Grzegorz J. Nalepa*

A NEW APPROACH TO THE RULE-BASED SYSTEMS DESIGN AND IMPLEMENTATION PROCESS

The paper discusses selected problems encountered in practical rule-based systems (RBS) design and implementation. To solve them XTT, a new visual knowledge representation is introduced. Then a complete, integrated RBS design, implementation and analysis methodology is presented. This methodology is supported by a visual CASE tool called Mirella. *The main goal is to move the design procedure to a more abstract, logical level, where knowledge specification is based on use of abstract rule representation. The design specification is automatically translated into* Prolog *code, so the designer can focus on logical specification of safety and reliability. On the other hand, system formal aspects are automatically verified on-line during the design, so that its verifiable characteristics are preserved.*

Keywords: *rule-based systems, visual design, implementation, formal analysis, CASE tools*

NOWE PODEJŚCIE DO PROCESU PROJEKTOWANIA I IMPLEMENTACJI SYSTEMÓW REGUŁOWYCH

W artykule omówione zostały wybrane problemy związane z projektowaniem i implementacją systemów regułowych. W celu rozwiązania najważniejszych z nich przedstawiona została nowa wizualna metoda reprezentacji wiedzy (XTT) oraz zaprezentowano nową metodologię projektowania, implementacji i analizy takich systemów. Podejście to jest wspierane przez narzędzie typu CASE nazwane Mirella. *Pozwala ono na projektowanie systemu na wysokim poziomie abstrakcji, przy równoczesnych zachowaniu jego własności logicznych i formalnych. Na podstawie projektu generowany jest prototypowy model systemu w języku* Prolog, *którego własności formalne mogą być na bieżąco analizowane.*

Słowa kluczowe: *systemy regułowe, wizualne projektowanie, formalna analiza, narzędzia CASE*

1. Introduction

Rule-Based Systems (RBS) constitute a powerful tool for specification of knowledge in design and implementation of knowledge-based systems (KBS) in applied Artificial Intelligence and Knowledge Engineering. They provide also a universal programming

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

paradigm in domains such as system monitoring, intelligent control, decision support, situation classification, system diagnosis and operational knowledge encoding [10, 8].

Moreover, although the rule-based programming paradigm seems relatively conceptually simple, in case of realistic systems it is a hard and tedious task to design and implement a rule-based system that works in a correct way. Problems occurs as the number of rules exceeds even relatively very low quantities. It is hard to keep the rules consistent, to cover all operation variants and to make the system work according to desired algorithm. A well-defined system should be safe, reliable, and efficient. These features are further translated into a set of precisely defined characteristics (e.g. completeness, determinism, correctness) which can be verified in a formal way.

Practical design of non-trivial rule-based systems requires a systematic, structured and consistent approach. Such an approach is usually referred to as a design methodology. The basic elements distinguishing one methodology from the other are the internal design process structure i.e. the way structuring of the design process and the components of the process. The structure can be linear, linear with loops, hierarchical (top-down, bottom-up), etc. The components may be various procedures, techniques, tools and documentation aids to support and facilitate the process of design [10, 8].

The paper outlines some of the main problems encountered in practical RBS design. To solve these problems a new, visual knowledge representation method called eXtended Tabular-Trees is introduced. Then a complete, integrated RBS design methodology is presented. This methodology is supported by a visual CASE tool called Mirella. The main goal of the system is to move the design procedure to a more abstract, logical and graphical level, where knowledge specification is based on use of abstract rule representation. The designed graphical specification is automatically translated into a predefined XML knowledge format, so the designer can focus on logical specification of safety and reliability; simultaneously, practical code can be generated for a wide class of systems. On the other hand, selected formal aspects may be automatically verified on-line during the design using Prolog-based representation, so that it verifiable characteristics are preserved. An example of practical application of Mirella is shown. The paper ends with concluding remarks along with some directions for future work.

2. Crictical Perspective on RBS Design Methods

Although rule-based technologies appear simple and intuitive at the first glimpse, designing a real-scale rule-base is both tedious and difficult task. The main problem is that in systems having more than several rules it becomes difficult to control their properties at the design stage. A well-defined system should be *safe*, *reliable*, and *efficient* and these features are further translated into a set of precisely defined characteristics which can be verified in a formal way. Here one typically considers that such a system must be *complete*, i.e. work in any input situation, *deterministic*, i.e.

its behavior must be predictable, *correct*, i.e. work according to desired specification, and it should be *minimal*, i.e. should incorporate only necessary set of rules [7, 10].

