# Creation and Static Verification of Extended Automata Knowledge Base

**Germanas Budnikas, Tadeuš Lozovski and Miroslav Šeibak**

Faculty of Economics and Informatics in Vilnius, University of Bialystok, Lithuania

**Summary**
The paper presents a technique for creation and static verification of extended automata knowledge base that is expressed in a form of productions and predicates. This knowledge representation is natural for a human. Such knowledge base is understandable for a domain expert too, while specification language might be complex for reading. The constructed knowledge base is verified to ensure satisfaction of its static properties — absence of redundancy, ambivalence and deficiency. The static verification is performed in these steps: productions and predicates of the knowledge base are transformed to lexicographically ordered decision tables; then the decision tables are checked for presence of anomalies that are counter-examples of the general static properties the knowledge base should satisfy. Static verification is performed in Prologa system. Statically verified extended automata knowledge base is intended for development of extended automata based specifications. A case study of Internet Cache Protocol illustrates suggested in the paper technique.

***Key words:***
*Extended automata, knowledge base, static verification, decision table.*

## 1. Introduction

Formal methods and specification languages are widely used for design of distributed systems. The most popular formal specification languages being used for description of distributed systems are Promela, SDL, Estelle [1]. Some analysed applications might be complex and its analysis can be difficult. Participation of a domain expert in the development and analysis of a specified application might be complicated due to unfamiliarity to the used specification language.

A knowledge based approach has several advantages in that problem. First, knowledge representation by production rules when situations are represented by IF-THEN rules is very usual and easy to understand. Knowledge bases can be enhanced with a formal model in order to make easier its construction as well as provide its correctness analysis. Such knowledge bases can be later transformed to formal specifications in order to perform more comprehensive analysis as it was done in [2]. In this paper an extended automata model [3] is used as a knowledge base formal model, which is a background of the most popular specification languages of distributed systems. Correctness of the knowledge base is ensured through performing its static verification. A static verification as an analysis step has such an advantage that it does not require execution of the constructed problem description, i.e. a knowledge base. Moreover, the analysis of description fragments becomes possible.

Extended automata knowledge base $KB_{eA}$ is created using the knowledge acquisition technique that was adapted for the creation of the specific knowledge base—the one, which is intended for mapping to extended automata based formal specification (a mapping is not covered in this paper). Knowledge about a problem domain is represented in a knowledge base in the context of the extended automata model. As a tabular representation is very close to production rule representation [4] and decision table verification method is computerised in software system Prologa, $KB_{eA}$ production rules are transformed to Prologa decision tables to perform static verification automatically. The Prologa system is an interactive design tool for computer-supported construction and manipulation of decision tables. The system offers design techniques and additional features to enhance the construction and validation of decision table [5]. The verification in Prologa is implemented using the tabular verification method [6] that belongs to a group of static verification methods. The transformation to Prologa decision tables is specific with respect to extended automata model. Production rules are transformed to decision tables of certain groups thus enabling to fully exploit advantages of tabular representation to perform static verification. Further, a verified $KB_{eA}$ can be used for creation of extended automata model based specifications, although this last stage is not covered in the paper.

A distinctive feature of the approach is the fact that verification task is performed before the creation of extended automata model based specification.

The proposed technique is similar to the one proposed by Fuchs and Schwitter [7] in such a way that they also offer transformation of problem domain description to representation structures and then to an executable language. However, the proposed technique checks general properties during verification while Fuchs and Schwitter technique checks specific invariant properties. The proposed and Mi & Scacchi [8] approaches are comparable in a viewpoint they both offer checking static properties. General structures of predicates and rules of

Manuscript received December 5, 2012
Manuscript revised December 20, 2012

knowledge base are defined in both approaches too. The approach proposed in the paper is similar to those [9, 10] since they all use decision tables for representation of state-based systems. Mappings between productions and decision tables in order to elaborate advantages of tabular representation are used in the suggested in the paper and Chambers and Parkinson [11], approaches. The suggested approach as well as many others, examples of which are [12, 13], exploit advantages of tabular representation in order to perform verification by transforming certain representation to decision tables. When performing static verification, results of Vanthienen *et al* [5, 6] are being used.

