Decision Tables in Petri Net Models

Marcin Szpyrka and Tomasz Szmuc

Institute of Automatics,
AGH University of Science and Technology,
Al. Mickiewicza 30, 30-059 Kraków, Poland
mszpyrka@agh.edu.pl, tsz@agh.edu.pl

Abstract. Many monitoring and control computer systems contain a rule-based system as a part of them. Such a rule based system is used to determine which actions should be taken depending on the data collected from sensors. For both embedded and rule-based systems many different approaches have been proposed, but it is hardly possible to find a formalism that can cope with both of them. The paper deals with a problem of including decision tables into colour Petri net models. A few kinds of decision tables are considered and methods of transformation them into coloured Petri nets form called D-nets are presented. Both non-hierarchical and hierarchical D-nets are considered in the paper.

1 Introduction

The use of formal methods for embedded system development is motivated by the expectation that performing appropriate mathematical analyses can contribute to the software quality. Formal methods are usually used in the development of safety-critical systems, i.e. systems that may result in injury, loss of life or serious environmental damage upon their failure [9]. The high cost of safety-critical systems failure means that trusted methods and techniques must be used for development. For such systems, the costs of verification and validation are usually very high (more than 50% of the total system development cost). Using of formal methods can reduce the amount of testing and ensure more dependable products [3].

Multiple embedded systems are control systems that monitor quantities of interest in an environment. In response to changes in the monitored quantities they perform control operations or other externally visible actions [1]. The process of making decision which actions should be performed may be based on a rule-based system that is incorporated into such an embedded system. Thus, formal methods are useful for modelling of such systems only if rule-based systems can be also expressed in the selected formalism.

Rule-based systems can be represented in various forms, e.g. decision tables, decision trees, extended tabular trees (XTT, [7]), Petri nets [4] etc. An interesting comparison of different forms of rule-based systems description can be found in [6]. Decision tables seem to be the most popular form of rule-based systems presentation. They vary widely in the way the condition and decision entries are represented. The entries can take the form of simple true/false values, atomic values of different types, non-atomic values or even fuzzy logic formulas.

A decision table represents a set of decision rules that can be given explicitly by an expert or generated from analyzed data automatically, e.g. using rough sets approach

[2]. The paper deals with a problem of including of an already constructed decision table into a hierarchical colour Petri net model (CP-net [5] or RTCP-net [10]). Both decision tables with atomic values of attributes [6] and with generalized rules [11] are considered in the paper.

The paper is organized as follows. Decision tables with atomic values of attributes are described in section 2. Generalized decision tables are presented in section 3. Hierarchical D-nets are considered in section 4. A more practical example of a decision table is presented in section 5. Computer software for decision tables called *Adder Designer* is described is section 6. The paper is shortly concluded in the final section.

2 Decision Tables with Atomic Values of Attributes

Let's recall the definition of a decision table presented in [8]. At first sight a decision table can be treated as an extension of a knowledge representation system. Such a system is a pair $\mathcal{K} = (U, A)$, where U is a nonempty, finite set called the *universe*, and A is a nonempty, finite set of *attributes*. Every attribute $a \in A$ is a function $a\colon U \to V_a$, where V_a denotes the domain of a.

To transform a knowledge representation system into a decision table, we have to distinguish two subsets of A called *conditional* (C) and *decision* (D) attributes respectively. In case of a decision table the elements of the set U denote not any real objects, but are identifiers of decision rules. Hence the symbol R will be used instead of U. Therefore, a decision table is a tuple $\mathcal{T} = (R, A, C, D)$, where $C, D \subset A$.

Such a decision table is often called a *table with atomic values of attributes* (or *simple decision table*, [6]). To construct such a decision table, we draw a column for each conditional and decision attribute. Then, for every decision rule a row should be drawn. We fill cells so as to reflect which decisions are generated for each combination of conditions. An example of a simple decision table is shown in Tab. 1.

