Marcin Szpyrka*

FAST AND FLEXIBLE MODELLING OF REAL-TIME SYSTEMS WITH RTCP-NETS

A large number of formalisms has been proposed for real-time systems modelling. However, formal methods are not widely used in industrial software development. Such a situation could be treated as a result of a lack of suitable tools for fast designing of a model, its analysis and modification. RTCP-nets have been defined to facilitate fast modelling of embedded systems incorporating rule-based systems. Computer tools that are being developed for RTCP-nets, use a template mechanism to allow users to design models and manipulate its properties fast and effectively. Both theoretical and practical aspects of RTCP-nets are presented in the paper.

Keywords: *RTCP-nets, real-time systems, modelling.*

EFEKTYWNA METODA MODELOWANIA SYSTEMÓW CZASU RZECZYWISTEGO Z WYKORZYSTANIEM SIECI RTCP

Pomimo różnorodności dostępnych formalizmów wspomagających modelowanie i analizę systemów czasu rzeczywistego, metody formalne nie są powszechnie wykorzystywane w procesie wytwarzania oprogramowania. Przyczyną takiej sytuacji może być brak odpowiednich narzędzi, które umożliwiałyby szybkie stworzenie modelu, jego analizę i łatwą modyfikację. RTCP-sieci zostały zdefiniowane w celu umożliwienia szybkiego modelowania systemów wbudowanych, w szczególności działających przy wykorzystaniu systemów regułowych. Rozwijane narzędzia wspierające wykorzystanie RTCP-sieci wykorzystują mechanizm szablonów umożliwiający szybkie stworzenie modelu i łatwą manipulację jego własnościami. W artykule przedstawiono zarówno teoretyczne, jak i praktyczne aspekty RTCP-sieci.

Słowa kluczowe: *sieci RTCP, systemy czasu rzeczywistego, modelowanie.*

1. Introduction

Real-time software is characterized by the presence of timing constrains on computations, such as deadlines on process execution. Therefore, the correctness of real-time software depends both on its functional correctness and on its time performance ([4, 1, 3]). Real-time software usually requires more extensive validation and testing

*Institute of Automatics, AGH University of Science and Technology, Kraków, Poland, mszpyrka@agh.edu.pl

than traditional software. Testing and debugging cannot guarantee that a system is correct. Software and system quality, consistency and integrity can be improved by formalising different products and processes in the development cycle. The use of formal methods is increasing in the area of critical systems development, where emergent system properties such as safety, reliability and security are very important.

RTCP-nets ([6, 8]) are formal mathematical construction used for modelling and analysis of embedded systems. They provide a framework for design, specification, validation, and verification of such systems. Based upon the experience with application of CP-nets ([2]) for real-time systems' modelling, some modifications were introduced in order to make CP-nets more suitable for this purpose. The new subclass of coloured Petri nets was defined to meet the following requirements.

- To speed up and facilitate drawing Petri nets.
- To equip Petri nets with capability of direct modelling of elements such as task priorities, over-timing, etc.
- To make formal analysis of high level Petri nets easier.

The first description of RTCP-nets was presented in [6]. However, some new elements have been introduced into the definition in [8]. RTCP-nets are supported by computer tools called *Adder Tools*. The tools use a *template mechanism* (similar to class templates in C++). Page templates can be treated as components used to construct an RTCP-net. Users can freely manipulate parameters of a page template and relationships between them to guarantee desired features of a modelled system. Therefore, it is easy to experiment on different versions of the same model with very little additional effort. A short description of *Adder Tools* can be found in [8].

2. RTCP-nets

The definition of RTCP-nets is based on the definition of coloured Petri nets presented in [2]. In most cases, notions typical for CP-nets are defined in similar way for RTCP-nets. In some cases, different symbols are used or some elements typical for RTCP-nets appear. Therefore, in this section, not only is presented a definition of RTCP-nets but also definitions of most important notions related to Petri nets are reminded.

The purpose of this definition is to give a mathematically sound and unambiguous definition of RTCP-nets. It is not necessary to fix the concrete syntax in which the net expressions are written. Thus it will be only assumed that such a syntax coexists with well-defined semantics and it is possible in an unambiguous way to talk about the following notions.