An applicability of the proposed technique is defined by the applicability of the extended automata model and its specifications. They are mostly used for formal specification and analysis of distributed systems. The technique was investigated while creating and analysing static properties of a real sized application – Internet Cache Protocol [14, 15]. The scalability of the proposed technique is limited by software tool that is used for static analysis. The limitation requirement for this tool is defined in [5].

The paper is structured as follows. Section 2 describes construction of extended automata knowledge base. Section 3 presents static verification technique for analysis of the extended automata knowledge base. Section 4 illustrates the proposed approach by analysing Internet Cache Protocol model. Conclusions sum up the proposed approach.

## 2. Creation of Extended Automata Knowledge Base

Extended automata knowledge base uses production rule representation. This representation is wide-spread due to its easy acceptance by non-experienced users. Since most of the common verification and validation problems in rule-based systems can be solved using decision tables and the tabular verification method is computerised in Prologa system, in the suggested approach static verification will be performed using this method. Thus, in order to perform the static verification of the knowledge base, its production rules have to be transformed to Prologa decision tables. Additionally, as stated in [4], a decision table is equivalent to a set of production rules, and their transformation to the tables can be performed without too much effort.

Because the knowledge base is intended to use for development of extended automata based specification, it has to contain knowledge about the extended automata model. To acquire this knowledge, the knowledge acquisition technique [16] was applied. The following knowledge about the extended automata model is used: input and output signals, states, their parameters, timers, conditions for state changes.

Next, examples of predicates and productions of the extended automata knowledge base are given.

$$(\text{Input}\, a_i\ x_i^j\ x_j^1 \ldots x_j^{c_{ic}^i}),$$

where $a_i$ – symbolic name of the *i*th automaton; $x_j^1,\ldots,x_j^{c_{ic}^j}$ – components of input signal $x_i^j$.

$$(\text{State}\ a_i\ w_i^1 \ldots w_i^{c_{cc}^i}\ d_i^1 \ldots d_i^{c_{dc}^i}),$$

where $w_i^1,...,w_i^{c_{cc}^i}$ are timers of operations performed; $d_i^1,...,d_i^{c_{dc}^i}$ additional parameters.

The production rule that describes state change and signal output after arrival of input signal has the following general form:

IF (Input $a_i\ x_i^j\ x_j^1 \ldots x_j^{c_{ic}^i}$)
AND (State $a_i\ w_i^1 \ldots w_i^{c_{cc}^i}\ d_i^1 \ldots d_i^{c_{dc}^i}$)
AND (Aux $a_i\ w_i^1 \ldots w_i^{c_{cc}^i}\ d_i^1 \ldots d_i^{c_{dc}^i}$)
THEN (State $a_i\ w_i^1* \ldots w_i^{c_{cc}^i}*\ d_i^1* \ldots d_i^{c_{dc}^i}*$)
AND (Output $a_i\ x_i^j\ x_j^1 \ldots x_j^{c_{oc}^j}$)

where (Aux $a_i\ w_i^1 \ldots w_i^{c_{cc}^i}\ d_i^1 \ldots d_i^{c_{dc}^i}$) describes auxiliary conditions $LC_1,\ldots,LC_{p_m}$ to be satisfied for state change:

(Aux $a_i\ w_i^1 \ldots w_i^{c_{cc}^i}\ d_i^1 \ldots d_i^{c_{dc}^i}$) ≡ $LC_1$ AND|OR…AND|OR $LC_{p_m}$, where $i=\overline{1,n}$, $p_m$ – the number of additional logical conditions for the *m*-th production rule that describes state change.

## 3. Static Verification of Extended Automata Knowledge Base

Most of the verification problems in rule-based systems like redundant, ambivalent, categorised, cyclic or missing rules, redundant conditions, and unused action parts may be resolved using decision tables [5].

Suggested technique uses decision table representation for static verification of extended automata knowledge base since decision tables clearly demonstrate incompleteness and inconsistency of knowledge [13] and software tool Prologa assists verification process. Anomalies, which are known as counter-examples of general properties, in decision tables have direct correspondence to anomalies in production rules that are being a part of the created knowledge base.