Table 1. Example of a simple decision table ($\mathcal{T}_1$)

	a	b	c	d	e
R1	1	2	1	1	1
R2	1	2	2	2	2
R3	2	1	1	1	1
R4	2	2	2	2	2
R5	3	1	2	2	1
R6	3	2	2	2	2
R7	4	1	1	2	1
R8	4	2	2	2	1

The table $\mathcal{T}_1$ contains three conditional and two decision attributes ($C = \{a, b, c\}$, $D = \{d, e\}$). In this case, for each attribute its domain is a subset of natural numbers, but in general other types are also possible (e.g. real, boolean, enumerated types, etc.)

To include the decision table $\mathcal{T}_1$ into a Petri net model (CP-net or RTCP-net), it must be first transformed into a D-net [11]. A D-net is a non-hierarchical coloured Petri net that represents a set of decision rules. It contains two places: a *conditional place* (input place) for values of conditional attributes and a *decision place* (output place) for values of decision attributes. Types for these places are defined as the Cartesian product of domains of conditional and decision attributes respectively.

Each decision rule is represented by one transition and its input and output arcs. The conditional part of a rule is represented by the expression attached to the input arc of the corresponding transition. Similarly, the decision part is represented by the expression attached to the output arc.

D-nets are used as the bottom level pages in hierarchical models [10]. The conditional and decision places are input and output ports respectively. The superpage (see [5]) for such a D-net is used to gather all necessary information for the D-net and to distribute the results of its activity. In other words, the superpage prepares a token that represents a sequence of values of conditional attributes and that is next placed on the conditional place. If at least one transition in the D-net is enabled, the token is removed from the conditional place and a new token that represents a decision is added to the decision place. Then the superpage removes the token and brings the decision into effect. The D-net form of the decision table $\mathcal{T}_1$ is shown in Fig. 1.

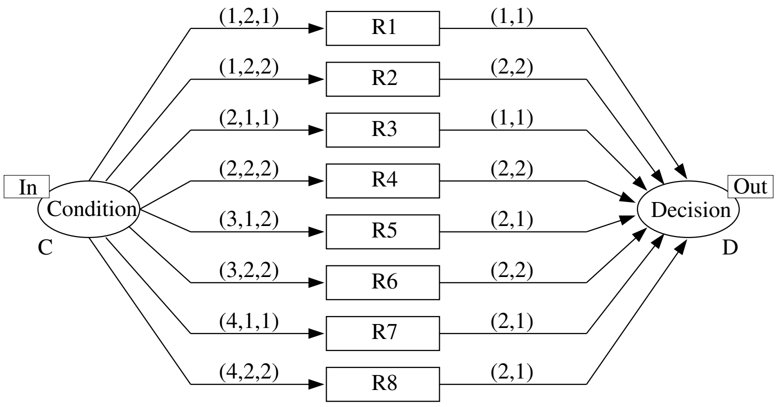


Fig. 1. D-net for the decision table $\mathcal{T}_1$

The decision table presented in Tab. 1 contains values for all conditional attributes. Methods based on the rough set theory can be used to reduce such a decision table. The reduction algorithm consists in the elimination of conditions from a decision table, which are unnecessary to make decisions specified in the table (see [8]).

Let's consider the decision tables presented in Tab. 2. Some redundant values of conditional attributes are omitted in the table. In such a case to transform a decision table into a D-net variables have to be used. In this case we need two variables for attributes a and b. An variable attached to an attribute $x \in C$ can take any value that belongs to the domain of the attribute.

Thus before the transformation algorithm can be applied, the table $\mathcal{T}_2$ is represented in the form shown in Tab. 3. For simplicity, the name of an variable is the same as the name of the corresponding attribute. The D-net form of the decision table $\mathcal{T}_2$ is shown in Fig. 2.