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, which can be obtained by evaluating the expression.

For any given set of variables V, the type of the set of variables is defined in the following way:

$$\mathcal{T}(V) = \{\mathcal{T}(v)\colon\ v \in V\} \tag{1}$$

Let *Bool* denote the boolean type (containing the elements $\{false, true\}$, and having the standard operations from propositional logic).

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* $\mathcal{N} = (\Sigma, P, P_M, T, T_M, A, C, G, I, E_M, E_S, M_0, S_0)$ *satisfying the following requirements.*

(1) Σ *is a finite set of non-empty* types (colour sets).
(2) P *is a finite set of* places.
(3) $P_M \subseteq P$ *is a set of* main places ($P_A = P - P_M$ *is a set of* auxiliary places).
(4) T *is a finite set of* transitions *such that* $P \cap T = \emptyset$.
(5) $T_M \subseteq T$ *is a set of* main transitions ($T_A = T - T_M$ *is a set of* auxiliary transitions).
(6) $A \subseteq (P \times T) \cup (T \times P)$ *is a* flow relation *such that:*
$A \cap ((P_M \times T_A) \cup (T_A \times P_M)) = \emptyset$
(7) $C\colon P \to \Sigma$ *is a* type function*, which maps each place to its type.*
(8) G *is a* guard function*, which maps each transition to an expression such that* $\forall t \in T\colon \mathcal{T}(G(t)) \subseteq Bool \wedge \mathcal{T}(\mathcal{V}(G(t))) \subseteq \Sigma$.
(9) $I\colon T \to \mathbb{N} \cup \{0\}$ *is a* priority function*, which maps each transition to a non-negative integer called* transition priority.
(10) E_M *is an* arc expression function*, which maps each arc to a weight expression such that:*
$\forall a \in A\colon \mathcal{T}(E_M(a)) \subseteq C(P(a)) \wedge \mathcal{T}(\mathcal{V}(E_M(a))) \subseteq \Sigma$.
(11) E_S *is an* arc time expression function*, which maps each arc to a time expression such that:*
- $\forall a \in A\colon P(a) \in P_M \Rightarrow (\mathcal{T}(E_S(a)) \subseteq \mathbb{R}^+ \cup \{0\} \wedge \mathcal{T}(\mathcal{V}(E_S(a))) \subseteq \Sigma)$.
- $\forall a \in A\colon P(a) \in P_A \Rightarrow E_S(a) = 0$.

(12) M_0 *is an* initial marking*, which maps each place to a multi-set* $M_0(p) \in 2^{C(p)^*}$*, where* $2^{C(p)^*}$ *denotes the set of all multi-sets over the set* $C(p)$.
(13) $S_0\colon P \to \mathbb{R}$ *is a* initial time stamp function*, which maps each place to a real value called* time stamp *such that* $\forall p \in P_A\colon S_0(p) = 0$.

It seems that the above definition needs some explanations.

(1). The set Σ determines the types, operations and functions that can be used in the net inscriptions.

(2), (3). The set of places P of an RTCP-net 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. Auxiliary places are used on subpages, which describe system activities in detail.

(4), (5). 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 transitions are used on subpages, which describe system activities in detail.

(6). The set of arcs is defined as a relation due to the fact that multiple arcs are not allowed. Main places may be connected to main transitions only.

(8). The guard function maps each transition to an expression of type Boolean. Such an expression may contain variables, but they must have types that belong to Σ.

(9). The priority function assigned to transitions allows direct modelling of deterministic choice.

(10). The arc expression function E_M maps each arc to a weight expression such that, for any arc, a, each evaluation of the arc expression must yield a single token belonging to the type that is attached to the place $P(a)$.

(11). The arc time expression function E_S maps each arc, a, to a time expression such that:

- if the place $P(a)$ is a main place, then each evaluation of the time expression must yield a non-negative real value;
- if the place $P(a)$ is an auxiliary place, then the corresponding time expression is equal to 0.

(12). The initial marking determines the initial distribution of tokens.