### 3.1 Static Anomalies in Rule Bases and Decision Tables

A decision table formally is described as follows [6]:

$SE = \{SE_1, SE_2, \ldots, SE_m\}$ is a set of condition subjects, $m$ - number of condition subjects;

$SD = \{SD_1, SD_2, \ldots, SD_m\}$ is a set of condition domains. $SD_i$ - domain of the $i$-th condition, i.e. the set of all possible values of condition subject $SE_i$;

$SB = \{SB_1, SB_2, \ldots, SB_m\}$ is a set of condition states. $SB_i = \{B_{i,1}, B_{i,2}, \ldots, B_{i,q_i}\}$ is an ordered set of $q_i$ condition states $B_{i,k}$. Each *condition state* is a logical expression concerning elements of $SD_i$, which determines a subset of $SD_i$ such that the set of all these subsets constitutes a partition of $SD_i$;

$V = \{V_1, V_2, \ldots, V_n\}$ is a set of action subjects;

$VB = \{VB_1, VB_2, \ldots, VB_n\}$ is a set of action value sets. $VB_j = \{\times, -, .\}$ is a set of possible values of action $V_j$, where "×" denotes "execute the action", "–" denotes "do not execute the action", "." – "action execution is not defined".

The decision table DT is a function from the Cartesian product of the condition states to the Cartesian product of the action values, by which every condition combination is mapped onto action configuration: $DT: SB_1 \times SB_2 \times \ldots \times SB_m \rightarrow VB_1 \times VB_2 \times \ldots \times VB_n$.

The most important criterion when distinguishing tables, is the question whether all columns are mutually exclusive (*single hit* versus *multiply hit*). In a single hit table, in contrast to multiply hit table, each possible combination of condition can be found in exactly one and only one column. Single hit decision tables are also known as *lexicographically ordered DTs*.

**Definition** [17]: Given an order $\prec_i$ of each condition state element from the set $SB_i$ $\left(i = \overline{1,m}\right)$ between the elements in the $i$-th row $B_{i,1} \prec_i B_{i,2} \prec_i \ldots \prec_i B_{i,q_i}$, let parts of condition state elements $c_{i,j}$ (in the columns $j$ and $k$) be ordered as follows $\left(c_{1,j}, c_{2,j}, \ldots, c_{m,j}\right) \prec \left(c_{1,k}, c_{2,k}, \ldots, c_{m,k}\right)$, and there is a row index $i$ such that $c_{i,j} \prec_i c_{i,k}$ and for $h = \overline{1,(i-1)}$: $c_{h,j} = c_{h,k}$. If the columns in the DT are arranged accordingly, we say that the DT is *lexicographically ordered.*

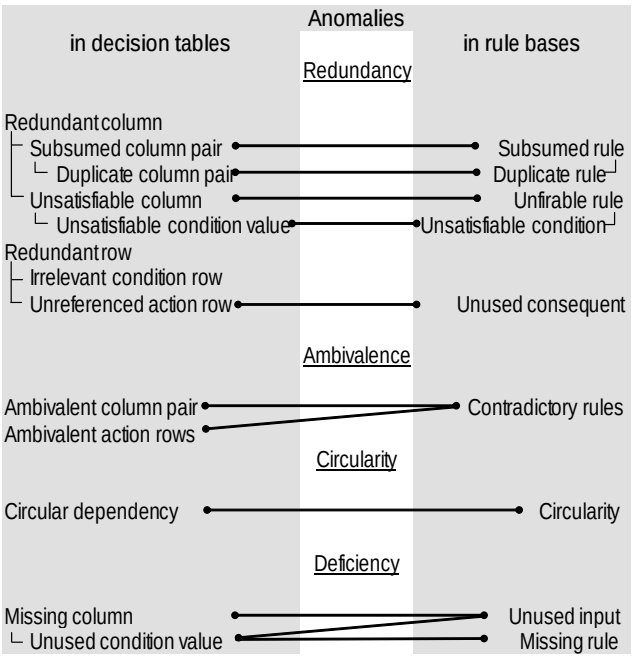


Fig. 1 Relation between anomalies in decision tables and anomalies in rule bases. Source: made by authors using [20] and [21].