Table 2. Example of a simple decision table ($\mathcal{T}_2$)

	a	b	c	d
R1	1	1	1	1
R2	1	2	1	2
R3	4	1	1	2
R4	4	2	1	1
R5	2	–	1	1
R6	3	–	1	1
R7	–	–	2	2
R8	–	–	3	2

Table 3. Decision table $\mathcal{T}_2$ with variables

	a	b	c	d
R1	1	1	1	1
R2	1	2	1	2
R3	4	1	1	2
R4	4	2	1	1
R5	2	b	1	1
R6	3	b	1	1
R7	a	b	2	2
R8	a	b	3	2

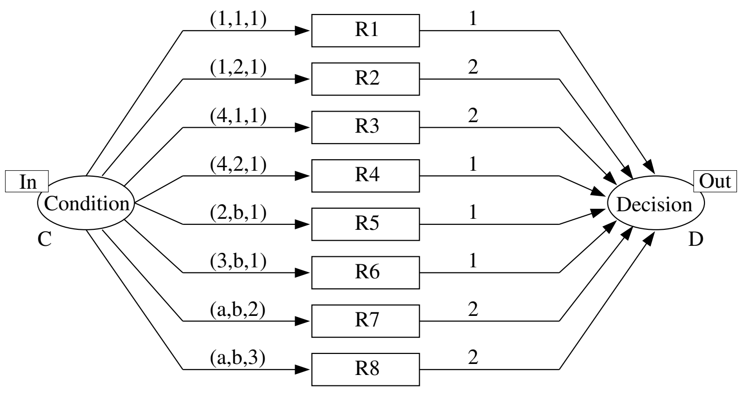


Fig. 2. D-net for the decision table $\mathcal{T}_2$

3 Generalized Decision Tables

Encoding decision tables with the use of atomic values of attributes only is not sufficient for many real applications. If the domains of attributes contain more than several values it may be really hard to cope with the number of decision rules. To handle the problem one can use formulas instead of atomic values of attributes. In such a case, a cell in a decision table will contain 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.

The result of this approach is a decision table with generalised decision rules (or rules' patterns). Each generalised decision rule covers a set of decision rules with atomic values of attributes. Such decision tables will be called *generalized decision tables*. An example of a generalized decision table is presented in Tab. 4. Domains for these attributes are defined as follows:

$V_a = V_d = \{1, 2, 3, 4, 5\}$,
$V_b = V_e = \{\mathit{off}, \mathit{on}\}$, (Boolean values)
$V_c = \{x, y, z\}$.

Table 4. Example of a generalized decision table

	a	b	c	d	e
R1	$a < 4$	$b = on$	$c = x$	$a + 2$	on
R2	a	$b = on$	$c \neq y$	3	off
R3	$a = 5$	b	c	2	$\neg b$
R4	$a > 2$	b	$c \neq x$	$a - 2$	on
R5	$a = 2$	b	$c = x$	4	on

A generalized decision table can be also transformed into the D-net form (see Fig. 3). Formulas that describe values of conditional attributes are usually attached to the guard of the corresponding transition. The algorithm of transformation of a generalized decision table into the D-net form can be found in [11].

