2

Modelling and Analysis of Real-Time Systems with RTCP-Nets

Marcin Szpyrka
AGH University of Science and Technology, Krakow
Poland

1. Introduction

RTCP-nets (Real-Time Coloured Petri nets, (Szpyrka 2006a), (Szpyrka & Szmuc 2006c)) are a subclass of timed coloured Petri nets (CP-nets, (Jensen 1992-1997)) defined for modelling and analysis of real-time systems. In comparison to timed CP-nets, RTCP-nets use a different time model, transitions' priorities and they are forced to fulfil some structural restrictions. These characteristics of RTCP-nets enable designers direct modelling of elements typical for concurrent programming (e.g. in Ada programming language, (Barnes 2006)), such as task priorities, timeouts, etc.
Formal methods (Cheng 2002) are used in the development of embedded systems for design, specification, validation, and verification of such systems. The use of formal methods can reduce the amount of testing and ensure more dependable products (Sommerville 2004). Especially, this is very important for safety-critical systems that may result in injury, loss of life or serious environmental damage upon their failure. A wide class of real time systems perform on the basis of a set of rules, which are used to compute outputs in response to current state of inputs that are monitored in such system environment. This set of rules specified in the analysis phase as functional requirements may be formally described, and then incorporated into the system model.
The presented approach uses RTCP-nets as modelling language for safety-critical systems. The modifications defining this subclass were introduced in order to improve modelling and verification means in the context of analysis and design of embedded systems. Especially, this technique has mostly been concerned with relatively small, critical kernel systems. RTCP-nets have been also prepared for modelling of embedded systems incorporating rule-based systems. A rule-based system in decision table form can be simply included into a model.
Another advantage of RTCP-nets is relatively simple transformation from a formal model into an implementation in Ada 2005 programming language. Such an implementation is done with the use of so-called Ravenscar profile (Burns et al. 2003). The profile is a subset of Ada language. It has been defined to allow implementation of safety-critical systems in Ada.
The goal of the chapter is to present the most important parts of the RTCP-nets theory and to describe the possibilities of practical applications of the nets. The chapter is organized as follows. The first section deals with a formal definition of RTCP-nets. The behaviour of the nets is presented in details so as to emphasize the differences between RTCP-nets and CP-nets. This part of the chapter is illustrated with an example of a non-hierarchical RTCP-net

(an example of a simple train protection systems).
The second section describes the analysis methods. It focuses on coverability graphs that are typical for RTCP-nets. If a net is strongly bounded, it is possible to construct a finite coverability graph that represents the set of all reachable states regardless of the fact the set is finite or infinite. Such a graph contains only one node for each equivalence class of the coverability relation. Not only can one use such a graph for the analysis of typical Petri nets' properties such as boundedness, liveness or fairness, but it also may be used for verification of timing properties, which are very important for most real-time embedded systems.
The last section deals with practical aspects of modelling with RTCP-nets. To speed up and facilitate drawing of more complex models the so-called *canonical form* of hierarchical RTCP-nets has been defined. The canonical form is shortly described in this section and an RTCP-net model of a real size railway traffic management system for a train station is presented to illustrate the possibilities of modelling with the nets.
The chapter is concluded with a short summary that describes possibilities of semiautomatic generation of an Ada 2005 source code from RTCP-nets models in canonical form.

2. RTCP- nets - basic notions

The definition of RTCP-nets is based on the definition of non-hierarchical timed CP-nets presented in (Jensen 1992-1997), but a few differences between timed CP-nets and RTCP-nets can be pointed out:

- Each transition has a priority value attached. The use of priorities allows direct modelling of deterministic choice.
- The set of arcs is defined as a relation due to the fact that multiple arcs are not allowed. Each arc has two expressions attached: a weight expression and a time expression. For any arc, each evaluation of the arc weight expression must yield a single token belonging to the type (colour) that is attached to the corresponding place; and each evaluation of the arc time expression must yield a non-negative rational value.
- The time model used by RTCP-nets differs from the one used by timed CP-nets. Time stamps are attached to places instead of tokens. Any positive value of a time stamp describes how long a token in the corresponding place will be inaccessible for any transition. A token is accessible for a transition, if the corresponding time stamp is equal to or less than zero. For example, if the stamp is equal to -3, it means the token is 3 time-units old. It is possible to specify how old a token should be so that a transition may consume it.

For any variable v, $\mathcal{T}(v)$ will be used to denote the *type* of the variable i.e. the set of all admissible values, the variable can be associated with. Let *x* be an expression. $\mathcal{V}(x)$ will denote the set of all variables in the expression *x*, and $\mathcal{T}(x)$ will denote the *type* of the expression, i.e. the set of all possible values that can be obtained by evaluating of the expression. For any given set of variables *V*, the type of the set of variables is defined as follows: $\mathcal{T}(V) = \{\mathcal{T}(v) : v \in V\}$.
Let *Bool* denote the boolean type (containing the elements *{false,true}*, and having the standard operations from propositional logic). Let $\mathbb{N} = \{1, 2, \ldots\}$, $\mathbb{Q}$, $\mathbb{Q}^+$ denote the set of natural, rational and non-negative rational numbers respectively. For an arc *a, P(a)* and *T(a)* will be used to denote the *place node* and the *transition node* of the arc, respectively.
Definition 1. An *RTCP-net* is a tuple $\mathcal{N} = (\Sigma, P, T, A, C, G, I, E_M, E_S, M_0, S_0)$ satisfying the following requirements.

1. Σ is a non-empty finite set of non-empty *types (colour sets).*
2. P is a non-empty finite set of *places.*
3. T is a non-empty finite set of *transitions* such that $P \cap T = \emptyset$.
4. $A \subseteq (P \times T) \cup (T \times P)$ is a *set of arcs.*
5. $C: P \rightarrow \Sigma$ is a fype *function,* which maps each place to its type.
6. G is a guard *function,* which maps each transition to an expression such that: $\forall t \in T: \mathcal{T}(G(t)) \subseteq Bool \wedge \mathcal{T}(\mathcal{V}(G(t))) \subseteq \Sigma$.
7. $I: T \rightarrow \mathbb{N} \cup \{0\}$ is a *priority function,* which maps each transition to a non-negative integer called *transition priority.*
8. E_M is an *arc expression function,* which maps each arc to a weight expression such that: $\forall a \in A: \mathcal{T}(E_M(a)) \subseteq C(P(a)) \wedge \mathcal{T}(\mathcal{V}(E_M(a))) \subseteq \Sigma$.
9. E_S is an arc time *expression function,* which maps each arc to a time expression such that: $\forall a \in A: \mathcal{T}(E_S(a)) \subseteq \mathbb{Q}^+ \cup \{0\} \wedge \mathcal{T}(\mathcal{V}(E_S(a))) \subseteq \Sigma$,
10. M_0 is an *initial marking,* which maps each place to a multiset $M_0(p) \in 2^{C(p)^*}$, where $2^{C(p)^*}$ denotes the set of all multisets over the set *C(p).*
11. $S_0: P \rightarrow \mathbb{Q}$ is an *initial time stamp function,* which maps each place to a rational value called *initial time stamp.*

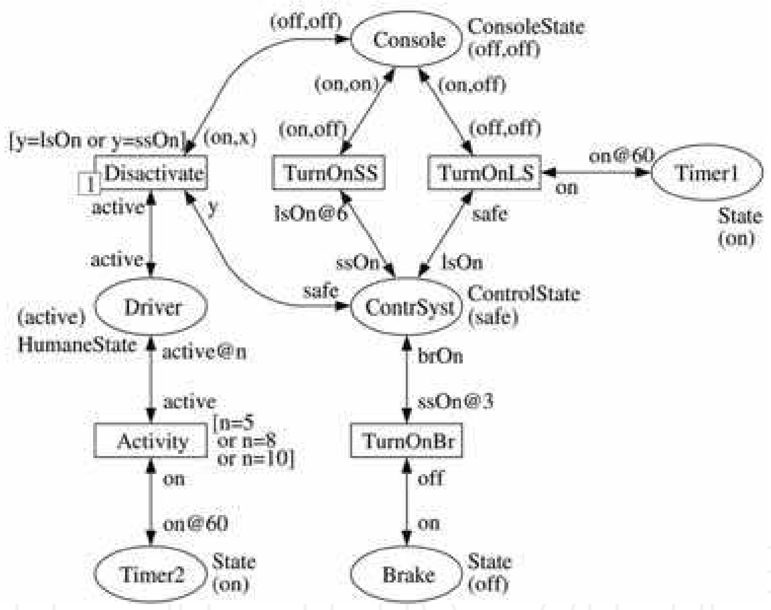


Fig. 1. Model of a simple ATS system

A model of a simple Automatic Train Stop (ATS) system is used to introduce main features of RTCP-nets. In the ATS system, a light signal is turned on every 60 seconds to check whether the driver controls the train. If the driver fails to acknowledge the signal within 6 seconds, a sound signal is turned on. Then, if the driver does not disactivate the signals within 3 seconds, using the acknowledge button, the emergency brakes are applied automatically to stop the train. A model of such a system is shown in Fig. 1. More information on using RTCP-nets for modelling train protection systems can be found in (Szpyrka & Szmuc 2006b).

The RTCP-net presented in Fig. 1 contains six places: *ContrSyst* (the control element of the ATS system), *Console* (to display warning signals), *Brake, Driver, Timerl* and *Timer2;* and five transitions: *TurnOnLS* (turn on light signal), *TurnOnSS* (turn on sound signal), *TurnOnBr* (turn on brake), *Disactivate* (driver disactivates warning signals) and *Activity* (to introduce into model some delays of the driver response). Initial markings are placed into parenthesis and initial time stamps equal to 0 are omitted. The transition's *Disactivate* priority is equal to 1, while other transition's priorities are equal to 0. The weight and time expressions are separated by the @ sign. If a time expression is equal to 0 it is omitted. Each arc with double arrows stands for a pair of arcs.

Definition 2. A *marking* of an RTCP-net $\mathcal{N}$ is a function *M* defined on the set of places *P,* such that: $\forall p \in P: M(p) \in 2^{C(p)^*}$. A *time stamp function* is a function *S* defined on the set of places *P,* such that: $\forall p \in P: S(p) \in \mathbb{Q}$.

If we assume that *P* is ordered set, both a marking *M* and a time stamp function *S* can be represented by vectors with $|P|$ entries. Therefore, the term a *time stamp vector* (or a *time vector)* will be used instead of a time stamp function.

Definition 3. A state of an RTCP-net is a pair (*M, S),* where *M* is a marking and *S* is a time stamp vector. The *initial state* is the pair (M_0, S_0).

Let's consider the net presented in Fig. 1 and let the set of places be ordered as follows *P* = {*ContrSyst, Timer1, Console, Brake, Driver, Timer2*}. The initial state of the considered net is as follows:

$$M_0 = (\text{safe, on, (off, off), off, active, on}),$$
$$S_0 = (0,0,0,0,0,0). \quad (1)$$

Let $X = P \cup T$ denote the set of all nodes of an RTCP-net and $x \in X$. *In(x)* and *Out(x)* denote the set of *input and output nodes* of the node *x,* i.e. $In(x) = \{y \in X: (y, x) \in A\}$ and $Out(x) = \{y \in X: (x, y) \in A\}$. Let $\mathcal{V}(t)$ be the set of variables that occur in the expressions of arcs surrounding the transition *t* and in the guard of the transition.