The static anomalies are characteristics of a KBS that can be evaluated without its execution. Such an evaluation is often referred to as static or structural verification. During static verification, a KB is checked for anomalies [18]. Preece and Shinghal [21] present a classification of the anomalies that may be present in rule-based systems. It is necessary to note the difference between an anomaly and error. The anomaly indicates the existence of a possible error. For contrast, the dynamic properties are those characteristics of a rule-based system that can be evaluated only by examining how the system operates at a run time. The most common techniques of validation and verification that have been developed for use on KBS are identified in [19].

The following anomalies are distinguished: *intra*-tabular, which occur in a single DT, and *inter*-tabular anomalies, that originate from interactions between several DTs [20]. A relation between *intra*-tabular anomalies and anomalies in rule bases is depicted in Figure 1. Using practical experiments it was noticed, that in decision tables made of extended automata knowledge base only intra-tabular anomalies are possible. Anomalies of only that type are being considered further.

### 3.2 Transformation of Extended Automata Knowledge Base Productions to Lexicographically Ordered Decision Tables

A static verification of the created knowledge base is performed using Prologa system in the proposed technique. The system uses lexicographically ordered decision table representation. Next, the suggested steps of transformation of knowledge base productions to lexicographically ordered decision tables are presented.

1. Productions that describe system functioning after event occurrence (in case of input signal arrival or timer expiration) are represented by single tables, which are referred to as *event master table*.
2. Productions that describe an initial state, conditions of event occurrences and signal interchange between automata are represented by single tables too and are referred to as *auxiliary tables*;
3. Predicates of antecedent part of a rule are written in a form of condition subjects and condition states in a decision table;
4. Condition subject states are determined with respect to a problem description and the considered production rule;
5. Predicates of consequent part of a rule are written in a form of actions in a decision table. Predicate State in the rules that describe an initial state and timer expirations as well as predicates Output, Input, Timer in all type rules are written along with additionally added ":=True" expression. In this way, a value that is defined in an auxiliary table is passed to a master one where this value is used.
6. If consequent parts of productions contain a pair of assignments $d_i^j = d_i^j + 1$ and $d_i^j = d_i^j - 1$, then before transforming $d_i^j = d_i^j - 1$ assignment to decision table action it is written in the following form: not ($d_i^j = d_i^j + 1$), where $d_i^j$ denotes parameter or variable.

Resulted decision tables are inspected in Prologa system using decision table verification method that analyses its different parts and comparing them. A verified $KB_{eA}$ can be further transformed to extended automata based specifications, e.g. Promela, SDL as it contains components of its background model.

## 4. Case Study - Creation and Analysis of ICP $KB_{eA}$ and Its Static Verification

A description of Internet Cache Protocol (ICP) is presented in RFC 2186 [14] and RFC 2187 [15] documents. The description consists of up to 30 pages and was used for creation of $KB_{eA}$ of a protocol model. Below a brief description is given.

ICP is a Web page caching protocol used to interchange information about an existence of Web pages between sibling caches. Wen caches communicate between themselves by sending queries and replies in order to gather information about the most appropriate location from which to retrieve an object. The protocol is being used in many software products [22].

A simplified view of the environment where ICP is used is given in Figure 2. This architecture was used in the paper.

Internet network contains servers – caches that may contain Web pages, which in their turn were requested by local network computers (or other caches) – clients. These servers are configured as a Local Cache or Parent Cache depending on their physical allocation in a hierarchy of computer networks. Local caches contain configurations how they are connected to other local or parent caches.

To retrieve a Web page, a Client internet browser sends a HTTP request using *Hyper Text Transfer Protocol* to a Local Cache. The last, after message receiving, performs such activities in the following order: (i) check a syntactical correctness of the requested URL address; (ii) check whether an address of a retrieved Web page is closest than a Sibling Cache; (iii) check whether a client retrieves a personal information; (iv) checks whether a retrieved Web page belongs to a domain covered by a Local Cache.

