectively. For net construction and simulation the *Design/CPN* tool [2] was used. The arc, initialization and guard expressions are written in CPN ML. The actual definitions of colour sets and expressions, written in CPN ML and SML, are omitted.

The reasoning process is performed by the execution of the net as follows. Initially there are three tokens in the $\mathrm{PNTT_{CPN}}$ subnet, and three tokens (one for each flag) in place p_F of the TLM subnet. Initial color of these tokens is specified by

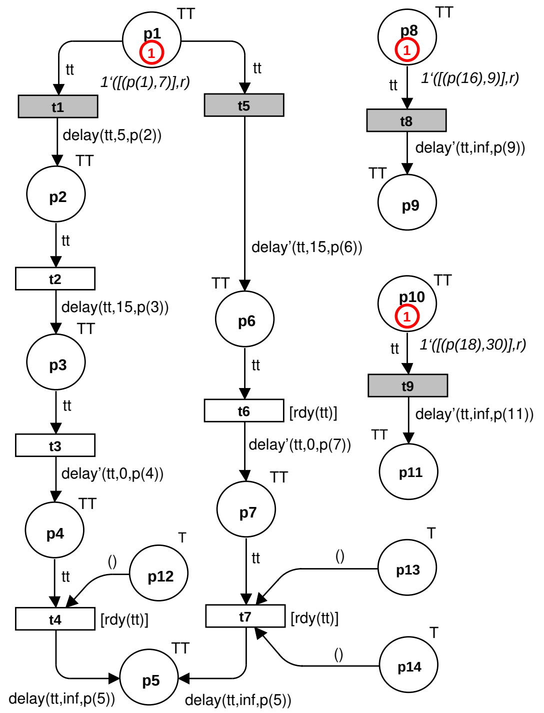

Fig. 3. A CP-net implementation of the PNTT model from Fig. 2.

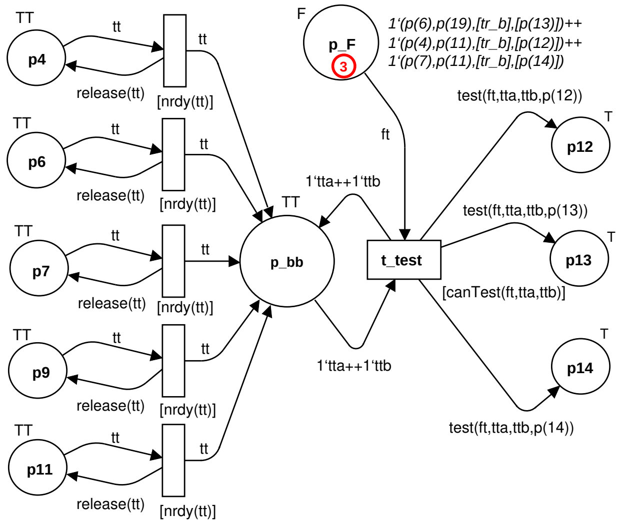

Fig. 4. A CP-net model representing the TLM for the described situation.

the initialization expressions of corresponding places (shown in italic). In the initial marking transitions t_1, t_5, t_8 and t_9 are enabled. Suppose t_1 fires first (*X decides to look for an umbrella*), followed by t_2 and t_3. Firing of each transition also updates the time track of the token. Additionally, firing of t_3 changes its status from r to w. Transition t_4 is not enabled because of both the guard (token status must be r for guard to evaluate to true) and the absence of token in place p_{12}. Since p_4 is a testing place, it is also a part of the TLM module which now can make a copy of the time token and put it in the blackboard place p_{bb}. At the same time the token colour is changed back to r as an indication that its copy has been made. Suppose t_8 and t_9 fire next. Since p_9 and p_{11} are both testing places, the TLM also generates copies of time tokens in place p_{bb}. Now transition t_{test} becomes enabled, as there exists a flag whose testing places are p_4 and p_{11}, and tokens from those places are indeed present in p_{bb}. Firing of t_{test} consumes the flag, and evaluates the specified temporal relation tr_b on each outgoing arc. The relation holds, but only in place p_{12} a control token is generated because p_{13} and p_{14} are not affecting places of the flag in question. Finally, t_4 becomes enabled making it possible to reach the goal state. The sequence of actions and states leading to the goal state can now be determined by inspecting the time track of token in place p_5.

VI. Conclusion

The KRPTT scheme is used for modeling and planning in temporally rich domains. The model integrates temporal information and metric, interval-based temporal logic, action specification and planning. The main advantages of the model are unified representation of different temporal information, simple modeling of different time relations among the actions and states given by the initial marking and initial time accumulation assigned to each time token, and well-defined semantics based on Petri Net theory for which formal analysis methods exist. The mapping of the KRPTT scheme into a functionally equivalent CP-net model allows for the use of common tools such as *Design/CPN* for model construction and analysis. The mapping is of importance if the scheme is used as a part of a larger CP-net model, especially when analyzed and executed at the same level of abstraction as the rest of the model.

References

[1] J. Allen, "Maintaining Knowledge about Temporal Intervals", *Communications of the ACM*, Vol. 26, No. 11, 1983, pp. 832–843.
[2] *Design/CPN Reference Manual for X-Windows, Version 2.0*, Meta Software Corporation, Cambridge, 1993.
[3] *Design/CPN Overview of CPN ML Syntax, Version 3.0*, Computer Science Dept., University of Aarhus, 1996.
[4] K. Jensen. *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use, Volume 1.* Springer-Verlag, Heidelberg, 1992.
[5] K. Jensen. *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use, Volume 2.* Springer-Verlag, Heidelberg, 1997.
[6] P. Merlin, "A Methodology for the Design and Implementation of Communicating Protocols", *IEEE Trans. on Communications*, Vol. 24, No. 6, June 1976, pp. 614–621.
[7] R. Milner, M. Tofte, R. Harper, D. MacQueen, *The Definition of Standard ML (Revised)*, The MIT Press, 1997.
[8] R. Pelavin, J. F. Allen, "A Formal Logic of Plans in Temporally Rich Domains", In *Proc. of the IEEE*, Vol. 74, No. 10, October 1986, pp. 1364–1382.
[9] J. L. Peterson, *Petri Net Theory and the Modeling of Systems*, Prentice-Hall, 1981.
[10] C. Ramchadani, "Analysis of Asynchronous Concurrent Systems by Petri Nets", *Ph.D. Dissertation*, M.I.T. Cambridge, 1974.
[11] S. Ribarić, B. D. Bašić, "Temporal Knowledge Representation and Reasoning Based on Petri Nets with Time Tokens", In *Proc. of Industrial Applications in Power Systems, Computer Science and Telecommunications*, Vol. 1, 1996, pp. 131–135.
[12] I. Suzuki, J. Lu, "Temporal Petri Nets and Their Application to Modeling and Analysis of a Handshake Daisy Chain Arbiter", *IEEE Trans. on Computers*, Vol. 38, No. 5, May 1989, pp. 696–704.
[13] J. D. Ullman, *Elements of ML Programming*, Prentice-Hall, 1998.