Definition 4. A *binding of* a transition $t \in T$ is a function *b* defined on $\mathcal{V}(t)$, such that: $\forall v \in \mathcal{V}(t): b(v) \in T(v) \wedge G(t)_b = true$.

Intuitively, a binding of a transition *t* is a substitution that replaces each variable of $\mathcal{V}(t)$ with a value of the corresponding type, such that the guard evaluates to *true.* The set of all bindings of a transition *t* is denoted by $\mathcal{B}(t)$. $G(t)_b$ denotes the evaluation of the guard expression in the binding *b.* Similarly, $E_M(p,t)_b$ and $E_S(p,t)_b$ denote the evaluation of the weight and the time expression in the binding *b,* respectively.

Definition 5. A transition $t \in T$ is *enabled* in a state (M, S) in a binding *b* iff the following conditions hold:

$$\begin{aligned} &\forall p \in In(t): E_M(p,t)_b \in M(p) \wedge E_S(p,t)_b \leq -S(p), \\ &\forall p \in Out(t): S(p) \leq 0, \end{aligned} \quad (2)$$

and for any transition $t' \neq t$ that satisfies the above conditions in some binding $b' \in \mathcal{B}(t')$, $I(t') \leq I(t)$ or $In(t) \cap In(t') = Out(t) \cap Out(t') = \emptyset$.

It means that a transition is enabled if all input places contain suitable tokens and have suitable time stamps, all output places are accessible and no other transition with a higher priority strives for the same input or output places.

A transition $t \in T$ is enabled in a state (*M, S)* if it is enabled in the state (*M, S)* in one of its bindings. If a transition $t \in T$ is enabled in a state (M_1, S_1) in a binding *b* it may *fire,* changing the state (M_1, S_1) to another state (M_2, S_2), such that $M_2(p) = M_1(p) - \{E_M(p,t)_b\} \cup \{E_M(t,p)_b\}$, $(\{E_M(x,y)_b\} = \emptyset \text{ if } (x,y) \notin A)$, and

$$S_2(p) = \begin{cases} E_S(t,p)_b & \text{for } p \in Out(t), \\ 0 & \text{for } p \in In(t) - Out(t), \\ S_1(p) & \text{otherwise.} \end{cases} \tag{3}$$

In other words, if a transition fires, it removes one token from each input place, adds one token to each output place, sets time stamps of input places to 0 and sets time stamps of output places to values specified by time expressions of arcs leading from the transition to the places.

If a transition $t \in T$ is enabled in a state (M_1, S_1) in a binding *b* and a state (M_2, S_2) is derived from firing of the transition, then we write $(M_1, S_1) \xrightarrow{(t,b)} (M_2, S_2)$. The binding *b* will be omitted if it is obvious or redundant.

Two transitions *Activity* and *TurnOnLS* are enabled in the initial state. The first transition is enabled in three different bindings: $b_1 = (5/n)$ (the value of the variable *n* is equal to 5), $b_2 = (8/n)$ and $b_3 = (10/n)$, while the second one is enabled in the binding *b* = *()* (a trivial binding). For example, the result of firing of the transition *TurnOnLS* in the initial state is the state (M_1, S_1), where:

$$\begin{aligned} &M_1 = (lsOn, on, (on, off), off, active, on), \\ &S_1 = (0, 60, 0, 0, 0, 0). \end{aligned} \tag{4}$$

A global clock is used to measure time. Every time the clock goes forward, all time stamps are decreased by the same value.

Definition 6. Let *(M, S)* be a state and $e = (1, 1, \ldots, 1)$ a vector with $|P|$ entries. The state (*M*, *S*) is changed into a state (*M*', *S*') by a passage *of time* $\tau \in \mathbb{Q}^+$, denoted by $(M, S) \xrightarrow{\tau} (M', S')$, iff $M = M'$ and the passage of time τ is possible, i.e., no transition is enabled in any state *(M, S")*, such that: $S'' = S - \tau' \cdot e$, for $0 \leq \tau' < \tau$ and $\tau' \in \mathbb{Q}^+$.

The result of firing of transitions *TurnOnLS* and *Activity* (in binding b_2) is the state *(M_2,S_2)*, where $M_2 = M_1$ and $S_2 = (0, 60, 0, 0, 8, 60)$. None transition is enabled in the state but it is possible a passage of time $\tau = 6$ that leads to the state (M_2, S_2'), where $S_2' = (-6, 54, -6, -6, 2, 54)$. A timeout occurs in this state. A token in the place *Console* is 6 seconds old (the driver did not response within 6 seconds), so the transition *TurnOnSS* will fire.

A *firing sequence* of an RTCP-net $\mathcal{N}$ is a sequence of pairs $\alpha = (t_1, b_1), (t_2, b_2), \ldots$, such that b_i is a binding of the transition t_i for $i = 1, 2, \ldots$ The firing sequence is *feasible* from a state (M_1, S_1) iff there exists a sequence of states such that:

$$(M_1, S_1) \xrightarrow{\tau_1} (M_1, S_1') \xrightarrow{(t_1,b_1)} (M_2, S_2) \xrightarrow{\tau_2} \ldots \xrightarrow{(t_n,b_n)} (M_{n+1}, S_{n+1}) \xrightarrow{\tau_{n+1}} \ldots \tag{5}$$

For the sake of simplicity, we will assume that there is at most one passage of time (sometimes equal to 0) between firings of two consecutive transitions. A firing sequence may be finite or infinite. The set of all firing sequences feasible from a state (*M*, *S*) is denoted by $\mathcal{L}(M, S)$.

A state (*M*', *S*') is *reachable* from a state (*M*, *S*) iff there exists a finite firing sequence α feasible from the state (*M*, *S*) and leading to the state (*M*', *S*'). In such case, we can also say that the marking *M*' is *reachable* from the marking *M*. The set of all states that are reachable from (*M*, *S*) is denoted by $\mathcal{R}(M,S)$, while $\mathcal{R}(M)$ denotes the set of all markings reachable from the marking *M*.

3. Analysis of RTCP-nets

A major strength of Petri nets is their support for analysis of many properties and problems associated with concurrent systems. Three types of properties are distinguished for RTCP-nets: boundedness, liveness and timing ones.

Definition 7. Let an RTCP-net $\mathcal{N}$, a place $p \in P$, *a* multiset $X \in 2^{C(p)^*}$and a non-negative integer *k* be given.

1. *k* is *upper integer bound* for p iff $\forall (M,S) \in \mathcal{R}(M_0,S_0)\colon |M(p)| \leq k$.
2. X is *upper multiset bound* for p iff: $\forall (M,S) \in \mathcal{R}(M_0,S_0)\colon M(p) \leq X$.

Lower bounds are defined analogously. A place p is said to be *bounded* if it has an upper integer bound. If the upper integer bound is equal to one, the place is said to be *safe*. A place p is said to be *strongly bounded* if it has a finite upper multiset bound. An RTCP-net $\mathcal{N}$is said to be bounded if each place $p \in P$ has an upper integer bound. Safe and *strongly bounded* RTCP-nets are defined analogously.

An RTCP-net is *conservative* iff the number of tokens in the net remains constant. An RTCP-net is conservative with respect to a weighting vector $w = (w_1, w_2, \ldots, w_n)$, where $w_i > 0$, $i = 1, 2, \ldots, n$, iff the weighted number of tokens remains constant, i.e. $\forall (M,S) \in \mathcal{R}(M_0,S_0)\colon \sum_{i=1}^{n} w_i |M(p_i)| = \sum_{i=1}^{n} w_i |M_0(p_i)|$.

The concept of liveness is closely related to the complete absence of deadlocks. Five different levels of liveness can be defined for Petri nets (see (Murata 1989)).

Definition 8. Let an RTCP-net $\mathcal{N}$ be given. A transition $t \in T$ is said to be:

(0) *dead* (*$\mathcal{L}0$-live*) if t does not appear in any firing sequence in $\mathcal{L}(M_0,S_0)$.
(1) *potentially fireable* (*$\mathcal{L}1$-live*) if t appears at least once in some firing sequence in $\mathcal{L}(M_0,S_0)$.
(2) *$\mathcal{L}2$-live* if, given any positive integer k, t appears at least k times in some firing sequence in $\mathcal{L}(M_0,S_0)$.
(3) *$\mathcal{L}3$-live* if t appears infinitely often in some firing sequence in $\mathcal{L}(M_0,S_0)$.
(4) *live* (*$\mathcal{L}4$-live*) if t is $\mathcal{L}1$-live for each state $(M,S) \in \mathcal{R}(M_0,S_0)$.

An RTCP-net is said to be *$\mathcal{L}k$-live* if each transition of the net is *$\mathcal{L}k$-live*, $k = 0, \ldots, 4$.

Definitions 9. Let an RTCP-net $\mathcal{N} = (\Sigma, P, T, A, C, G, I, E_M, E_S, M_0, S_0)$ be given. A marking *M* is said to be dead if the net $\mathcal{N}' = (\Sigma, P, T, A, C, G, I, E_M, E_S, M, S_0)$ is dead. A state (*M*, *S*) is said to be dead if the marking *M* is dead.

Live markings and states are defined analogously. A live net does not guarantee that each transition fires as often as the others. Some transitions may be *starved by* others.

Definition 10. Let an RTCP-net $\mathcal{N}$ and a firing sequence $\alpha \in \mathcal{L}(M_0,S_0)$ be given. A firing sequence α is said to be *fair* if it is either finite or infinite and each transition $t \in T$ appears infinitely often in α. The net $\mathcal{N}$ is said to be *fair* if every firing sequence $\alpha \in \mathcal{L}(M_0,S_0)$is fair.

Definition 11. The *duration of* a firing sequence $\alpha = (t_1, b_1), (t_2, b_2), \ldots$ is the sum:

$$\delta(\alpha) = \sum_i \tau_i, \tag{6}$$

where values τ_i, $i = 1, 2, \ldots, n-1$ denote passages of time between consecutive states (see

equation (5)).
Definition 12. Let (*M*, *S)* and (*M*', *S*') be the states of an RTCP-net $\mathcal{N}$ such that $(M', S') \in \mathcal{R}(M, S)$. A *time of transition from* the state (M, S) to (M', S'), denoted by $\delta((M, S), (M', S'))$, is the duration of any sequence a leading from the state (M, S) to (M', S').
The duration of a firing sequence is unambiguous, while a time of transition from one state to another is not. If there are a few firing sequences leading from the state *(M, S)* to *(M',S'),* we receive a few possibly different times of transition between these states. The most important ones are the minimal and maximal times of transition.
Analysis of RTCP-nets may be carried out using reachability graphs. The set of reachable states $\mathcal{R}(M_0, S_0)$ is represented as a weighted, directed graph. Each node corresponds to a unique state, consisting of a net marking and a time vector, such that the state is a result of firing of a transition. Each arc represents a change from a state (M_i, S_i) to a state (M_j, S_j) resulting from a passage of time $\tau \geq 0$ and a firing of a transition *t* in a binding $b \in \mathcal{B}(t)$.
Let's consider the net presented in Fig. 1. None transition is enabled in the state (M_2, S_2), but it is possible a passage of time $\tau = 6$ that leads to the state (M_2, S_2'). The transition *TurnOnSS* is enabled in the state and its firing leads to the state (M_3, S_3), where:

$$M_3 = (ssOn, on, (on, on), off, active, on),\\ S_3 = (0, 54, 0, -6, 2, 54). \qquad (7)$$

Thus, in the reachability graph, there will be nodes for the states (M_2, S_2) and (M_3, S_3), and an arc going from (M_2, S_2) to (M_3, S_3) with label $((TurnOnSS, ()), 6)$.
A finite reachability graph may be used to verify the RTCP-nets' properties presented in this section. Analysis of boundedness and conservativeness properties may be carried out by using markings of the graph nodes, while analysis of liveness and fairness properties may be carried out by using labels of arcs. Each label of an arc is a pair of a transition with its binding and a passage of time. The second element of a pair can be treated as the weight of the arc. Thus, arcs' weights capture the time taken by transition from one state to the next. (We consider only states that are results of transitions' firing). Using the reachability graph, one can find the minimal and maximal times of transition from one state to another. To do this we can use typical algorithms for finding the shortest or longest paths between two nodes in a directed graph (multigraph). However, a reachability graph for an RTCP-nets may be infinite even though the net is strongly bounded. In such a case it is not very useful for analysis purposes. More detail description of reachability graphs can be found in (Szpyrka 2006a).
One of the main advantages of strongly bounded RTCP-nets (in practical applications RTCP-nets are usually strongly bounded) is the possibility to present the set of reachable states of an RTCP-net using a finite coverability graph. Such a graph can be used to verify most of the RTCP-net's properties, including the timing ones.