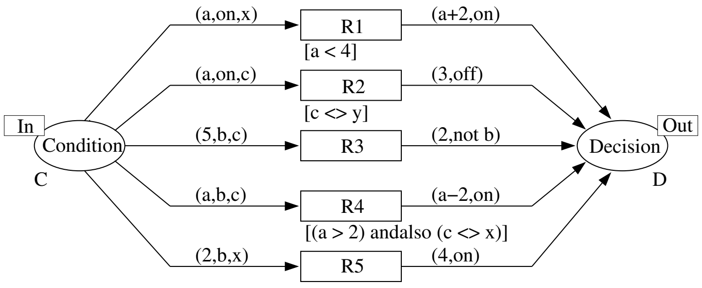


Fig. 3. D-net form of the decision table presented in Tab. 4

4 Hierarchical D-Nets

The main problem in the rule-based system design process is that it is difficult to cope with systems having more than several rules. To simplify the design process a decision table $\mathcal{T} = (R, A, C, D)$ can divided into a set of tables $\mathcal{T}_1 = (R_1, A_1, C_1, D), \cdots, \mathcal{T}_n = (R_n, A_n, C_n, D)$ such that $R_1, \cdots, R_n \subseteq R$, $R_i \cap R_j = \emptyset$ for $i \neq j$, $R = \bigcup_{i=1}^{n} R_i$, $A_1, \cdots, A_n \subseteq A$, $C_1, \cdots, C_n \subseteq C$ and $C_i = C \cap A_i$ for $i = 1, \ldots, n$.

Such decomposition for the rule-based system presented in Tab. 3 is shown in Tab. 5.

Table 5. Table $\mathcal{T}_2$ split into parts

	a	b	c	d
R1	1	1	1	1
R2	1	2	1	2
R3	4	1	1	2
R4	4	2	1	1

	a	c	d
R5	2	1	1
R6	3	1	1

	c	d
R7	2	2
R8	3	2

For each of the three decision tables a D-net (called *sub-D-net*) can be constructed as it was shown in section 2. Next such sub-D-nets are combine into one hierarchical structure. In order to design a hierarchical D-net, a superpage with a substitution transition for each sub-D-net is constructed. The superpage for the consider rule-based system is presented in Fig. 4.

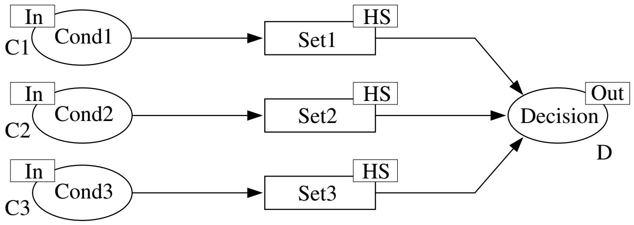


Fig. 4. Superpage for the hierarchical D-net

The superpage together with three sub-D-nets constitute a *hierarchical D-net*. In this case the superpage contains three different conditional places because we have three different sets of conditional attributes. In general the set of rules can be divided into subsets such that some of them share the same set of conditional attributes. In such a case the number of conditional places in the corresponding superpage is less than the number of sub-D-nets. In particular only one conditional place can be used.

On the other hand, the one conditional place is used if we define its colour as a union [5]. For the considered example the conditional place colour should be defined as $C = C1 \cup C2 \cup C3$. However, using of a few conditional places seems to be more practical. The general scheme of a hierarchical D-net is shown in Fig. 5.