After performing these operations, a Local Cache searches for a requested Web page in its repository. If failed, it sends ICP requests to Sibling Caches and Parent Cache as well as packets in order to evaluate a response time from a Remote Server where the requested Web page is located (*ping* command). Additionally, Local Cache sets a timer to define a time to wait for the response after which expiration; the ICP protocol chooses a Remote Server as a source for Web page retrieval. Sibling Caches as well as Parent Cache after reception of ICP request performs such operations:

- check a syntactical correctness of the requested URL address;
- check whether a request sender is authorised to access Cache resources. If the access is restricted – a corresponding *ICP* message is returned to a requester and as well as a counter of access denied messages is incremented. If the counter exceeds the defined threshold, requests from such senders are ignored;
- calculation of the Web page size found in Cache repository. Depending on ICP query parameters a copy of the requested Web page can be sent out together with a positive reply if it does not exceeds an allowable message limit.

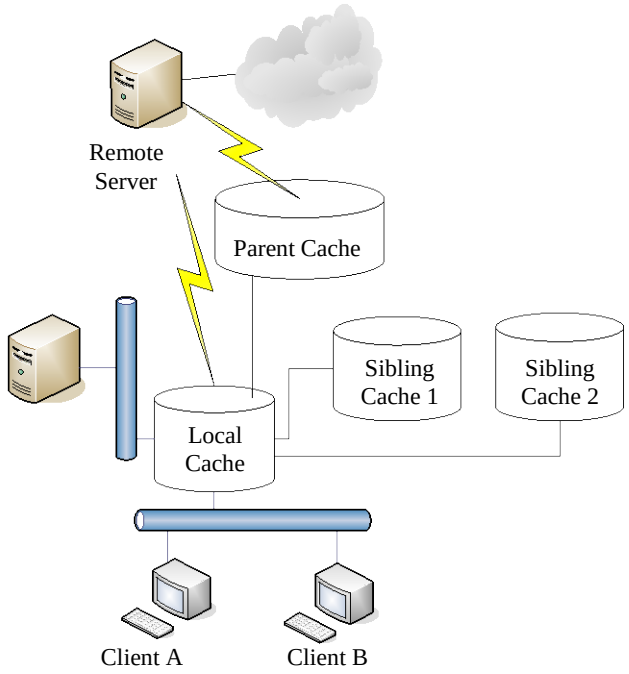


Fig. 2 Internet Cache protocol architecture analysed in the paper. Source: made by authors using [15].

Local Cache selects either a Sibling Cache or a Remote Server depending on their response time. A fragment of a created $KB_{eA}$ productions and decision tables used for static verification are given below.

| **A part of conceptual description** |
|---|
| **$KB_{eA}$ predicates and productions** |

*Local Cache LC inputs and input parameters*

| |
|---|
| Arrived *HTTP* request from *ClientA*, where *msg* – a type of message, *mthd* – a method, *url* – address of a retrieved Web page, *prgm* – a sign, *sender* – sender IP address. |
| Input(*LC,MsgFromClientA,msg,mthd,url,prgm,sender*)<br>*Local Cache is characterised by up to 40 parameters. For the briefness, the predicate State_LC will be used instead of predicate State (LC, …) naming all 40 parameters.* |

*State change after arrival of input message*

| |
|---|
| After arrival of a message from a Client which type is *Request* to a non-occupied Local Cache, the Cache saves parameters of the arrived message and starts syntax checking of the requested URL address. |
| **LC_R411a:**<br>IF Input (*LC,MsgFromClientA,msg, mthd,url, prgm, sender*)<br>and State_LC<br>and *msg* = *Request*<br>and *busy* = False<br><br>THEN *requester** = *Sender*<br>and *pragma** = *prgm*<br>and *method** = *mthd*<br>and *url_parsing** = Active<br>and *payload** = *url*<br>and *busy** = True and State_LC |