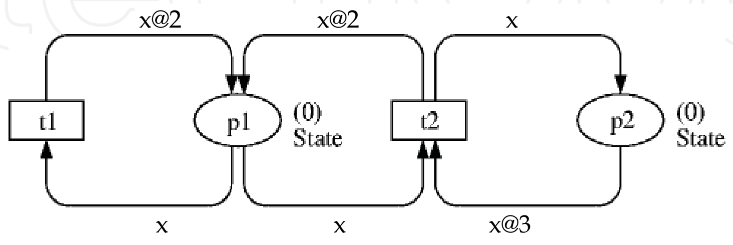


Fig. 2. Example of an unfair RTCP-net

Let's consider the RTCP-net presented in Fig. 2. The set Σ contains only one element `State = int with 0..1` and only one variable x is used. The initial marking $M_0 = (0,0)$ does not change while the net is working. The states change due to the changing of time stamps. The RTCP-net is not fair. The transition *t2* may be starved by the other one. Let's consider a firing sequence where only the transition *t1* is fired. In such a case the time stamp of the place *p2* will be infinitely decreasing. Therefore, the reachability graph for the considered net is infinite. A part of the reachability graph for the RTCP-net is shown in Fig. 3.

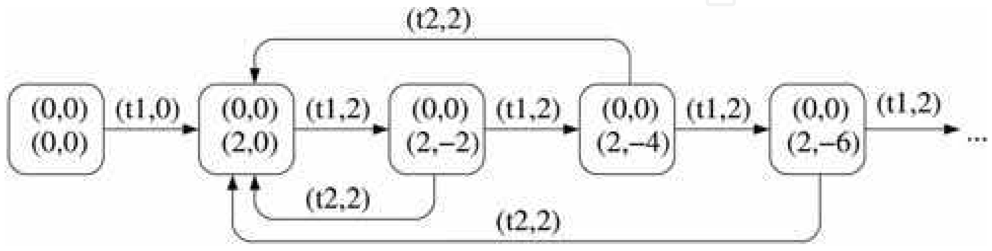


Fig. 3. Part of the reachability graph for the RTCP-net presented in Fig. 2

Let's consider two states of the RTCP-net (M_0, S_1) and (M_0, S_2), where $M_0 = (0,0)$, $S_1 = (2,-4)$ and $S_2 = (2,-6)$. The same transitions are enabled in both states and the same sequences of actions are feasible from the states. Both states have the same markings and the same *level of* tokens accessibility, i.e. we have to wait 2 time-units to take the token from the place *p1* and the token in the place *p*2 is already accessible. The token in the place *p*2 is accessible if its age is at least 3 time-units, i.e. the value of the time stamp is equal to or less than -3. It makes no difference whether the time stamp is equal to -4, -6, etc. The states (M_0, S_1) and (M_0, S_2) will be said to cover each other and only one node in the coverability graph will be used to represent them.

Definition 13. Let $p \in P$ be a place of an RTCP-net $\mathcal{N}$ and let $Out_A(p)$ denote the set of output arcs of the place *p. The maximal accessibility age* of the place *p* is the number:

$$\delta_{max}(p) = \max_{a \in Out_A(p)} \left\{ \max_{b \in B(T(a))} E_S(a)_b \right\}. \tag{8}$$

The maximal accessibility age of a place *p* denotes the age when tokens in the place become accessible for all output transitions of the place.

Definition 14. Let $\mathcal{N}$ be an RTCP-net and let (M_1, S_1) and (M_2, S_2) be states of the net. The state (M_1, S_1) is said to cover the state (M_2, S_2) $((M_1, S_1) \simeq (M_2, S_2))$ iff $M_1 = M_2$ and the following condition holds:

$$\forall p \in P \colon (S_1(p) = S_2(p)) \vee (S_1(p) \leq -\delta_{max}(p) \wedge S_2(p) \leq -\delta_{max}(p)). \tag{9}$$

Proposition 1. The *coverability relation* $\simeq$ is an equivalence relation on $\mathcal{R}(M_0, S_0)$.

Proposition 2. Let (M_1, S_1) and (M_2, S_2) be the states of an RTCP-net $\mathcal{N}$ such that $(M_1, S_1) \simeq (M_2, S_2)$. If $(M_1, S_1) \xrightarrow{(t,b)} (M_1', S_1')$ and $(M_2, S_2) \xrightarrow{(t,b)} (M_2', S_2')$, then $(M_1', S_1') \simeq (M_2', S_2')$.

Proposition 3. Let (M_1, S_1) and (M_2, S_2) be the states of an RTCP-net $\mathcal{N}$ such that $(M_1, S_1) \simeq (M_2, S_2)$. The following equality holds: $\mathcal{L}(M_1, S_1) = \mathcal{L}(M_2, S_2)$.

The reachability and coverability graphs are constructed in a similar way. They differ only about the way a new node is added to the graph. For the coverability graph, after calculating a new node, we check first whether there already exists a node that covers the new one. If so, we add only a new arc that goes to the found state and the new one is omitted. Otherwise, the new state is added to the coverability graph together with the corresponding arc. The coverability graph contains only one node for each equivalence class of the coverability relation.

Let's consider coverability graph for the net presented in Fig. 2. After calculating the state (M_0, S_2) we affirm that there already exists the state (M_0, S_1) that covers it. Therefore, we add only an arc that goes back to the state (M_0, S_1). The coverability graph for the RTCP-net is shown in Fig. 4. The coverability graph for the net presented in Fig. 1 is shown in Fig. 5.

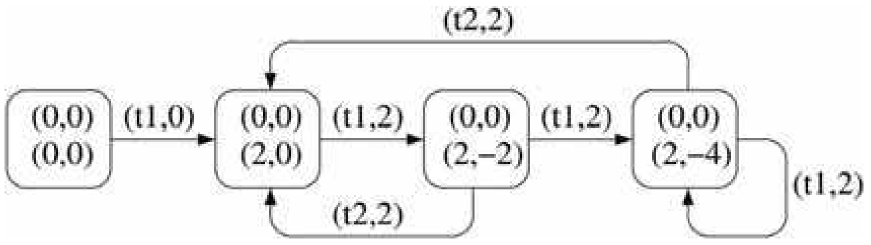


Fig. 4. Coverability graph for the RTCP-net presented in Fig. 2

Proposition 4. If an RTCP-net $\mathcal{N}$ is strongly bounded and each type $\Sigma_i \in \Sigma$ is finite, then the coverability graph is also finite.

Proofs for the presented propositions can be found in (Szpyrka 2006a).

The coverability graph for an RTCP-net provides similar capabilities of analysis of the net properties as the full reachability graph. It contains all reachable markings so it is possible to check the boundedness properties. The coverability graph contains similar arcs' labels as the reachability one (with the same pairs *(t,b)),* therefore, it is also possible to check the liveness properties. Possibilities of analysis of timing properties using coverability graphs are limited insignificantly so some states are not presented directly. To find the minimal and maximal times of the transition from one state to another we use the same algorithms as for reachability graphs. For more details see (Szpyrka 2006a).

4. Practical modelling with RTCP-nets

For the effective modelling RTCP-nets enable to distribute parts of the net across multiple subnets called pages. Hierarchical RTCP-nets are based on hierarchical CP-nets. Substitution transitions and fusion places (Jensen 1992-1997) are used to combine pages but they are a mere designing convenience. The former idea allows the user to refine a transition and its surrounding arcs to a more complex net, which usually gives a more precise and detailed description of the activity represented by the substitution transition. In comparison with CP-nets general ports are not allowed in RTCP-nets. Moreover, each socket node must have only one port node assigned and vice versa. Thus, a hierarchical net can be easily "squash" to a non-hierarchical one.

A fusion of places allows users to specify a set of places that should be considered as a single one. It means, that they all represent a single conceptual place, but are drawn as separate individual places (e.g. for clarity reasons). The places participating in such a fusion set may belong to several different pages. They must have the same types and initial markings. Global fusion sets only are allowed in RTCP-nets.