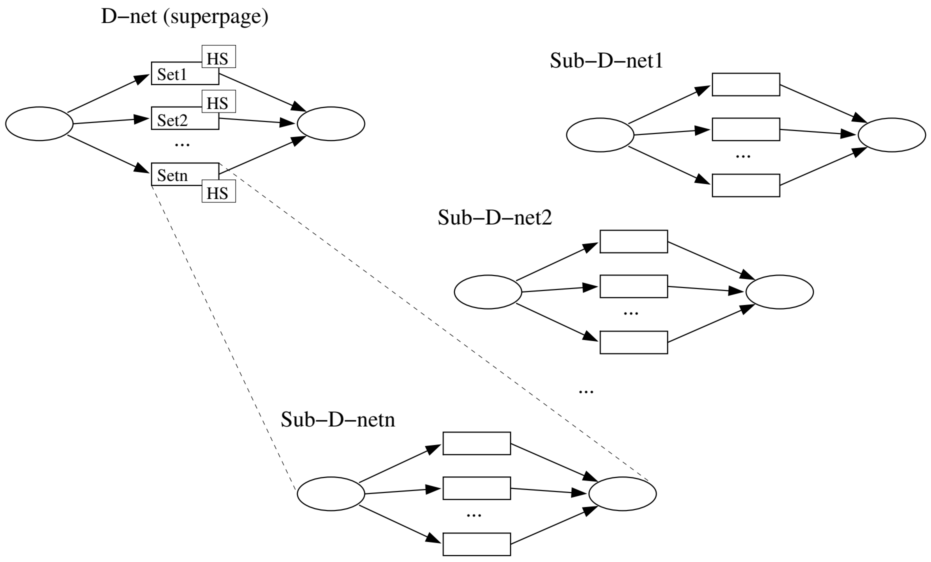


Fig. 5. General structure of a hierarchical D-net

5 Example

Let's consider an example of computer network design, presented in Fig. 6 [7]. It is a typical configuration for many security-aware small office, or company networks. The network is composed of three subnetworks: LAN (local area network), DMZ (the so-called *demilitarized zone*), and INET (Internet connection). The subnetworks are separated by a *firewall* having three network interfaces.

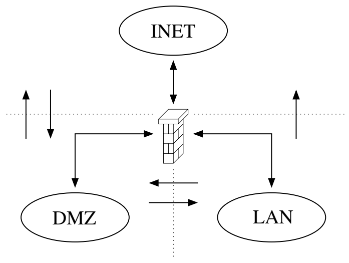


Fig. 6. Network firewall configuration

The firewall controlls the input and output and decides whether the request should be accepted or rejected. Decision table for such a firewall system contains three conditional (service, source address, destination address) and one decision attribute (routing). The attribute $Service$ stands for a type of the net service, attributes $Srcaddr$ and $Destaddr$

are connected with source and destination IP addresses respectively, and the attribute *Routing* stands for the final routing decision. Domains for these attributes are defined as follows:

$D_{Service} = \{ssh, smtp, http, imap\}$,
$D_{Srcaddr} = D_{Destaddr} = \{inet, dmz, lan\}$;
$D_{Routing} = \{accept, reject\}$.

A complete decision table for the firewall system (presented in Tab. 6) contains eleven positive and four negatives rules. The negative rules (without values of decision attributes) are used to state in an explicit way that the particular combinations of input values (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.

D-net generated for the considered decision table is shown in Fig. 7.

Table 6. Decision table for the firewall system

Service	*Srcaddr*	*Destaddr*	*Routing*
$Service = http$	$Srcaddr = inet$	$Destaddr = dmz$	*accept*
$Service = http$	$Srcaddr = inet$	$Destaddr = lan$	*reject*
$Service = http$	$Srcaddr = lan$	$Destaddr$	*accept*
$Service = smtp$	$Srcaddr$	$Destaddr = lan$	*reject*
$Service = smtp$	$Srcaddr$	$Destaddr = dmz$	*accept*
$Service = smtp$	$Srcaddr = lan$	$Destaddr = inet$	*reject*
$Service = imap$	$Srcaddr = lan$	$Destaddr = dmz$	*accept*
$Service = imap$	$Srcaddr \neq lan$	$Destaddr$	*reject*
$Service = ssh$	$Srcaddr = inet$	$Destaddr$	*reject*
$Service = ssh$	$Srcaddr = lan$	$Destaddr$	*accept*
$Service = ssh$	$Srcaddr = dmz$	$Destaddr$	*accept*
$Service = http$	$Srcaddr = dmz$	$Destaddr$	
$Service = http$	$Srcaddr = inet$	$Destaddr = inet$	
$Service = imap$	$Srcaddr = lan$	$Destaddr \neq dmz$	
$Service = smtp$	$Srcaddr \neq lan$	$Destaddr = inet$	

6 Adder Designer

Manual analysis of a decision table can be time-consuming even for very small sets of decision rules. *Adder Designer* supports design and analysis of both simple and generalized decision tables. The tool is equipped with a decision table editor and verification procedures.