*Change of state after timer expiration*

| |
|---|
| If search in a Cache has ended and both a retrieved Web page has been found as well as Cache is not being updated either a length of Cache queue is of a certain size then a retrieval of a Web page from a Remote Server and its transmission to a Client is starting. |
| **LC_R561:**<br>IF Timer(*LC,Search_in_Cache*)<br>and State_LC<br>and ((*search_res* =True and<br>*upd_timer* =Passive)<br>or (*queue_len*>=100))<br>THEN *fetch_from** = *RemoteServer*<br>and *fetch_url** = Active<br>and State_LC |
| If search in a Cache has ended and both a retrieved Web page has been found as well as Cache is being updated and Web page requester is a Client (Client A) then a retrieval of a Web page from a Local Cache and its transmission to a Client is starting. |
| **LC_R563a:**<br>IF Timer(*LC,Search_in_Cache*)<br>and State_LC<br>and *search_res* = True<br>and *upd_timer* = Active<br>and *requester* = *ClientA*<br>THEN *fetch_from** = *LocalCache*<br>and *fetch_url** = Active<br>and State_LC |
| If search in a Cache has ended and both a retrieved Web page has been found as well as Cache is being updated and Web page requester is a Client (Client B) then a retrieval of a Web page from a Local Cache and its transmission to a Client is starting. |
| **LC_R563b:**<br>IF Timer(*LC,Search_in_Cache*)<br>and State_LC<br>and *search_res* = True<br>and *upd_timer* = Active<br>and *requester* = *ClientB*<br>THEN *fetch_from** = *LocalCache*<br>and *fetch_url** = Active<br>and State_LC |

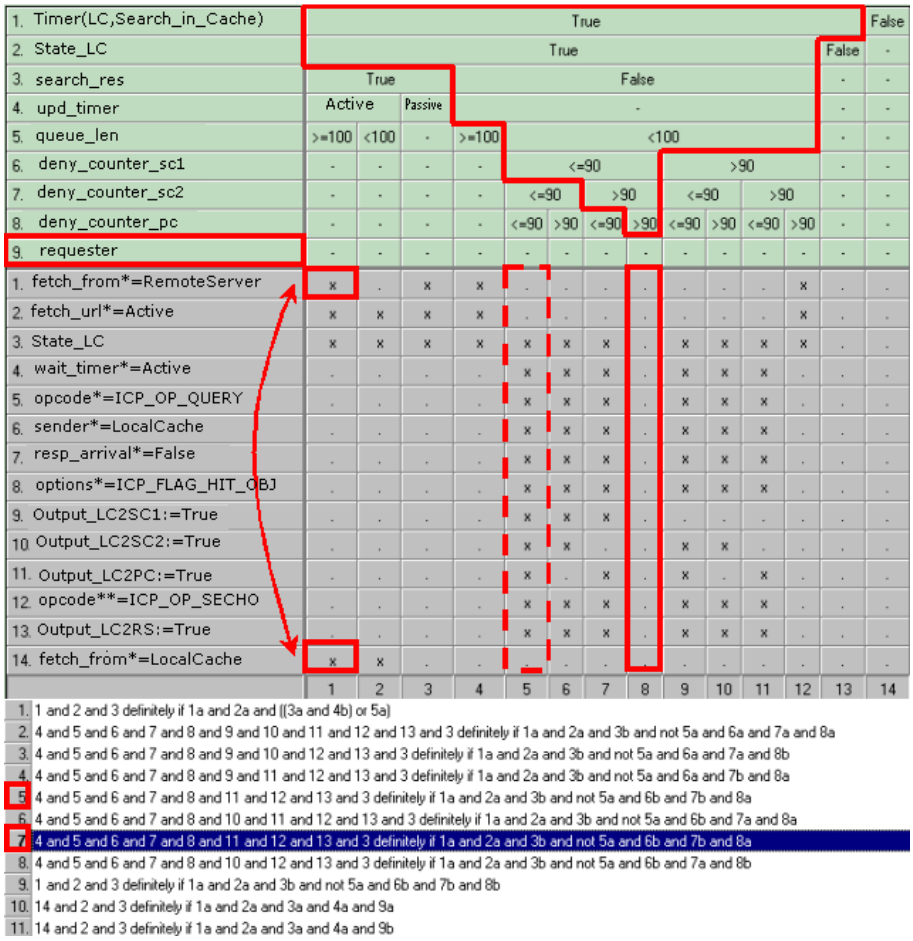


Fig. 3. Prologa Decision Table Made of $KB_{eA}$ Productions. Decision Table Describes Activities After Search in Local Cache as Ended.

In order to perform static verification, ICP protocol model $KB_{eA}$ productions and predicates was transformed to Prologa system decision tables. A decision table that corresponds to a set of productions describing *Search_in_Cache*, is given below in Figure 3.

The table represents all productions that describe activities after *Search_in_Cache* has ended while only a part of them is presented in the paper. Some anomalies (or counter-examples of general static properties) have been detected during static verification; these are marked directly on the Figure 3 and explained next.