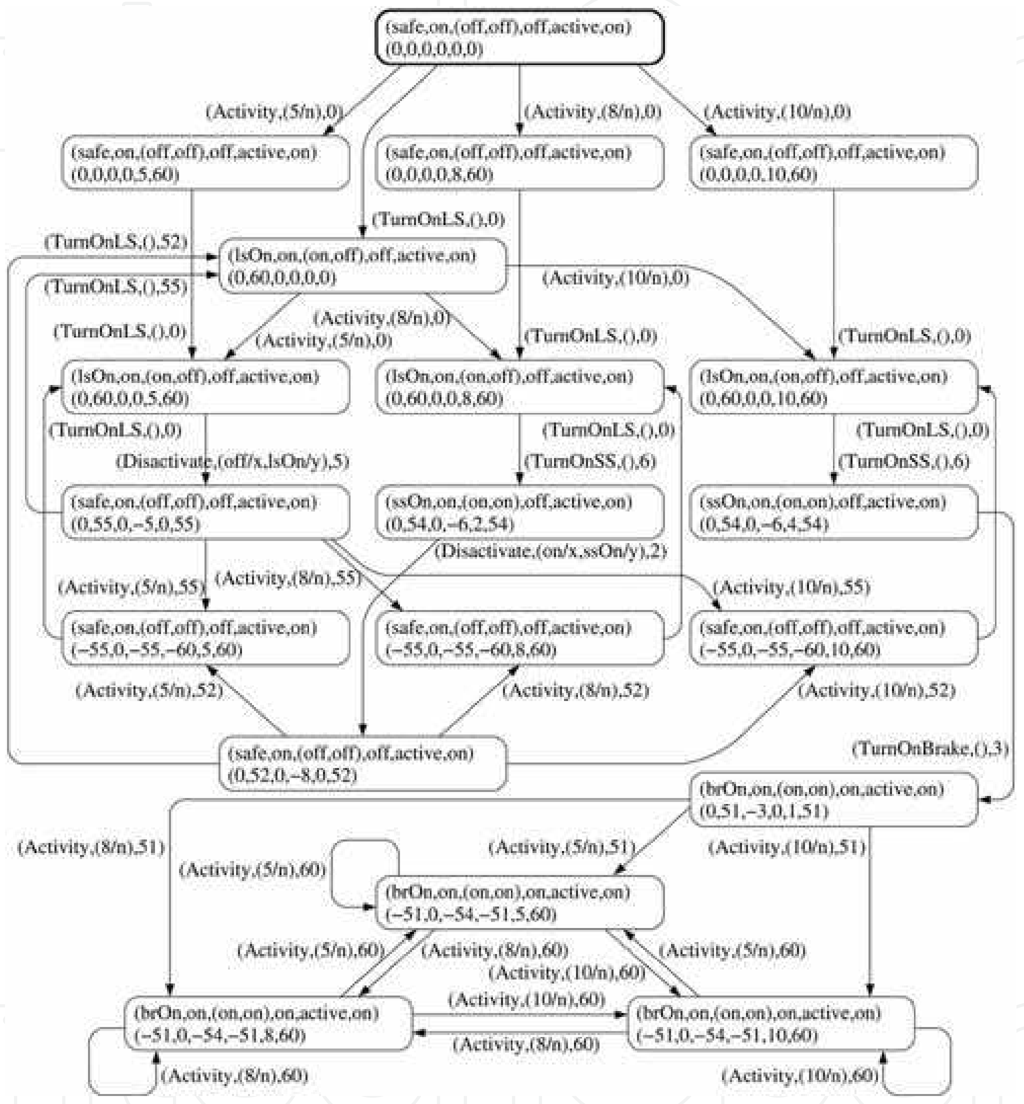


Fig. 5. Coverability graph for the RTCP-net presented in Fig. 1

4.1 Canonical form

A special form of hierarchical RTCP-nets called *canonical form* has been defined to speed up and facilitate drawing of models (Szpyrka and Szmuc 2006c). RTCP-nets in canonical form consist of four types of subnets with precisely defined structures: primary place pages, primary transition pages, linking pages, and D-nets. Such a model describes the structure of the corresponding system as well as its behaviour and functional aspects. Furthermore, rule-based systems can be simply included into such models. The general structure of an RTCP-net in canonical form is shown in Fig. 6.

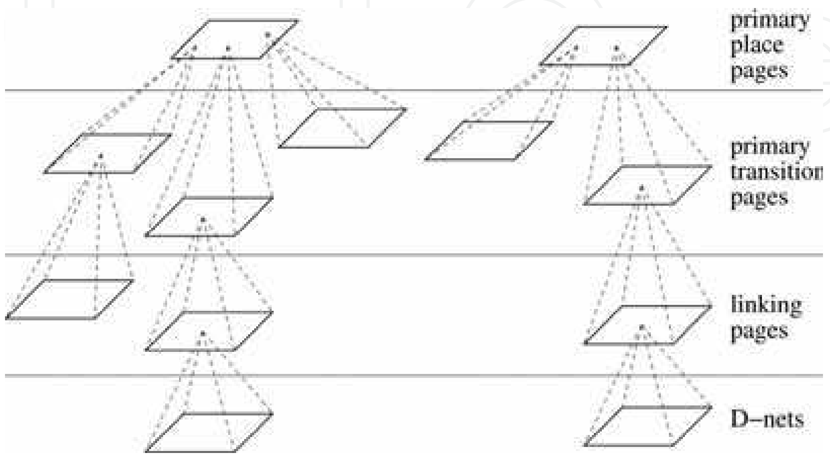


Fig. 6. General structure of an RTCP-net in canonical form

Moreover, it is assumed that an RTCP-net in canonical form satisfies some extra conditions. The set of places P is divided into two subsets: P_M, the set of *main places* and P_A, the set of *auxiliary places.* Main places represent the distinguished parts (elements) of a modelled system, e.g. objects. The set T of all transitions is also divided into two subsets: T_M *(main transitions)* and T_A *(auxiliary transitions).* Main transitions represent actions of a modelled system. Auxiliary places and transitions are used on subpages, which describe system activities in detail. Main places may be connected to main transitions only. Initial time stamps of auxiliary places must be equal to or less than 0. Moreover, if an arc goes from or to an auxiliary place, its time expression must be equal to 0.

Primary place pages are used to represent active objects (i.e. objects performing activities) and their activities. They are oriented towards objects presentation and are top level pages. Such a page is composed of one main place that represents the object and one main transition for each object activity. *Primary transition pages* are oriented towards activities' presentation and are second level pages. Such a page contains all the places, the values of which are necessary to execute the activity, i.e. the page is composed of one main transition that represents the activity and a few main places.

Linking pages belong to the functional level of a model. They are used (if necessary) to represent an algorithm that describes an activity in details. Moreover, a linking page is used as an interface for gluing the corresponding D-net into a model. Such a page is used to gather all necessary information for the D-net and to distribute the results of the D-net activity. A linking page contains port nodes for socket nodes from the corresponding primary transition page. The substitution transition (from the corresponding primary transition page) is split into two main transitions an input and an output one. All elements placed between those transitions are auxiliary ones, so there is no delay between firing of the input and output transitions. Hence, if time properties are considered, we can focus on primary transition pages and pass over their subpages. Any activity of a linking page starts with the firing of the input transition and ends with the firing of the output one. In addition, each occurrence of the input

transition must be followed by a sequence of transitions' occurrences such that the last of them is the output transition, and all the others are auxiliary ones. Any such activity is similar to a procedure call in programming languages.
D-nets (Szpyrka & Szmuc 2006a) are used to represent rule-based systems in a Petri net form. They are utilized to verify a rule-based system properties and constitute parts of an RTCP-net model. A D-net contains two places: a *conditional* and a *decision place.* Each decision rule is represented by a transition and its input and output arcs. A token placed in the conditional place denotes a sequence of values of conditional attributes. Similarly, a token placed in the decision place denotes a sequence of values of decision attributes. D-nets belong to the bottom level of the model. All its nodes belong to auxiliary ones. A simplified structure of these four types of pages is shown in Fig. 7.

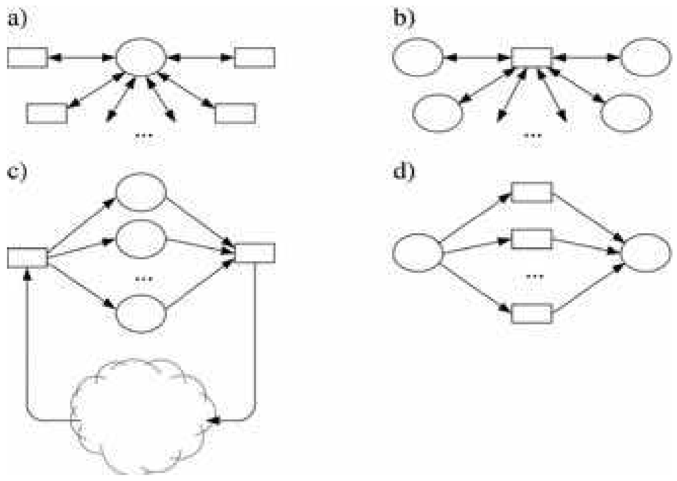


Fig. 7. Simplified structure of RTCP-net pages: a) primary place page; b) primary transition page; c) linking page; d) D-net

All connections among pages are presented using a page hierarchy graph. A node in such a graph represents a single page, and an arc represents a connection between a subpage and its substitution transition.
System decomposition is the first step of a model development. It starts with distinguishing objects that constitute the system. Objects are divided into *active,* i.e., objects performing activities, and *passive ones,* that do not perform any individual activity. An object is represented by a main place. For each object, a list of attributes and their types are defined. The Cartesian product of the defined types specifies the corresponding place type. Construction of primary place pages for active objects ends this development stage.
The next stage deals with description of model dynamic that is especially important for reactive systems. Transitions placed in primary place pages are usually substitution transitions. For each of these substitution transitions a primary transition page is drawn. Designing of a primary transition page is similar to declaring a procedure in Ada programming language. It is necessary to describe input, output and input/output parameters. If a primary transition page does not contain a substitution transition, then it constitutes a complete definition of the corresponding activity. After completion of this stage, RTCP-net represents all elements (objects) that constitute the modelled system and all

its activities.
The last stage is related to development of functional aspects of the system. Linking pages and D-nets (if necessary) are used for this purpose.

4.2 Railway traffic management system – case study

RTCP-nets can be used as modelling language for real embedded systems. A model of railway traffic management system for a real train station is discussed in this subsection. The system is used to ensure safe riding of trains through the station. It collects some information about current railway traffic and uses a rule-based system to choose routes for trains. The presented approach based on RTCP-nets seems to be valuable and worth consideration as an alternative for other approaches such as SDL language (Bacherini et al. 2003), statecharts (Banci et al. 2004) and others.
The size of a train station has a great influence on the size of the corresponding RTCPnet model. To give a brief outline of the presented approach a small train station (Czarna Tarnowska) has been chosen. The station belongs to the Polish railway line no 91 from Kraków to Medyka. This example seems to be suitable for RTCP-nets presentation.

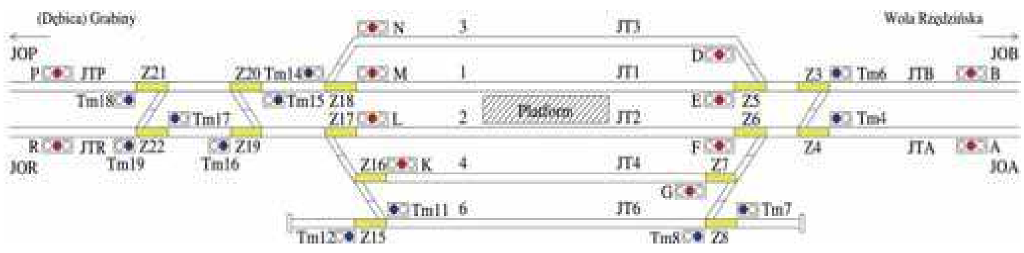


Fig. 8. Czarna Tarnowska – topology of the train station

The topology of the train station with original signs is shown in Fig. 8. The letters A, B, D, etc. stand for color light signals, the symbols Z3, Z4, Z5, etc. stand for turnouts and JTA, JTB, JT1, etc. stand for track segments. Some simplification have been introduced to reduced the size of the model. We are not interested in controlling local shunts so the track segment JT6 will not be considered. We assume that light signals display only two signals: *stop, way free*. Moreover, outside the station the trains can ride using the right track only.
A train can ride through the station only if a suitable route has been prepared for it i.e., suitable track segments must be free, we have to set turnouts and light signals and to guarantee exclusive rights to these elements for the train. Required position of turnouts for all possible routes are shown in Tab. 1. For example, the symbol B4 stands for the input route from the light signal B to the track no. 4. The symbol F2W stands for the output route from the track no. 2 (from the light signal F) to the right (to Wola Rzedzinska), etc. The route B4 can be used by a train only if: turnouts 7, 8, 15, 16 are closed, turnouts 3, 4, 6 are open, and the track segments JTB, JT4, JZ4/6 (a segment between turnouts 4 and 6), JZ7 (diagonal segment leading to the turnout 7) and JZ16 are free. The Tab. 2 shows which routes are mutually exclusive. The system is expected to choose suitable routes for moving trains. It should take under consideration that some trains should stop at the platform, while others are only moving through the station and two routes (an input and an output one) should be

prepared for them. In such a case, if it is not possible to prepare two routes, only an input one can be prepared.