Adder Designer is a free software covered by the GNU Library General Public License. It is being implemented in the GNU/Linux environment by the use of the Qt Open Source Edition. The Qt library is freely available for the development of Open Source software for Linux, Unix, Mac OS X and Windows under the GPL license. Code written for either environment compiles and runs with the other ones. *Adder Tools home page*, hosting information about the current status of the project, is located at

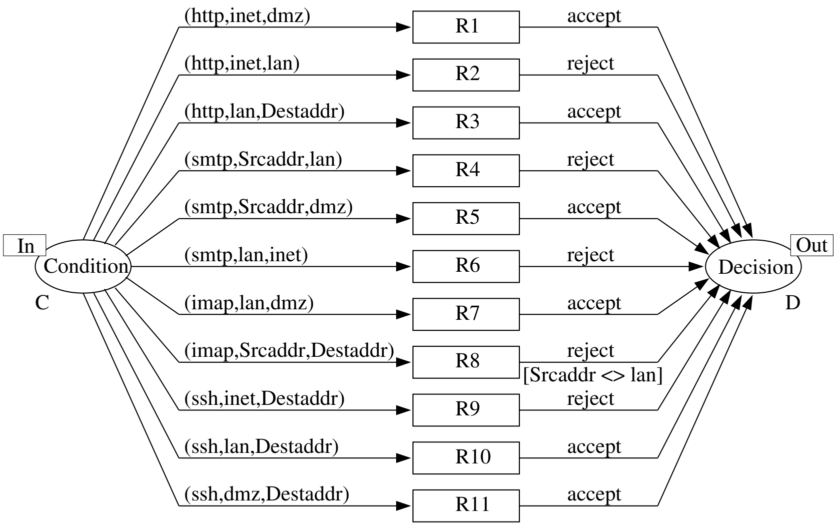
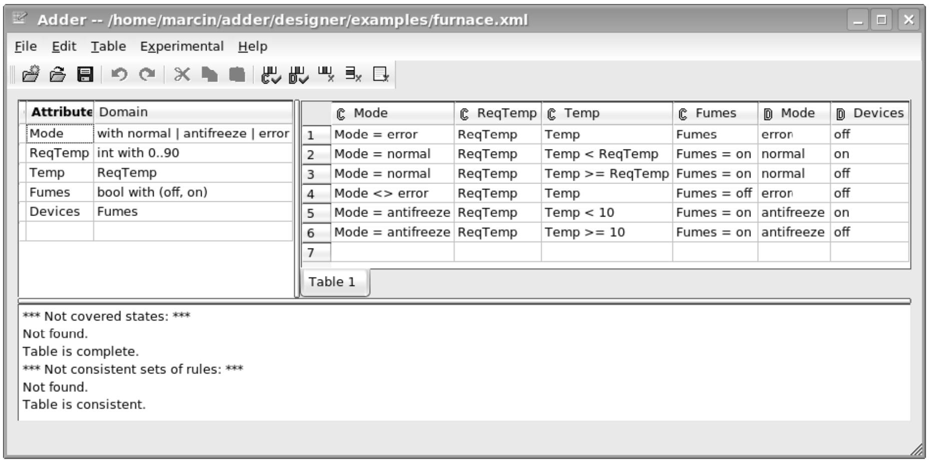


Fig. 7. D-net for the network firewall system


Adder -- /home/marcin/adder/designer/examples/furnace.xml
File Edit Table Experimental Help
Table 1

Attribute	Domain
Mode	with normal | antifreeze | error
ReqTemp	int with 0..90
Temp	ReqTemp
Fumes	bool with (off, on)
Devices	Fumes

	C Mode	C ReqTemp	C Temp	C Fumes	D Mode	D Devices
1	Mode = error	ReqTemp	Temp	Fumes	error	off
2	Mode = normal	ReqTemp	Temp < ReqTemp	Fumes = on	normal	on
3	Mode = normal	ReqTemp	Temp >= ReqTemp	Fumes = on	normal	off
4	Mode <> error	ReqTemp	Temp	Fumes = off	error	off
5	Mode = antifreeze	ReqTemp	Temp < 10	Fumes = on	antifreeze	on
6	Mode = antifreeze	ReqTemp	Temp >= 10	Fumes = on	antifreeze	off
7						

```
*** Not covered states: ***
Not found.
Table is complete.
*** Not consistent sets of rules: ***
Not found.
Table is consistent.
```