(13). Time stamps are attached to places (instead of to tokens). It is assumed that time stamps attached to auxiliary places are always equal to 0.

3. Behaviour of RTCP-nets

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-unit old. It is possible to specify how old a token should be so that a transition may consume it.

Let $V = P \cup T$ denote the set of all nodes of an RTCP-net. $In(x)$ and $Out(x)$ denote the set of *input* and *output nodes* of the node x, i.e.:

$$\begin{aligned} In(x) &= \{y \in V \colon (y,x) \in A\} \\ Out(x) &= \{y \in V \colon (x,y) \in A\} \end{aligned} \tag{2}$$

Let $\mathcal{V}(t)$ be the set of variables occurring in the expressions of arcs surrounding the transition t and in the guard of the transition.

Definition 2. *A* binding *of a transition $t \in T$ is a function b defined on $\mathcal{V}(t)$, such that:*

$$\forall v \in \mathcal{V}(t) \colon b(v) \in \mathcal{T}(v) \wedge G(t)_b = \text{true} \tag{3}$$

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*. $G(t)_b$ denotes the evaluation of the guard expression in the binding b. Similarly, $E_M(a)_b$ (where $a \in A$) denotes the evaluation of the weight expression in the binding b, and $E_S(a)_b$ denotes the evaluation of the time expression.

Definition 3. *A* marking *of an RTCP-net $\mathcal{N}$ is a function M defined on the set of places P, such that:*

$$\forall p \in P\colon M(p) \in 2^{C(p)^*} \tag{4}$$

Definition 4. *A* time stamp function *of an RTCP-net $\mathcal{N}$ is a function S defined on the set of places P, such that:*

$$\forall p \in P\colon S(p) \in \mathbb{R} \tag{5}$$

Let P be an ordered set, $P = \{P_1, P_2, \ldots, P_n\}$. Both, a marking M and time stamp function S, can be represented by vectors with $|P|$ entries. Therefore, the phrase *a time stamp vector* (a time vector) will be used instead of a time stamp function.

Definition 5. *A* state *of an RTCP-net is a pair (M, S), where M is a marking and S is a time stamp function. The pair (M_0, P_0) is the* initial state.

Definition 6. *A transition $t \in T$ is* enabled *in a state (M, S) in a binding b iff the following condition holds:*

$$\begin{aligned} &\forall p \in In(t)\colon E_M(p,t)_b \in M(p) \wedge E_S(p,t)_b \leq -S(p) \\ &\forall p \in Out(t)\colon S(p) \leq 0 \end{aligned} \tag{6}$$

and for any other transition $t' \neq t$ that satisfies the above two conditions, $I(t) \geq I(t')$ or $In(t) \cap In(t') = Out(t) \cap Out(t') = \emptyset$.

A transition $t \in T$ is enabled in a state (M, S) means that the transition t is enabled in a state (M, S) in one of its bindings.

Definition 7. *If a transition $t \in T$ is enabled in a state (M_1, S_1) in a binding b it may* occur*, changing the state (M_1, S_1) to another state (M_2, S_2), such that:*

$$M_2(p) = \begin{cases} M_1(p) - \{E_M(p,t)_b\} & \text{for } p \in In(t) - Out(t) \\ M_1(p) - \{E_M(p,t)_b\} \cup \{E_M(t,p)_b\} & \text{for } p \in In(t) \cap Out(t) \\ M_1(p) \cup \{E_M(t,p)_b\} & \text{for } p \in Out(t) - In(t) \\ M_1(p) & \text{otherwise} \end{cases} \tag{7}$$

$$S_2(p) = \begin{cases} E_S(t,p)_b & \text{for } p \in Out(t) \\ 0 & \text{for } p \in In(t) - Out(t) \\ S_1(p) & \text{otherwise} \end{cases} \tag{8}$$

A global clock is used to measure time. Every time the clock goes forward, all time stamps are decreased by the same value. Therefore, each occurrence of a transition may change both vectors M and S, while each clock step changes the vector S but not the vector M.