	Turnouts								
Routes	3/4	5	6	7/8	15/16	17	18	19/20	21/22
B1	+	+							
B2	–		+	o+					
B3	+	–							
B4	–		–	+	o+				
R2				o+	o+	+		+	+
R4				o+	+	–		+	+
F2W	+		+	o+					
G2W	+		–	+					
K1D					+	–		–	+
L1D					o+	+		–	+
M1D							+	+	+
N1D							–	+	+

+	closed turnout (the straight route);
–	open turnout (the diverging route);
o+	closed turnout (from safety reasons).

Table 1. Required position of turnouts for all possible routes

	B1	B2	B3	B4	R2	R4	F2W	G2W	K1D	L1D	M1D	N1D
B1	–	x	x	x								xx
B2	x	–	x	x	xx		x	x				
B3	x	x	–	x								
B4	x	x	x	–	xx	xx	x	x				
R2		xx		xx	–	x		xx	x	x		
R4				xx	x	–			x	x		
F2W		x		x			–	x				
G2W		x		x	xx		x	–				
K1D					x	x			–	x	x	x
L1D					x	x			x	–	x	x
M1D									x	x	–	x
N1D	xx								x	x	x	–

x	mutually exclusive (different position of turnouts);
xx	mutually exclusive (safety reasons).

Table 2. Relationships between routs

The main part of the developed system is a rule-based system that is used to determine which routes should be prepared depending on the data collected from sensors. In the considered approach generalized decision tables (tables with non-atomic values of attributes, (Szpyrka & Szmuc 2006a)) are used to represent rule-based systems. A cell in such a decision table contains a formula that evaluates to a boolean value for conditional attributes, and to a single value (that belongs to the corresponding domain) for decision attributes. After verification such a decision table is transformed into a Petri nets form called D-net (Szpyrka & Szmuc 2006a).

The decision table for the considered model contains 20 conditional and 2 decision attributes. The conditional attributes stand for: information about current position of the train (attribute JT) - before the light signal B, F, G, etc.; information about type of the train (attribute TT) - only moves through the station (1) or must stop at the platform (0); information about current status of track segments (attributes JT1, JT2, JT3, JT4, JOA, JOP) - a segment is free (0) or is taken (1); information about already prepared routes (attributes B1, B2, B3, etc.) - a route is already set (1) or not (0). The decision attributes In and Out represent the input and output routes (that will be prepared for the train) respectively. Domains for these attributes are defined as follows:

$$D_{\mathrm{JT}} = \textbf{with}\ \mathrm{b} \mid \mathrm{f} \mid \mathrm{g} \mid \mathrm{k} \mid \mathrm{l} \mid \mathrm{m} \mid \mathrm{n} \mid \mathrm{r},$$
$$D_{\mathrm{TT}} = \textbf{int with}\ 1..2,$$
$$D_{\mathrm{JT1}} = D_{\mathrm{JT2}} = D_{\mathrm{JT3}} = D_{\mathrm{JT4}} = D_{\mathrm{JOA}} = D_{\mathrm{JOP}} = \textbf{int with}\ 0..2,$$
$$D_{\mathrm{B1}} = D_{\mathrm{B2}} = \ldots = D_{\mathrm{N1D}} = \textbf{int with}\ 0..1,$$
$$D_{\mathrm{In}} = \textbf{with}\ \mathrm{b1} \mid \mathrm{b2} \mid \mathrm{b3} \mid \mathrm{b4} \mid \mathrm{r2} \mid \mathrm{r4} \mid \mathrm{none},$$
$$D_{\mathrm{Out}} = \textbf{with}\ \mathrm{f2w} \mid \mathrm{g2w} \mid \mathrm{k1d} \mid \mathrm{l1d} \mid \mathrm{m1d} \mid \mathrm{n1d} \mid \mathrm{none}.$$

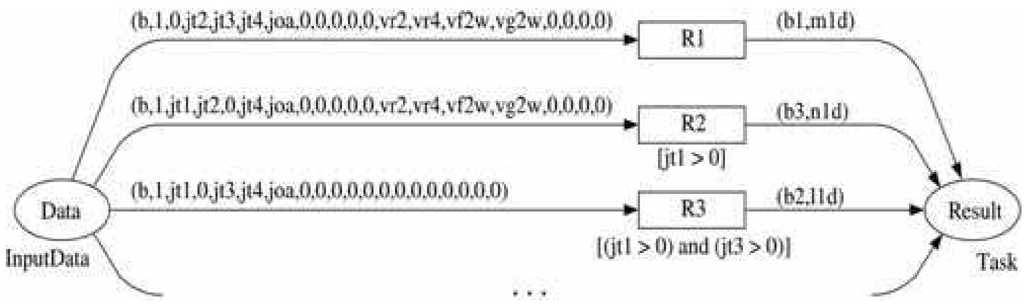

(b,1,0,jt2,jt3,jt4,joa,0,0,0,0,0,vr2,vr4,vf2w,vg2w,0,0,0,0)
R1
(b1,m1d)
(b,1,jt1,jt2,0,jt4,joa,0,0,0,0,0,vr2,vr4,vf2w,vg2w,0,0,0,0)
R2
[jt1 > 0]
(b3,n1d)
(b,1,jt1,0,jt3,jt4,joa,0,0,0,0,0,0,0,0,0,0,0,0,0)
R3
[(jt1 > 0) and (jt3 > 0)]
(b2,l1d)
Data
InputData
Result
Task
. . .

Fig. 9. Part of the D-net for the decision table presented in Table 3 and 4

The decision table is shown in Tab. 3 and 4. Moreover, the table contains 81 so-called negative rules (Szpyrka & Szmuc 2006a) that state in an explicit way that the particular combinations of values of conditional attributes are impossible or not allowed. The negative rules are used to check whether the table is complete and are usually omitted when the corresponding D-net is generated. Thus they will not be considered in the paper. Together with the negative rules, the rule-based system is complete and deterministic. A small part of the D-net for the decision table (only three rules) is shown in Fig. 9. Names beginning with "v" (e.g. vf2w) denote variables.

In the considered model two active objects are distinguished: *Driver* used to handle with trains requests and *SignalBox* used to set and unset routes. The first object contains information about all requests waiting for service, while the second one about states of all routes. Definitions of main types and variables used in the model are as follows:

```
color State = int with 0..1;
color TrainType = int with 0..2;
```

	JT	TT	JT1	JT2	JT3	JT4	JOA	JOP	B1	B2	B3
R1	JT = b	TT = 1	JT1 = 0	JT2	JT3	JT4	JOA	JOP = 0	B1 = 0	B2 = 0	B3 = 0
R2	JT = b	TT = 1	JT1 > 0	JT2	JT3 = 0	JT4	JOA	JOP = 0	B1 = 0	B2 = 0	B3 = 0
R3	JT = b	TT = 1	JT1 > 0	JT2 = 0	JT3 > 0	JT4	JOA	JOP = 0	B1 = 0	B2 = 0	B3 = 0
R4	JT = b	TT = 1	JT1 > 0	JT2 > 0	JT3 > 0	JT4 = 0	JOA	JOP = 0	B1 = 0	B2 = 0	B3 = 0
R5	JT = r	TT = 1	JT1	JT2 = 0	JT3	JT4	JOA = 0	JOP	B1	B2 = 0	B3
R6	JT = r	TT = 1	JT1	JT2 > 0	JT3	JT4 = 0	JOA = 0	JOP	B1	B2 = 0	B3
R7	JT = b	TT = 2	JT1 = 0	JT2	JT3	JT4	JOA	JOP	B1 = 0	B2 = 0	B3 = 0
R8	JT = b	TT = 2	JT1 > 0	JT2 = 0	JT3	JT4	JOA	JOP	B1 = 0	B2 = 0	B3 = 0
R9	JT = b	TT = 2	JT1	JT2 = 0	JT3	JT4	JOA	JOP	B1 = 0	B2 = 0	B3 = 0
R10	JT = r	TT = 2	JT1	JT2 = 0	JT3	JT4	JOA	JOP	B1	B2 = 0	B3
R11	JT = b	TT = 1	JT1 = 0	JT2	JT3	JT4	JOA	JOP > 0	B1 = 0	B2 = 0	B3 = 0
R12	JT = b	TT = 1	JT1 = 0	JT2	JT3	JT4	JOA	JOP	B1 = 0	B2 = 0	B3 = 0
R13	JT = b	TT = 1	JT1 = 0	JT2	JT3	JT4	JOA	JOP	B1 = 0	B2 = 0	B3 = 0
R14	JT = b	TT = 1	JT1 = 0	JT2	JT3	JT4	JOA	JOP	B1 = 0	B2 = 0	B3 = 0
R15	JT = b	TT = 1	JT1 > 0	JT2 = 0	JT3 > 0	JT4	JOA	JOP > 0	B1 = 0	B2 = 0	B3 = 0
R16	JT = b	TT = 1	JT1 > 0	JT2 = 0	JT3 > 0	JT4	JOA	JOP	B1 = 0	B2 = 0	B3 = 0
R17	JT = b	TT = 1	JT1 > 0	JT2 = 0	JT3 > 0	JT4	JOA	JOP	B1 = 0	B2 = 0	B3 = 0
R18	JT = b	TT = 1	JT1 > 0	JT2 = 0	JT3 > 0	JT4	JOA	JOP	B1 = 0	B2 = 0	B3 = 0
R19	JT = b	TT = 1	JT1 > 0	JT2 = 0	JT3 > 0	JT4	JOA	JOP	B1 = 0	B2 = 0	B3 = 0
R20	JT = b	TT = 1	JT1	JT2 = 0	JT3 > 0	JT4	JOA	JOP	B1 = 0	B2 = 0	B3 = 0
R21	JT = b	TT = 1	JT1 > 0	JT2	JT3 = 0	JT4	JOA	JOP > 0	B1 = 0	B2 = 0	B3 = 0
R22	JT = b	TT = 1	JT1 > 0	JT2	JT3 = 0	JT4	JOA	JOP	B1 = 0	B2 = 0	B3 = 0
R23	JT = b	TT = 1	JT1 > 0	JT2	JT3 = 0	JT4	JOA	JOP	B1 = 0	B2 = 0	B3 = 0
R24	JT = b	TT = 1	JT1 > 0	JT2	JT3 = 0	JT4	JOA	JOP	B1 = 0	B2 = 0	B3 = 0
R25	JT = b	TT = 1	JT1	JT2	JT3 = 0	JT4	JOA	JOP	B1 = 0	B2 = 0	B3 = 0
R26	JT = b	TT = 1	JT1 > 0	JT2 > 0	JT3 > 0	JT4 = 0	JOA	JOP > 0	B1 = 0	B2 = 0	B3 = 0
R27	JT = b	TT = 1	JT1 > 0	JT2 > 0	JT3 > 0	JT4 = 0	JOA	JOP	B1 = 0	B2 = 0	B3 = 0
R28	JT = b	TT = 1	JT1 > 0	JT2 > 0	JT3 > 0	JT4 = 0	JOA	JOP	B1 = 0	B2 = 0	B3 = 0
R29	JT = b	TT = 1	JT1 > 0	JT2 > 0	JT3 > 0	JT4 = 0	JOA	JOP	B1 = 0	B2 = 0	B3 = 0
R30	JT = b	TT = 1	JT1	JT2 > 0	JT3 > 0	JT4 = 0	JOA	JOP	B1 = 0	B2 = 0	B3 = 0
R31	JT = r	TT = 1	JT1	JT2 = 0	JT3	JT4	JOA > 0	JOP	B1	B2 = 0	B3
R32	JT = r	TT = 1	JT1	JT2 = 0	JT3	JT4	JOA	JOP	B1	B2 = 0	B3
R33	JT = r	TT = 1	JT1	JT2 > 0	JT3	JT4 = 0	JOA > 0	JOP	B1	B2	B3
R34	JT = r	TT = 1	JT1	JT2	JT3	JT4 = 0	JOA	JOP	B1 = 0	B2 = 1	B3 = 0
R35	JT = r	TT = 1	JT1	JT2 > 0	JT3	JT4 = 0	JOA	JOP	B1	B2	B3
R36	JT = r	TT = 1	JT1	JT2	JT3	JT4 = 0	JOA	JOP	B1	B2	B3
R37	JT = k	TT = 1	JT1	JT2	JT3	JT4 > 0	JOA	JOP = 0	B1	B2	B3
R38	JT = l	TT	JT1	JT2 > 0	JT3	JT4	JOA	JOP = 0	B1	B2	B3
R39	JT = m	TT	JT1 > 0	JT2	JT3	JT4	JOA	JOP = 0	B1	B2	B3
R40	JT = n	TT = 1	JT1	JT2	JT3 > 0	JT4	JOA	JOP = 0	B1 = 0	B2	B3
R41	JT = f	TT	JT1	JT2 > 0	JT3	JT4	JOA = 0	JOP	B1	B2 = 0	B3
R42	JT = g	TT = 1	JT1	JT2	JT3	JT4 > 0	JOA = 0	JOP	B1	B2 = 0	B3