Fig. 8. Example of Adder Designer session

http://adder.ia.agh.edu.pl. An example of Adder Designer session is shown in Fig. 8. The figure contains a decision table for a home heating system with a boiler fueled by natural gas and results of completeness and consistency (determinism) analysis.

Using of Adder Designer for design of decision tables consists of a few steps. It is first necessary to define attributes selected to describe important features of the system under consideration. There are possible three types of domains: integer, boolean and enumerated data type. Moreover, a new domain may be defined as an alias for already defined one. Secondly, it is necessary to choose conditional and decision attributes. Each attribute can be used twice. Finally, the set of decision rules should be defined.

The verification stage is included into design process. At any time, during the design stage, users can check whether a decision table is complete, consistent (deterministic) or it contains some dependent rules. Moreover, the tool enables users to *pack* a simple decision table to a generalized one and vice versa.

The tool and the presented approach have been successfully used for developing a few practical examples of rule-based systems, e.g. for a railway traffic management system (22 attributes, 123 decision rules).

7 Summary

Methods of transformation of decision tables into a coloured Petri net form called D-net were presented in the paper. The presented approach can be used to transform into D-nets both simple and generalized decision tables. Moreover, it is also possible to construct hierarchical D-nets that can be treated as a structural form of presentation of rule based systems with many decision rules. The presented approach is supported with computer tools for design and verification of decision tables.

References

1. Adamski, M., Karatkevich, A., Węgrzyn, M. (eds.): Design of Embedded Control Systems. Springer Science+Business Media. Springer, Heidelberg (2005)
2. Bazan, J., Nguyen, S., Nguyen, T., Skowron, A., Stepaniuk, J.: Decision rules synthesis for object classification. In: Orłowska, E. (ed.) Incomplete Information: Rough Set Analysis, pp. 23–57. Physica-Verlag, Heidelberg (1998)
3. Cheng, A.M.K.: Real-time Systems. Scheduling, Analysis, and Verification. Wiley Interscience, New Jersey (2002)
4. Fryc, B., Pancerz, K., Suraj, Z.: Approximate Petri nets for rule-based decision making. In: Tsumoto, S., Słowiński, R., Komorowski, J., Grzymała-Busse, J.W. (eds.) Rough Sets and Current Trends in Computing. LNCS (LNAI), vol. 3066, pp. 733–742. Springer, Heidelberg (2004)
5. Jensen, K.: Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use, vol. 1–3. Springer, Heidelberg (1992-1997)
6. Ligęza, A.: Logical foundations of rule-based systems. Studies in Computational Intelligence, vol. 11. Springer, Heidelberg (2006)
7. Nalepa, G.J., Ligęza, A.: Designing reliable web security systems using rule-based systems approach. In: Menasalvas, E., Segovia, J., Szczepaniak, P.S. (eds.) Advances in Web Intelligence. LNCS (LNAI), vol. 2663, pp. 124–133. Springer, Heidelberg (2003)
8. Pawlak, Z.: Rough Sets. Theoretical Aspects of Reasoning about Data. Kluwer Academic Publishers, Dordrecht (1991)
9. Sommerville, I.: Software Engineering. Pearson Education Limited, London (2004)
10. Szpyrka, M., Szmuc, T.: Integrated approach to modelling and analysis using RTCP-nets. IFIP International Federation for Information Processing 227, 115–120 (2006)
11. Szpyrka, M., Szmuc, T.: D-nets – Petri net form of rule-based systems. Foundations of Computing and Decision Sciences 31, 157–167 (2006)