4. Hierarchical RTCP-nets

There is no a special definition of hierarchical RTCP-nets. Hierarchical RTCP-nets are based on hierarchical CP-nets. Substitution transitions and fusion places 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 RTCP-net, which usually gives a more precise and detailed description of the activity represented by the substitution transition. The substitution transition is called a *super node*, and the page containing it is called a *super page*. Any page attached to a substitution transition is called a *subpage*. The interface between a subpage and the corresponding superpage (substitution transition) is specified by *port assignments*. The assignment connects each *socket node* in the superpage (i.e., places surrounding the substitution transition) with a *port node* in the subpage (i.e., one of the places with *port-tag*). Three different kinds of port-tags are possible. An *In-tag* indicates that the port node must be connected to a socket node, which is an input but not output node of a substitution transition. *Out-tag* and I/O-*tag* are defined in similar way. In RTCP-nets each socket node must have only one port node assigned and vice versa. A socket node and the corresponding port node must have the same types and initial markings. A socket node name covers the name of the corresponding port node, so for formal considerations only the socket node name is important. A port node tag is determined by the connections between the corresponding socket node and the substitution transition. Therefore, 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. A fusion name covers names of places belonging to the fusion set, so for formal considerations only the fusion name is important.

Hierarchical RTCP-nets may be composed of four types of pages with precisely defined structure: primary place pages, primary transition pages, MG-subpages and D-nets. *Primary place pages* are used to represent active objects and their activities. They are oriented towards objects presentation and are top level pages. *Primary transition pages* are oriented towards activities presentation and are second level pages. D-nets are used to represent decision tables and MG-subpages to glue such D-nets into a model. All these types of pages have precisely defined structure so it is possible to generate an RTCP-net (or part of it) automatically. Furthermore, page templates may be used while designing an RTCP-net model. They are used to make designing of

RTCP-net models easier and faster (similar to class templates in C++). More detailed description of these types of pages is presented in the next section. Using only these types of pages facilitates analysis of RTCP-nets, e.g. such a net is *safe* (1-bounded). Models composed of these types of pages according to algorithms presented in the next section will be said to be in a *canonical form*.

5. Modelling with RTCP-nets

An abstract model of a control system will be used to present the different types of pages and the modelling methods. Let us consider a control system with two processors and three sensors. The sensors serially make some measurements and send results to processors. Usually, the first processor makes computations but if it takes too much time the second processor helps. Results of these computations are presented on an LCD display.

It is first necessary to distinguish between *active objects*, i.e., objects performing activities, and *passive objects*, 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. There are five active objects in the considered system (two processors and four sensors) and one passive object (the LCD display). Definitions of types and declarations of variables used in the model are as follows:

```
color Value    = int with 0..9;
color State    = with free | data | result;
color Level    = int with 0..2;
color Procesor = product State * Level * Value * Value * Value;
color TempIn   = product Value * Value * Value;
var y1 : Level;
var y2 : Level;
var x1 : Value;
var x2 : Value;
var x3 : Value;
```

5.1. Primary place pages

Primary place pages are used to represent active objects and their activities. Such a page is composed of one main place that represents the object and one main transition for each object activity. Each connection between the main place and a main transition creates a self-loop. Passive object are represented by single main places or fusions of places and they are not placed on separate pages. Each main place contains exactly one token in the initial marking and auxiliary places remain empty.

Each *Processor* is characterized by five attributes:

- the state of the processor (*free* – the processor is ready to collect new data, *data* – the processor has collected data from sensors, *result* – the processor has computed the result that should be displayed next);

- the computed result (or 0 if the result is not computed yet);
- three values collected from sensors (or 0 if the values are not collected yet).

The primary place page for a processor is shown in Figure 1. The type of the place *Proc%* is set to *Processor* while the initial marking is set to the tuple $(free, 0, 0, 0, 0)$. *Processor* can perform three activities:

- *Read* – collects data from sensors;
- *Compute* – computes the result;
- *Display* – displays the result on the *LCD*.

Each of these activities is represented by a main transition that is usually a substitution transition. (This is denoted by the label HS.) Priorities of these transitions are set to 2. Durations for actions such as *Read* and *Display* are so small that they can be disregarded. A non-zero time period is considered only for the *Compute* action. It takes from 4 to 7 time-units to complete the action (Time expressions follow the @ sign. A lack of a time expresion denotes the default, 0 value.)