Table 3. Decision table (part 1)

```
color RoutesStates = product State * State *... (State^12);
color Route = with b1 | b2 | b3 | ... | n1d | none;
color Task = product Route * Route;
color LightSignal = with a | b | d | e | f | ...;
color Request = product LightSignal * TrainType;
var x1, x2, x3,..., y1, y2, y3,... : State;
var z1, z2 : Route;
var s1 : LightSignal;
var p1, p2, p3, p4, p5, p6, p7 : TrainType;
```

B4	R2	R4	F2W	G2W	K1D	L1D	M1D	N1D	In	Out
B4 = 0	R2	R4	F2W	G2W	K1D = 0	L1D = 0	M1D = 0	N1D = 0	b1	m1d
B4 = 0	R2	R4	F2W	G2W	K1D = 0	L1D = 0	M1D = 0	N1D = 0	b3	n1d
B4 = 0	R2 = 0	R4 = 0	F2W = 0	G2W = 0	K1D = 0	L1D = 0	M1D = 0	N1D = 0	b2	l1d
B4 = 0	R2 = 0	R4 = 0	F2W = 0	G2W = 0	K1D = 0	L1D = 0	M1D = 0	N1D = 0	b4	k1d
B4 = 0	R2 = 0	R4 = 0	F2W = 0	G2W = 0	K1D = 0	L1D = 0	M1D	N1D	r2	f2w
B4 = 0	R2 = 0	R4 = 0	F2W = 0	G2W = 0	K1D = 0	L1D = 0	M1D	N1D	r4	g2w
B4 = 0	R2	R4	F2W	G2W	K1D	L1D	M1D	N1D = 0	b1	none
B4 = 0	R2 = 0	R4	F2W = 0	G2W = 0	K1D	L1D	M1D	N1D	b2	none
B4 = 0	R2 = 0	R4	F2W = 0	G2W = 0	K1D = 0	L1D = 0	M1D = 0	N1D = 1	b2	none
B4 = 0	R2 = 0	R4 = 0	F2W	G2W = 0	K1D = 0	L1D = 0	M1D	N1D	r2	none
B4 = 0	R2	R4	F2W	G2W	K1D	L1D	M1D	N1D = 0	b1	none
B4 = 0	R2	R4	F2W	G2W	K1D = 1	L1D = 0	M1D = 0	N1D = 0	b1	none
B4 = 0	R2	R4	F2W	G2W	K1D = 0	L1D = 1	M1D = 0	N1D = 0	b1	none
B4 = 0	R2	R4	F2W	G2W	K1D = 0	L1D = 0	M1D = 1	N1D = 0	b1	none
B4 = 0	R2 = 0	R4	F2W = 0	G2W = 0	K1D	L1D	M1D	N1D	b2	none
B4 = 0	R2 = 0	R4 = 1	F2W = 0	G2W = 0	K1D = 0	L1D = 0	M1D	N1D	b2	none
B4 = 0	R2 = 0	R4	F2W = 0	G2W = 0	K1D = 1	L1D = 0	M1D = 0	N1D = 0	b2	none
B4 = 0	R2 = 0	R4	F2W = 0	G2W = 0	K1D = 0	L1D = 1	M1D = 0	N1D = 0	b2	none
B4 = 0	R2 = 0	R4	F2W = 0	G2W = 0	K1D = 0	L1D = 0	M1D = 1	N1D = 0	b2	none
B4 = 0	R2 = 0	R4	F2W = 0	G2W = 0	K1D = 0	L1D = 0	M1D = 0	N1D = 1	b2	none
B4 = 0	R2	R4	F2W	G2W	K1D	L1D	M1D	N1D	b3	none
B4 = 0	R2	R4	F2W	G2W	K1D = 1	L1D = 0	M1D = 0	N1D = 0	b3	none
B4 = 0	R2	R4	F2W	G2W	K1D = 0	L1D = 1	M1D = 0	N1D = 0	b3	none
B4 = 0	R2	R4	F2W	G2W	K1D = 0	L1D = 0	M1D = 1	N1D = 0	b3	none
B4 = 0	R2	R4	F2W	G2W	K1D = 0	L1D = 0	M1D = 0	N1D = 1	b3	none
B4 = 0	R2 = 0	R4 = 0	F2W = 0	G2W = 0	K1D	L1D	M1D	N1D	b4	none
B4 = 0	R2 = 0	R4 = 0	F2W = 0	G2W = 0	K1D = 1	L1D = 0	M1D = 0	N1D = 0	b4	none
B4 = 0	R2 = 0	R4 = 0	F2W = 0	G2W = 0	K1D = 0	L1D = 1	M1D = 0	N1D = 0	b4	none
B4 = 0	R2 = 0	R4 = 0	F2W = 0	G2W = 0	K1D = 0	L1D = 0	M1D = 1	N1D = 0	b4	none
B4 = 0	R2 = 0	R4 = 0	F2W = 0	G2W = 0	K1D = 0	L1D = 0	M1D = 0	N1D = 1	b4	none
B4 = 0	R2 = 0	R4 = 0	F2W	G2W = 0	K1D = 0	L1D = 0	M1D	N1D	r2	none
B4 = 0	R2 = 0	R4 = 0	F2W = 1	G2W = 0	K1D = 0	L1D = 0	M1D	N1D	r2	none
B4 = 0	R2 = 0	R4 = 0	F2W	G2W	K1D = 0	L1D = 0	M1D	N1D	r4	none
B4 = 0	R2 = 0	R4 = 0	F2W	G2W	K1D = 0	L1D = 0	M1D	N1D	r4	none
B4 = 0	R2 = 0	R4 = 0	F2W = 1	G2W = 0	K1D = 0	L1D = 0	M1D	N1D	r4	none
B4 = 0	R2 = 0	R4 = 0	F2W = 0	G2W = 1	K1D = 0	L1D = 0	M1D	N1D	r4	none
B4	R2 = 0	R4 = 0	F2W	G2W	K1D = 0	L1D = 0	M1D = 0	N1D = 0	none	k1d
B4	R2 = 0	R4 = 0	F2W	G2W	K1D = 0	L1D = 0	M1D = 0	N1D = 0	none	l1d
B4	R2	R4	F2W	G2W	K1D = 0	L1D = 0	M1D = 0	N1D = 0	none	m1d
B4	R2	R4	F2W	G2W	K1D = 0	L1D = 0	M1D = 0	N1D = 0	none	n1d
B4 = 0	R2	R4	F2W = 0	G2W = 0	K1D	L1D	M1D	N1D	none	f2w
B4 = 0	R2 = 0	R4	F2W = 0	G2W = 0	K1D	L1D	M1D	N1D	none	g2w

Table 4. Decision table (part 2)

As it was said before, primary place pages are used to represent active objects. Such pages for objects *Driver* and *SignalBox* are shown in Fig. 10 and 11 respectively.

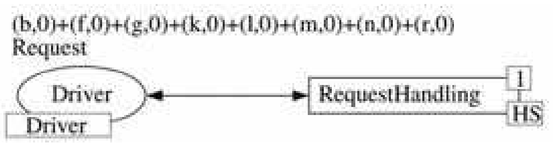


Fig. 10. Primary place page *Driver*

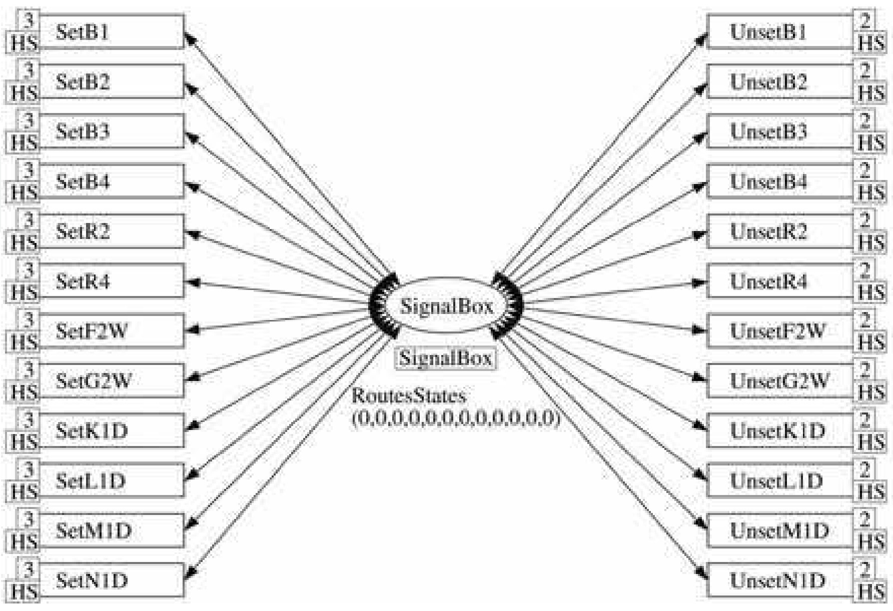


Fig. 11. Primary place page *SignalBox*

The RTCP-net model contains some passive objects such as light signals, turnouts and track segments. For simplicity turnouts that work together e.g. Z34 and Z78 are considered as single objects. Places that represent light signals and turnouts have the colour State assigned. The value 0 denotes that the stop signal is displayed or that the turnout is closed. Places that represent track segments have the colour TrainType assigned. A positive value denotes that the track segment is taken while the value 2 denotes that the train that occupies it must stop at the platform.