*Ambivalence – ambivalent action rows anomaly.* Production LC_R561 is described by 1,3,4 table columns; and productions LC_R563a and LC_R563b are described by 1 and 2 columns. Hence, condition parts of these rules are overlapping. 1st action row of LC_R561 rule *fetch_from** = *RemoteServer* as well as 14th action row of the rules LC_R563a and LC_R563b *fetch_from** = *LocalCache* are ambivalent since different fetch-from objects are specified at the overlapping conditions (see table first column);

This anomaly is fixed by inserting additional condition *queue_len < 100 to the productions* LC_R563a and LC_R563b.

*Redundancy – irrelevant condition row anomaly.* Productions LC_R563a and LC_R563b are described by the same 1st table column and its 9th condition subject *requester* is an irrelevant condition row– in spite of possible values either *ClientA* or *ClientB* of this condition item a set of performed actions is the same.

This anomaly is fixed by removing conditions related to requester from aforementioned productions. Thus, a single rule is obtained.

*Redundancy – duplicated column pair anomaly.* 5th decision table column corresponds to two productions, which are described by 5th and 7th decision rules (marked in a bottom part of the Figure 3).

This anomaly is fixed by removing from $KB_{eA}$ a duplicated production rule.

*Deficiency – unknown action state anomaly.* 8th column of the decision table does not specify actions to be performed in case of conditions, which are marked by a contour in the Figure 3.

The anomaly is fixed by supplementing $KB_{eA}$ with the corresponding rule that specifies actions with respect to the conditions marked out in the Figure 3.
After fixing these anomalies, the $KB_{eA}$ of the ICP protocol model satisfies the general static properties – absence of redundancy, ambivalence and deficiency.

## 5. Conclusions

Use of the proposed technique gives the following advantages:

- *creation* of a model description is understandable by a domain expert due to applied knowledge representation;
- possibility to analyse completeness and consistency of the description by performing its static verification;
- due to background model of the knowledge base - an extended automaton, - it can be used further for succeeding more comprehensive analysis of an analysed problem domain.

## References


[1] Rychwalski, P., Wytrębowicz, J. Unix streams generation from a formal specification, 1-14 // Proc. of the 23rd IFIP WG 6.1 International Conference "Formal Techniques for Networked and Distributed Systems - FORTE2003", H.Konig, M.Heiner, A.Wolisz (Eds.) Springer (2003)

[2] Budnikas, G. Application of knowledge-based techniques for creation of ESTELLE/AG specifications. In proc. of the International Conference Modelling and Simulation of Business Systems, Eds. H. Pranevicius, E. Zavadskas, B. Rapp. Kaunas, ,,Technologija" , 2003: pp. 172-176. ISBN 9955-09-420-6.

[3] Sharafi, M., Aliee, F.S., Movaghar, A. (2007). A Review on Specifying Software Architectures Using Extended Automata-Based Models. In International Symposium on Fundamentals of Software Engineering. Lecture Notes in Computer Science. Springer Berlin / Heidelberg. 423-431; ISBN: 978-3-540-75697-2

[4] Goedertier, S., Vanthienen, J. (2005) Rule-based business process modeling and execution. In: Proceedings of the IEEE EDOC Workshop on Vocabularies Ontologies and Rules for The Enterprise. CTIT Workshop Proceeding Series (ISSN 0929-0672)

[5] Vanthienen, J. (2000). Prologa v.5 User's manual, Katholieke Universiteit Leuven. 121 pp.

[6] Vanthienen, J., C. Mues, and G. Wets (1997). Inter-tabular Verification in an Interactive Environment. In J. Vanthienen, F.v. Harmelen (Eds.) Proc. 4th European Symposium on the Validation and Verification of Knowledge Based Systems. Katholieke Universiteit Leuven, pp. 155-165.

[7] Fuchs, N.E., R. Schwitter (1996). Attempto Controlled English. In Proc. of the CLAW 96, The First International Workshop on Controlled Language Applications, Katholieke Universitet, Leuven, 26-27 March, 1996.

[8] Mi, P., W. Scacchi (1995). A Knowledge-based Environment for Modeling and Simulating Software Engineering Processes. IEEE Trans. on Knowledge and Data Engineering, 2(3), 283-294.