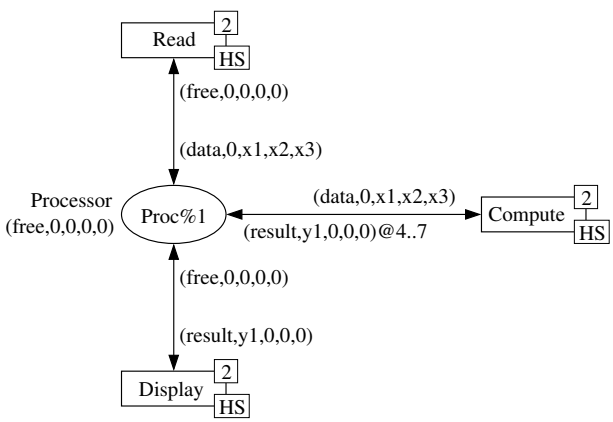

Fig. 1. Page template used to generate primary place pages for processors

The page presented in Figure 1 is in fact a *page template* with one parameter (%1). A parameter can take any string value. The parameter denotes the number of a processor. If pages are similar then only one *page template* with a number of parameters may be designed. All the similar pages will be generated according to parameter values as instances of the template. Values of the parameters are determined in the hierarchy composition stage. Page templates are not treated as a new element of the RTCP-net definition, they are used to make designing of RTCP-net models faster and more flexible.

The page template used to generate primary place pages for sensors is shown in Figure 2. The only parameter (%1) stands for a sensor number. It is used both in the place name and the fusion set name. A *Sensor* can perform only one activity (*Check*) that denotes a measurement process.

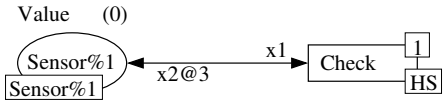

Fig. 2. Page template used to generate primary place pages for sensors

5.2. Primary transition pages

Transitions placed in primary place pages are usually substitution transitions. For each of these substitution transitions a *primary transition page* is drawn. Such a primary transition 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. Each connection between the main transition and a main place creates a self-loop. The page template used to generate primary transition pages for *Read* activity is presented in Figure 3.

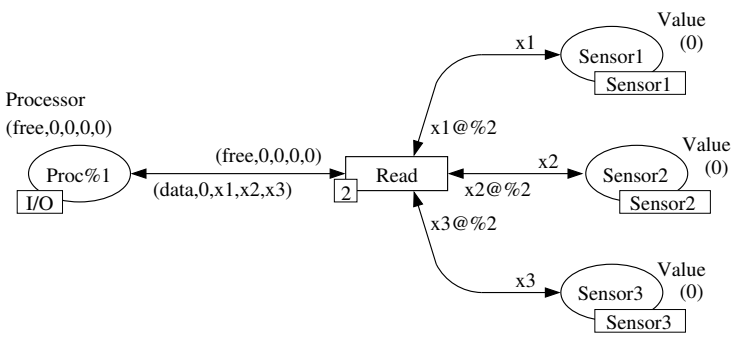

Fig. 3. Page template used to generate primary transition pages for *Read* activity

Let us consider the interface between a primary place and a primary transition page. The place from the corresponding primary place page is an input/output socket node. The primary transition page contains an input/output port node (treated as a main place) that is assigned to the socket node from the primary place page. This type of connection between these two primary pages is treated as kind of fusion sets. All other places, the values of which are necessary to execute the activity, are represented by main places belonging to corresponding fusion sets, if necessary.

The page template shown in Figure 3 contains two parameters. The first of them stands for the processor number, the second for arc time expressions for three input arcs. Let us assume that the value of the parameter %2 is equal to 0 for the first *Processor* and to 3 for the second one. It means that the transition *Read* of the first *Processor* may remove tokens from places *Sensor1..3* if they are accessible, but the transition *Read* of the second *Processor* has to be postponed till the tokens in the places *Sensor1..3* are *3 time-units old*. Therefore, if within duration of 3 time-units the first transition does not remove the tokens, the second transition will do it.