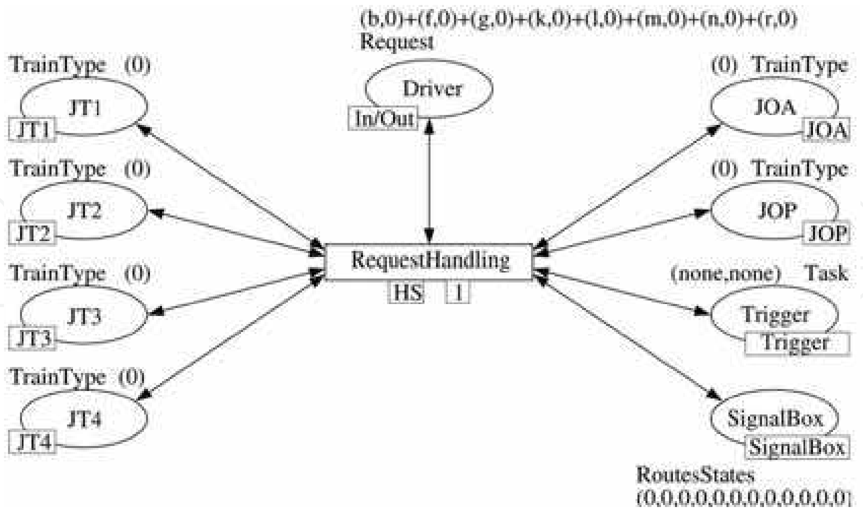
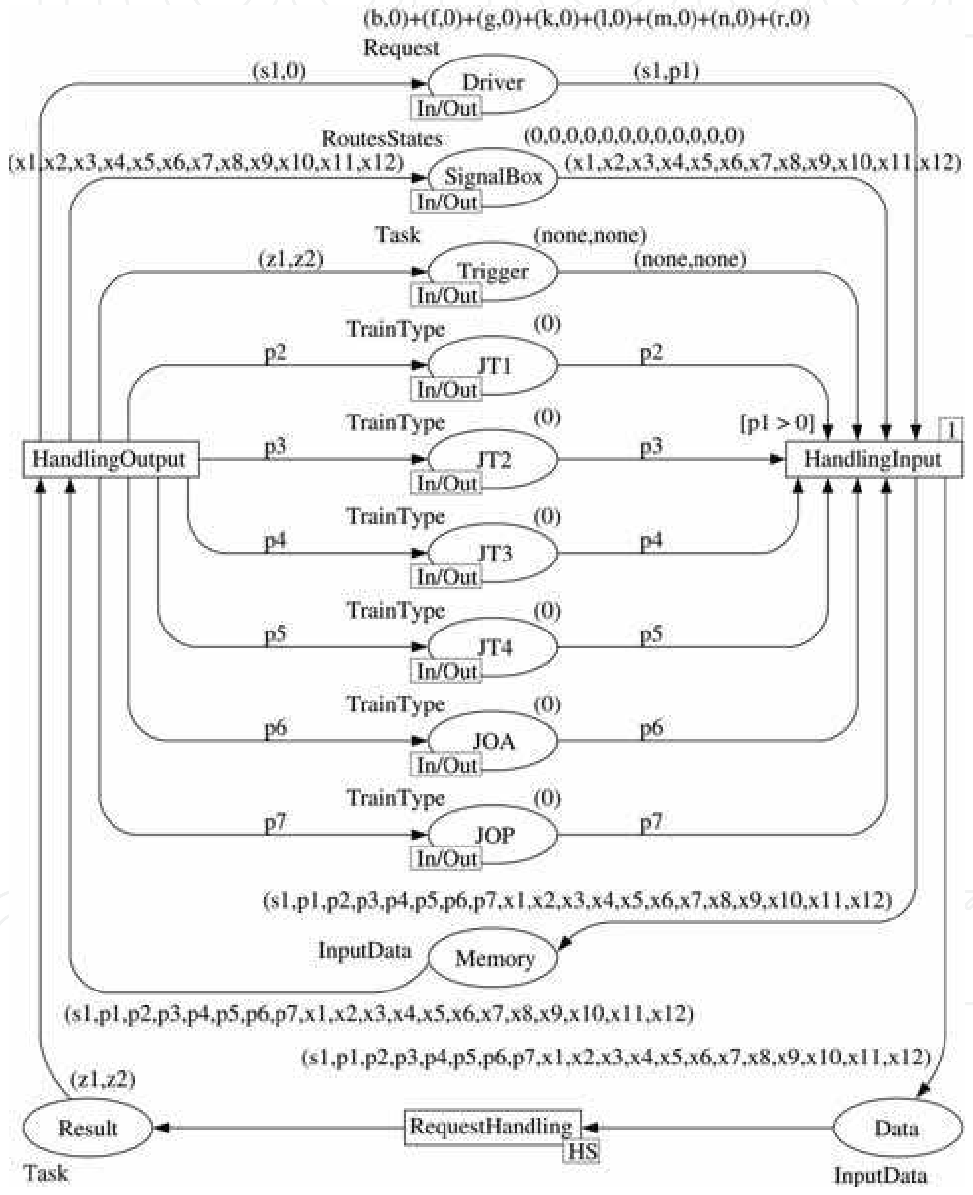


Fig. 12. Primary transition page *RequestHandling*

Transitions placed in primary place pages are usually substitution transitions. For each of them a *primary transition page* is drawn. Primary transition page for the *RequestHandling* is shown in Fig. 12. The transition performs on basis of the considered D-net. To connect the D-net with the substitution transition *RequestHandling* from the Fig. 12 a linking page must be used. The linking page used to gather all necessary information for the D-net and to distribute the results of the D-net activity is shown in Fig. 13.


(b,0)+(f,0)+(g,0)+(k,0)+(l,0)+(m,0)+(n,0)+(r,0)
Request
(s1,0)
Driver
(s1,p1)
In/Out
RoutesStates
(0,0,0,0,0,0,0,0,0,0,0,0)
(x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12)
SignalBox
(x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12)
Task
(none,none)
(z1,z2)
Trigger
(none,none)
TrainType
(0)
p2
JT1
p2
[p1 > 0]
1
HandlingOutput
p3
JT2
p3
HandlingInput
p4
JT3
p4
p5
JT4
p5
p6
JOA
p6
p7
JOP
p7
(s1,p1,p2,p3,p4,p5,p6,p7,x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12)
InputData
Memory
(s1,p1,p2,p3,p4,p5,p6,p7,x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12)
(s1,p1,p2,p3,p4,p5,p6,p7,x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12)
(z1,z2)
Result
Task
RequestHandling
HS
Data
InputData
Fig. 13. Linking page for the transition

Primary transition pages for transitions *SetB1* and *UnsetK1D* are shown in Fig. 14 and 15 respectively. A route is unset if the second of the light signals (set for the route) displays the stop signal (light signals switch to stop signal when trains move) and the corresponding track segment is free. As the result of unset operation all used turnouts are switch to the closed position. Pages for other *Set* and *Unset* transitions are designed in similar way.

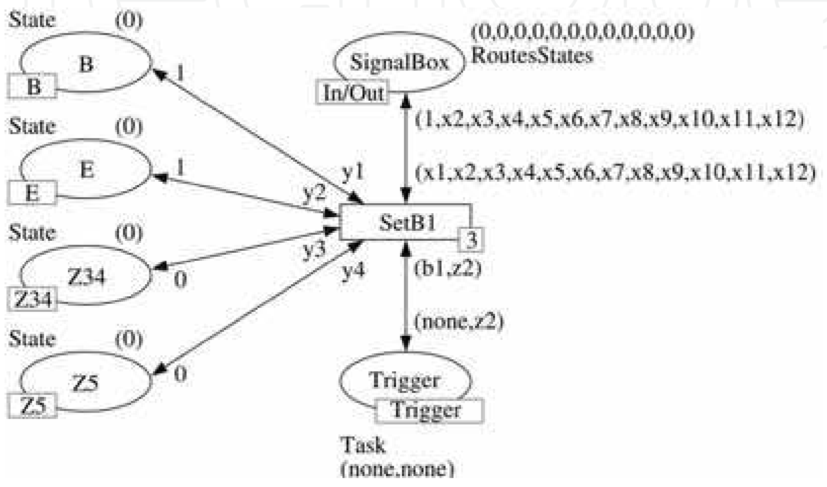


Fig. 14. Primary transition page *SetB1*

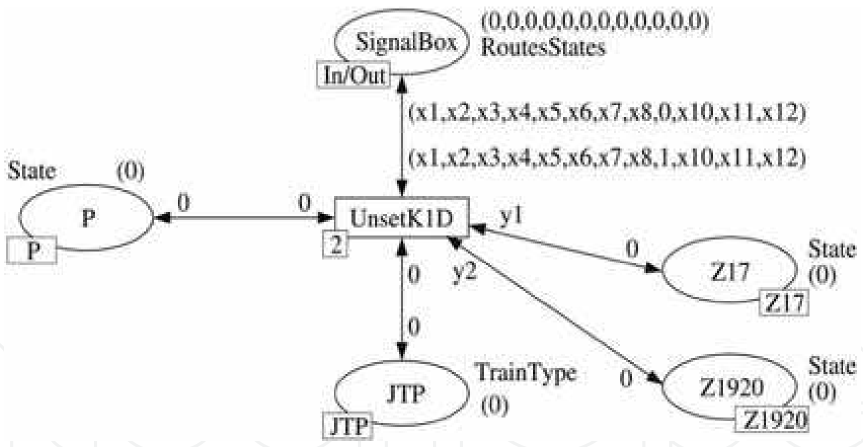


Fig. 15. Primary transition page *UnsetK1D*

A part of the page hierarchy graph for the RTCP-net is shown in Fig. 16. The complete model contains also pages that represent trains and are use to simulate the train traffic.

Verification of such a model is carried out in two ways. In the first one, simulation is used to check how the model works. Simulation of an RTCP-net model is similar to a program debugging. It can be used to check whether the model performs as expected or for some statistical analysis of its properties. A small part of a simulation report for the considered model is presented below. Each part of the report contains the following pieces of information: a state number, markings of places, time stamps of places (in square brackets), a transition name, a binding of the transition and a passage of time (before the transition can fire). For example in the state 13 there is possible a passage of time equal to 20 time-units and then the transition *FreeFA1* (connected with a train moving) in the binding *b = (1/p1)* is fired.

```
13:
Driver   : [0] (b,0)+(f,0)+(g,0)+(k,0)+(l,0)+(m,0)+(n,0)+(r,2)
SignalBox: [0] (0,0,0,0,0,0,1,0,0,0,0,0)
Trigger  : [0] (none,none)
A, B, D, E, F, G, K, L, M, N, P, R
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
[0, -80, -80, -80, 0, -80, -80, -60, -80, -80, -80, -80]

JOA, JOB, JOP, JOR, JTA, JTB, JTP, JTR, JT1, JT2, JT3, JT4
1,   0,   0,   2,   1,   0,   0,   0,   0,   1,   0,   0
[0, -20, 0, -20, 0, -80, -80, -60, 0, 0, 0, 0]
Z34, Z5, Z6, Z78, Z1516, Z17, Z18, Z1920, Z2122
0,   0,  0,  0,   0,     0,   0,   0,     0,
[0, -80, 0, 0, -80, -80, -80, -80, -80]
Trains: DW1,   DW2,   DW3,   WD1,   WD2,   WD3,   WD4,   WD5
        (w,f), (r,x), (x,x), (x,x), (x,x), (x,x), (x,x), (x,x)
[20, -20, -80, -20, -80, -80, -80, -80]
TimerDW, TimerWD
(0)+(1), (0)+(1)
[40, 40]
------------------------------------------------------------------
--((FreeFA1.(1/p1)), 20)--> 14
```