[9] Arentze, T.A., Borgers, A.W.J., Timmermans, H.J.F. The integration of expert knowledge in decision support systems for facility location planning", Computers Environment and Urban Systems, 19 (4), July-August, 1995: pp.227-247.

[10] Haughton H.. "Formal development of communications software-a research project", Mathematical Structures for Software Engineering, based on the Proceedings of a Conference, Clarendon Press, Oxford, UK (1991) p. 253-274.

[11] Chambers, T.L., Parkinson, A.R. Knowledge representation and conversion for hybrid expert systems", Journal of Mechanical Design, 120 (3), September, 1998: pp.: 468-474.

[12] Colomb R., Chung C., "Very fast decision table execution of propositional expert systems", Eighth National Conference on Artificial Intelligence, MIT Press, Cambridge, MA, USA. 1990 vol.2, pp. 671-676.

[13] Degelder-J; Steenhuis-M, "A knowledge-based system approach for code-checking of steel structures according to Eurocode 3", Computers-and-Structures. Jun 1998; 67 (5) : 347-355

[14] Wessels, D., K. Claffy. RFC 2186 - Internet Cache Protocol (ICP), version 2. National Laboratory for Applied, Network Research/UCSD. September 1997a.

[15] Wessels, D., K. Claffy. RFC 2187 - Application of Internet Cache Protocol (ICP), version 2. National Laboratory for Applied, Network Research/UCSD. September 1997 b.

[16] Russel, S., P. Norvig (2009). Artificial Intelligence—a Modern Approach. Prentice Hall, Inc.

[17] Mues C. On the Use of Decision Tables and Diagrams in Knowledge Modeling and Verification, PhD dissertation, K.U. Leuven, 2002: 223 p.

[18] Meseguer, P., A. Preece (1996) Assessing the Role of Formal Specifications in Verification and Validation of Knowledge Based Systems. In Proc. 3rd IFIP International Conference on Achieving Quality in Software, Chapman and Hall. pp. 317-328.

[19] Preece, A. (2001). Evaluating Verification and Validation Methods in Knowledge Engineering. In R. Roy (Ed.), Micro-Level Knowledge Management, Morgan-Kaufman, pp. 123-145.

[20] Vanthienen, J., C. Mues, A. Aerts. An illustration of verification and validation in the modelling phase of KBS development. Data & Knowledge Engineering 27, 1998: pp. 337-352.

[21] Preece, A., R. Shinghal. (1994). Foundation and Application of Knowledge Base Verification. International Journal of Intelligent Systems, 9, 683-702.

[22] Examples of ICP applications:
http://www.squid-cache.org/ ;
http://www.netapp.com/products/netcache/ ;
http://www.microsoft.com/isaserver/ ;
http://www.cisco.com/go/cache/ ;
http://www.novell.com/products/volera/ ;
http://www.bluecoat. com/ ; http://www.imimic.com/

**Germanas Budnikas** received the B.S., M.S. and Ph.D. degrees in Informatics from Kaunas University of Technology in 1994, 1996 and 2004 respectively. His research interests include artificial intelligence, formal specifications, and their static and dynamic analyses.

**Tadeuš Lozovski** received the M.E. degrees, from Kaunas Technical University in 1967. He received the Dr. Eng. degree from Kaunas Technical University in 1985. After working as a research assistant and assistant professor (from 1991) in the Dept of Solid State Electronics, Vilnius University, and an associate professor (from 1999) he received the Dr. Sc. degree from Wroclaw Technical University (Poland) in 2001, he has been a professor at Bialystok University since 2007. His research interest includes nondestructive method research surface potential thin semiconductor and dielectric layer. He is a member of STIPL and SNPL Lithuania.

**Miroslav Šeibak** received the M.Sc. degree from Vilnius University in 1986. He received the Ph.D. degree from Vilnius University in 1993. After working as a research assistant in the Department of Differential Equations and Numerical Analysis at Faculty of Mathematics and Informatics, Vilnius University, he has been an assistant professor at the Faculty of Economics and Informatics of the University of Bialystok in Vilnius since 2007. His research interest includes numerical analysis.