Primary transition page for the *Display* activity is shown in Figure 4. This is an example of a simple page without any parameters.

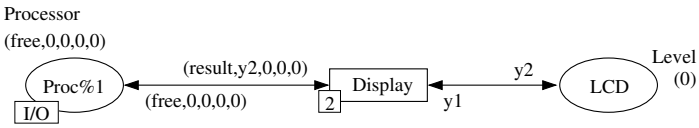

Fig. 4. Primary transition page for the *Display* activity

6. D-nets

Sometimes, it is necessary to attach a rule-based system to a transition, e.g. if a controller is developed. A particular form of such a rule-based systems is a decision table with generalized decision rules. To construct a decision table, we have to define a set of attributes selected to describe important features of the system under consideration, i.e., conditions and actions. The set of attributes is divided into two parts, *conditional* and *decision attributes*. The set of rules is represented as a decision table with columns labelled with attributes names and rows labelled with rule names. Each cell of a decision table should contain a formula, which evaluates to a boolean value for conditional attributes, and to a single value (which belongs to the corresponding domain) for decision attributes. Each row of such a decision table is called a *general decision rule*, because non-atomic values of attributes are allowed.

Table 1
Decision table for the *Compute* activity

	Sensor1	**Sensor2**	**Sensor3**	**Level**
R1	Sensor1 <= 5	Sensor2	Sensor3	0
R2	Sensor1 = 6	Sensor2	Sensor3 > Sensor2	0
R3	Sensor1 = 6	Sensor2	Sensor3 <= Sensor2	1
R4	Sensor1 = 7	Sensor2 < 5	Sensor3	1
R5	Sensor1 = 7	Sensor2 >= 5	Sensor3	2
R6	Sensor1 >= 8	Sensor2	Sensor3	2

A simple decision table that may be used to compute the result to be displayed on the *LCD* is presented in Table 1. The table contains three conditional and one decision attribute. Let us consider the first rule. The rule is used if the value of the *Sensor1* is equal to or less than 5. Values of other attributes are unimportant. The decision (the value of the *Level* attribute) is equal to 0.

In order to assure reliable and efficient performance, analysis and verification of selected qualitative properties (such as completeness, consistency (determinism) and optimality) should be carried out. For intuition, a decision table is considered to be *complete* if for any possible input situation at least one rule can produce a decision.

A decision table is *deterministic* if no two different rules can produce different results for the same input situation. The last property means that any dependent (non-necessary) rules were removed. For more details see [5].

D-nets are used to represent decision tables in a Petri net form. They are used to verify decision table properties and constitute parts of an RTCP-net model. A D-net contains two places: a *conditional* and a *decision place*. Each positive 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-net form of the considered decision table in presented in Figure 5.

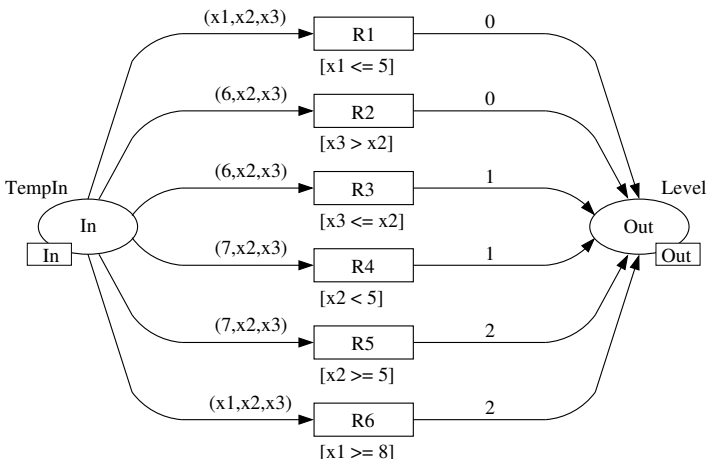

Fig. 5. D-net form of the Table 1

Analysis of decision tables is carried out after their transformation into D-nets ([5]). Methods typical for Petri nets are used for that purpose. The D-net presented in Figure 5 (the decision table 1) is complete, consistent and semi-optimal.