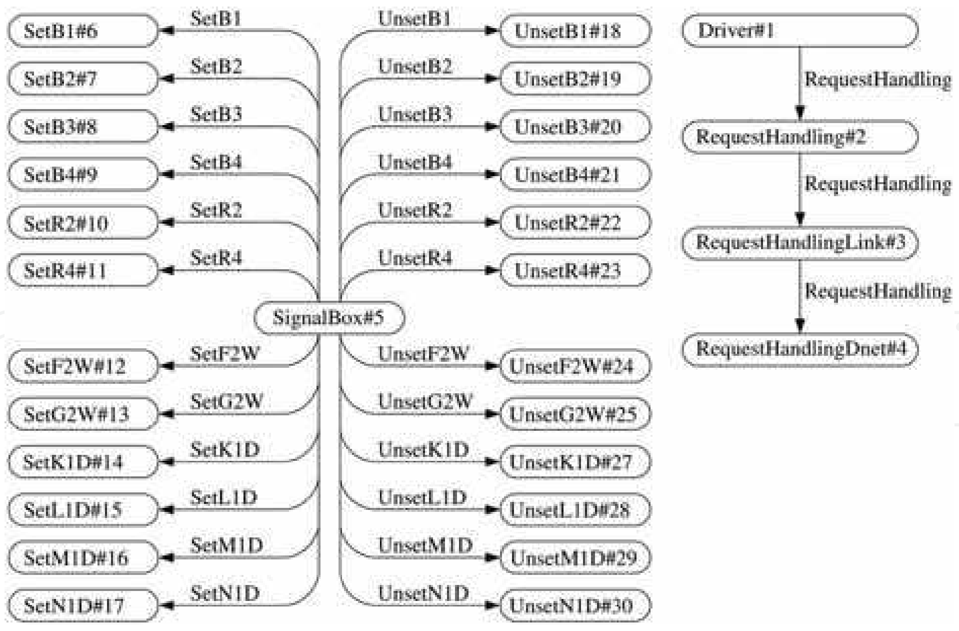


Fig. 16. Part of the page hierarchy graph

The real formal verification is based on coverability graphs. The textual form of the coverability graph is similar to the one of simulation report. In comparison with the simulation report, the coverability graph represents not one but all possible paths of the system performance. The following conclusions are results of the coverability graph analysis:

- The net is strongly bounded, live but it is not fair.
- Mutually exclusive routes are never set at the same time.
- Routes are set only if there is a need to do so, and are unset immediately after the corresponding train leaves the suitable track segments.
- The *way free* signal is displayed only if a suitable route is set and the *stop* signal is set only as a result of trains mowing.
- Trains never ride through taken track segments.

It should be also emphasized that the verification of the decision table (at its design stage) has also great influence on the presented model properties. The complete and deterministic decision table guarantees correct performance of the *RequestHandling* transition.

The design and verification of the presented model has been done with a software support. CASE tools for RTCP-nets called *Adder Tools* are being developed at AGH University of Science and Technology in Krakow. *Adder Tools* contain:

- *Adder Designer*- for design and verification of rule-based systems;
- *Adder Editor*- for design of RTCP-nets;
- *Adder Simulator*- for simulation of RTCP-nets.

Some preliminary version of the software and more information about it can be found at *http://adder. ia. agh. edu.pl.*

5. Conclusions

RTCP-nets are an adaptation of CP-nets to make modelling and verification of embedded systems easier and more efficient. Based upon the experience with application of CP-nets for embedded systems' modelling, some modifications were introduced in order to make timed CP-nets more suitable for this purpose. The main advantage of the presented formalism is the new time model. Together with transitions' priorities, the time model enable designers direct modelling of task priorities, timeouts, etc. that are typical for concurrent programming.

The next advantage of RTCP-nets is the possibility of analysis of model properties with coverability graphs. Timed CP-nets can be also used to model embedded systems. A few different analysis methods have been proposed for untimed CP-nets but analysis of the timed ones may be difficult. In most cases, the state space of a live timed CP-net is infinite, so it is impossible to construct a full reachability graph that allows to analyse timing properties. To reduce such an infinite state space a few kinds of reduced reachability graphs have been defined, for example: graphs with stubborn sets, (Kristensen & Valmari 1998), graphs with equivalence classes (Jorgensen & Kristensen 1997), graphs with symmetries (Jorgensen & Kristensen 1999) and others. In most cases, analysis of time properties is impossible or limited significantly. In case of RTCP-nets coverability graphs can be used for these purposes. It has been proved (Szpyrka 2006a) that for strongly bounded RTCP-nets we may construct a finite coverability graph. Such a graph can be used for the analysis of typical Petri nets' properties as well as timing ones.

The other advantage of RTCP-nets is the way hierarchical models are constructed. Using of

the canonical form speeds up and facilitate drawing of models. Moreover, some parts of a model can be generated automatically, because they have precisely defined structure.
Furthermore, an RTCP-net in canonical form represents the structure of a modelled system, its dynamic and also functional relationships. Each part (object) of the modelled system and each system activity is represented by a distinguished part of the corresponding RTCP-net. Hence, it is possible to identify parts of Ada source code that should be defined e.g.: tasks, protected objects, suspending objects, etc. Some aspects of the transformation from the formal RTCP-net model into an implementation in Ada 2005 programming language can be found in (Szpyrka 2006b). The received source code meets the requirements of the Ravenscar profile (Burns et al. 2003). The transformation algorithm is the first step toward working out tools for automatic transformation of an RTCP-net into Ada source code framework. The algorithm has been successfully used for generation Ada 2005 source code for the railway traffic management system presented in the chapter.
Our future plans will focus on the development of Adder tools capabilities (e.g. wizards for automatic generation of some part of models) and implementation of analysis methods. Development of the software seems to be the most important task to put the presented theory into practice.

6. References

Bacherini, S.; Bianchi, S.; Capecchi, L; Becheri, C.; Felleca, A.; Fantechi, A. & Spinicci, E. (2003). Modelling a railway signalling system using SDL, *Proceedings of FORMS 2003 Symposium on Formal Methods for Railway Operation and Control Systems,* pp. 107-113, ISBN 963-9457-450, Budapest, Hungary, May 2003, L'Harmattan Hongrie, Budapest

Banci, M.; Fantechi, A. & Gnesi, S. (2004). The role of formal methods in developing a distributed railway interlocking system, *Proceedings of the 5th Symposium on Formal Methods for Automation and Safety in Railway and Automotive Systems (FORMS/FORMAT 2004),* pp. 220-230, ISBN 3-9803363-8-7, Braunschweig, Germany, December 2004, Technical University of Braunschweig

Barnes, J. (2006). *Programming in Ada 2005,* Addison Wesley, ISBN 0-321-34078-7, Harlow, England

Burns, A.; Dobbing, B. & Vardanega, T. (2003). *Guide for the Use of the Ada Ravenscar Profile in High Integrity Systems,* Technical Report No. YCS-2003-348, University of York

Cheng, A. M. K. (2002). *Real-time Systems. Scheduling, Analysis, and Verification,* Wiley Interscience, ISBN 978-0-471-18406-5, New Jersey

Jensen, K. (1992-1997). *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use,* Vol. 1-3, Springer-Verlag, ISBN 3-540-62867-3, Berlin

Jorgensen, J.B. & Kristensen, L.M. (1997). Verification of coloured Petri nets using state spaces with equivalence classes, *Proceedings of the Workshop on Petri Nets in System Engineering, Modelling, Verification and Validation,* pp. 20-31, ISBN 3-89586-597-4, Hamburg, Germany, September 1997, Universitat Hamburg

Jorgensen, J.B. & Kristensen, L.M. (1999). Computer aided verification of Lamport's fast mutual exclusion algorithm using coloured Petri nets occurrence graphs with symmetries. *IEEE Transactions on Parallel and Distributed Systems,* Vol. 10, No. 7 (1999) 714-732, ISSN 1045-9219

Kristensen, L.M. & Valmari, A. (1998). Finding stubborn sets of coloured Petri nets without unfolding, *Proceedings of the 19th International Conference on Application and Theory of Petri Nets,* pp. 104-123, ISBN 3-540-64677-9, Lisbon, Portugal, June 1998, LNCS, Vol. 1420, Springer-Verlag, London

Murata, T. (1989). Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE,* Vol. 77, No. 4, (April 1989) 541-580, ISSN 0018-9219

Sommerville, I. (2004). *Software Engineering,* Pearson Education Limited, ISBN 0-321-21026-3, Boston

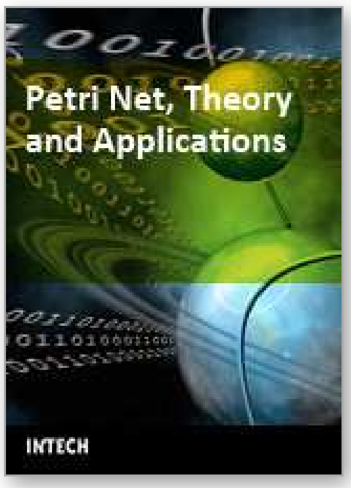
Szpyrka, M. (2006a). Analysis of RTCP-nets with reachability graphs. *Fundamenta Informaticae,* Vol. 74, No. 2-3, (2006) 375-390, ISSN 0169-2968

Szpyrka, M. (2006b). Development of embedded systems - from RTCP-net model to Ada code. *Concurrency, Specification and Programming CS&P'2006,* Vol. 2, pp. 231-242, ISSN 0863-095X, Wandlitz, Germany, September 2006, Informatik Berichte, No. 206, Humboldt-Universitatzu Berlin

Szpyrka, M. & Szmuc, T. (2006a). D-nets- Petri net form of rule-based systems. *Foundations of Computing and Decision Sciences,* Vol. 31, No. 2 (2006) 157-167, ISSN 0867-6356

Szpyrka, M. & Szmuc, T. (2006b). Verification of automatic train protection systems with RTCP-nets. Proceedings of the 25th International Conference on Computer Safety, Security and Reliability, pp. 344-357, ISBN 3-540-45762-3, Gdansk, Poland, September 2006, LNCS, Vol. 4166, Springer-Verlag, Berlin

Szpyrka, M. & Szmuc, T. (2006c). Integrated approach to modelling and analysis using RTCP-nets. *IFIP International Federation for Information Processing,* Vol. 227, 115-120, ISBN 978-0-387-39387-2, Springer, New York



Petri Net, Theory and Applications
Edited by Vedran Kordic

ISBN 978-3-902613-12-7
Hard cover, 534 pages
Publisher I-Tech Education and Publishing
Published online 01, February, 2008
Published in print edition February, 2008

Although many other models of concurrent and distributed systems have been de- veloped since the introduction in 1964 Petri nets are still an essential model for concurrent systems with respect to both the theory and the applications. The main attraction of Petri nets is the way in which the basic aspects of concurrent systems are captured both conceptually and mathematically. The intuitively appealing graphical notation makes Petri nets the model of choice in many applications. The natural way in which Petri nets allow one to formally capture many of the basic notions and issues of concurrent systems has contributed greatly to the development of a rich theory of concurrent systems based on Petri nets. This book brings together reputable researchers from all over the world in order to provide a comprehensive coverage of advanced and modern topics not yet reflected by other books. The book consists of 23 chapters written by 53 authors from 12 different countries.

How to reference
In order to correctly reference this scholarly work, feel free to copy and paste the following:

Marcin Szpyrka (2008). Modelling and Analysis of Real-Time Systems with RTCP-Nets, Petri Net, Theory and Applications, Vedran Kordic (Ed.), ISBN: 978-3-902613-12-7, InTech, Available from: http://www.intechopen.com/books/petri_net_theory_and_applications/modelling_and_analysis_of_real-time_systems_with_rtcp-nets