7. Marked graph subpages (MG-subpages)

The D-net shown in Figure 5 should be attached to the *Compute* transition. In such a case, an extra subpage called *MG-subpage* is used as an interface to glue the corresponding D-net into a model. It is used to gather all necessary information for the D-net and to distribute results of the D-net activity.

An MG-subpage 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 *In* and *Out*. All elements placed between those transitions are auxiliary ones. An MG-subpage must form a *marked graph*, i.e. each place must have exactly one input and one output transition. Besides,

each occurrence of the *In* main transition must be followed by a sequence of transitions' occurrences such that the last of them is the *Out* transition and all the others are auxiliary transitions. It should be underlined that there is 0 time-units between occurrences of the *In* and *Out* main transitions. MG-subpages may be also used to represent an algorithm without any D-net. Primary transition page for the *Compute* activity is shown in Figure 6, while an MG-subpaged used to glue the D-net into the model is presented in Figure 7.

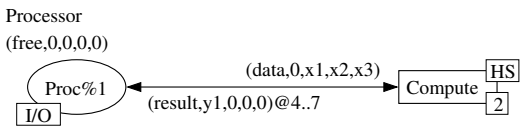

Fig. 6. Primary transition page for the *Compute* activity

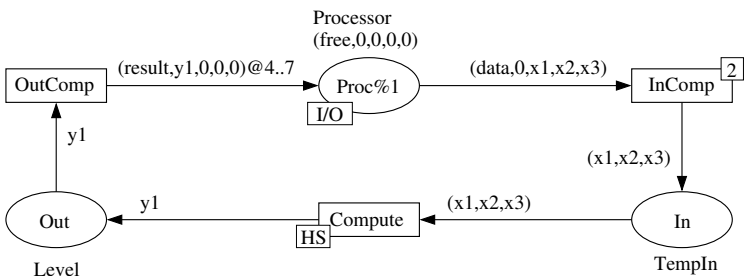

Fig. 7. MG-subpaged used to glue the D-net shown in Figure 5 into the model

8. Hierarchy graph

Drawing of all pages or page templates completes the first stage of the RTCP-net design process. Then, some connections among pages must be constituted to compose one hierarchical model. All connections among pages are presented using a page hierarchy graph. Such a graph for the considered model is presented in Figure 8. Each node represents a simple page, and an arc represents a connection between a subpage and its substitution transition. If a page is generated from a template, values of its parameters are placed next to the page node.

Just changing the parameter values in the graph hierarchy page is enough to change some features of a modelled system. Therefore, it is easy to experiment on different versions of the same model with very little additional effort. It is also very easy to reorganize page templates in order to model a different structure of a system.

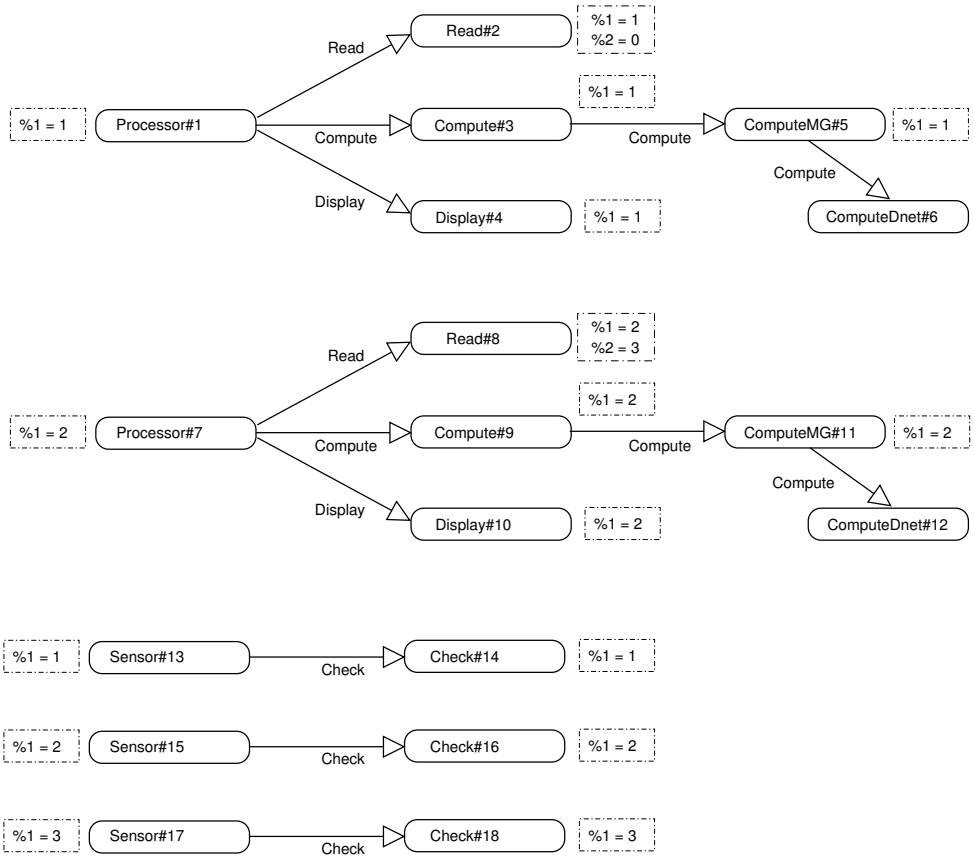

Fig. 8. Page hierarchy graph

9. Adder Tools

Computer tools for RTCP-nets, called Adder Tools (Analysis anD Design of Embedded Real-time systems), are being developed at AGH University of Science and Technology in Kraków, Poland. Adder Tools contain: Adder D-nets Designer for the design and verification of rule-based systems, Adder Editor for designing RTCP-nets, and Adder Simulator for the simulation of RTCP-nets. *Adder Tools home page*, hosting information about current status of the project, is located at *http://adder.ia.agh.edu.pl*.

10. Summary

An RTCP-net approach to modelling of real-time systems was presented in the paper. RTCP-nets are subclass of time coloured Petri nets, but some modifications were introduced into the definition of RTCP-nets and the design process. RTCP-nets to-

gether with computer tools allow users to design models and manipulate its properties fast and effectively. A formal definition of RTCP-nets has been presented in the paper and some practical aspects of RTCP-nets have been put forward.

Acknowledgements

Supported by KBN Research Project No 4 T11C 035 24.

References

[1] Cheng A. M. K.: *Real-time systems. Scheduling, Analysis, and Verification*. New Yersey, Wiley Interscience 2002

[2] Jensen K.: *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use*. Vol. 1–3, Springer 1992–97

[3] Sommerville I.: *Software Engineering*. Pearson-Addison Wesley 2004

[4] Szmuc T.: *Zaawansowane metody tworzenia oprogramowania systemów czasu rzeczywistego*. Kraków, CCATIE 1998

[5] Szpyrka M.: *Algebraic-Graphical Methods of Knowledge Analysis and Verification – theoretical aspects*. In: Cybernetics and Systems 2002 – Proceedings of the Sixteenth European Meeting on Cybernetics and Systems Research, Vienna, Austria, 2002, p 812–817

[6] Szpyrka M., Szmuc T.: *RTCP-nets as a Tool for Real-Time Systems Modelling and Analysis*. In: Colnaric M., Adamski M., Węgrzyn M. (Eds): Real-Time Programming 2003 – Proceedings of the 27th IFAC/IFIP/IEEE Workshop on Real-Time Programming, Łagów, Poland 2003, p. 21–26

[7] Szpyrka M., Szmuc T., Matyasik P., Szmuc W.: *Formal Approach to Modelling of Real-time Systems with RTCP-nets*. Foundation of Computing and Decision Sciences, vol. 30, no. 1, 2005, pp. 61–71

[8] Szpyrka M., Szmuc T.: *New Time Model and Design Method for RTCP-nets*. The Proceedings of 29th IFAC/IFIP Workshop on Real-Time Programming WRTP 2004, Istanbul, Turkey, 2004