On the base of research and evaluation of multiple rule-based systems design methods [1, 18], supported by development tools [9, 11, 13, 10], a conclusion has been drawn, that existing methods and tools have some serious limitations.

These limitations are located in the following *three areas*:

1) knowledge representation method,
2) framework for analysis and verification of formal properties, and
3) computer tools supporting the integrated design process.

Some most important limitations concerning the knowledge representation methods consist in using system-specific knowledge representation formalisms. This results in restricted application area, scalability problems, and possibly limited CASE tools support. These limitations often restrict real-life applications of existing knowledge representation methods.

With respect to the practical analysis approaches, the main problem is that no formal verification of system properties in early stages of the development cycle is carried out, which leads to so-called late verification problem and usually inefficient development cycle.

The third issue is that design approaches do not offer integrated computer development tools (*CASE*) supporting the rule-based system building process at all stages – from the design to implementation phase. Existing methods support mainly subsequent stages of the conceptual design in case of large systems, while direct technical support of the logical design and during the implementation phase is mostly limited to providing a context-sensitive, syntax checking editors, or simple wizards that support the design process. Most of the available CASE tools have the following limitations: no integrated design and implementation process specialization, and limited analysis facilities.

To overcome limitations outlined above, a new approach to rule-based systems design process, supported by an integrated CASE tool, can be proposed.

3. New Approach to RBS Design and Implementation

The approach to Rule-Based Systems design process proposed in the following sections is based on the idea of integrated design and analysis supported by an intelligent, interactive tool. The main goal is to move the design procedure to a more abstract, logical level, where knowledge specification is based on the use of abstract rule representation. The design specification can be automatically translated into a low-level code, including logical PROLOG and XML, so that the designer can focus on logical specification of safety and reliability.

On the other hand, selected system properties can be automatically analyzed on-line during the design, so that its characteristics are preserved. From a practical

point of view, the design process is performed with a full-screen intelligent, interactive graphical tool [10].

The generated PROLOG code constitutes a *prototype implementation* of the system. Since it is equivalent to the visual design specification it can be considered to represent an *executable specification*.

To overcome limitations discussed in Section 2 the approach offers new solutions in all the three areas mentioned above:

1) A new *knowledge representation* method for visual and logical knowledge specification, called eXtended Tabular Trees (XTT for short), is offered.
2) An integrated rule-based system *design, implementation, and analysis processes* is proposed.
3) A new, visual, XTT-based *CASE tool* supporting this process, and allowing for system prototype generation, is introduced.

This kind of approach allows for creating a framework for hierarchical and modular synthesis and on-line analysis of the knowledge base. In particular the following specific *advantages* of this approach include:

- clear separation of *logical* and *physical* design phases, which makes the design of the knowledge base independent of the target application,
- a hierarchical design methodology, improving possibilities of creating a structured rule base,
- possibility of including explicit inference control information in the rule base,
- enhanced rule base portability, obtained through domain-independent knowledge base specification,
- an easy-to-follow framework consistent with information flow and incorporating partial inference control,
- possibility of incorporating on-line analysis and verification of formal properties.

In next sections these solutions are explained in more detail.

4. Extended Tabular Trees

The main idea behind the new visual knowledge representation language called EXTENDED TABULAR TREES [10, 14] aims at combining some of the existing approaches such as *decision-tables* and *decision-trees* by building a special hierarchy of Object-Attribute-Tables [5, 6].

The language uses some basic concepts which are present in other knowledge representation methods (see [4, 3, 2]), such as *rules*, which give logical representation of the system knowledge base, *decision tables*, which are basic components containing knowledge, and *decision trees*, which serve as a tool for connecting knowledge components.

EXTENDED TABULAR TREES, XTT for short, are a formalism that allows for a *visual knowledge representation and design*.

The language has some unique features such as:

- simplicity and transparency,
- an intuitive hierarchical, tree-like knowledge representation,
- possibility of including explicit inference control information,
- highly efficient way of visualization with high data density,
- power of the decision table representation,
- flexibility with respect to knowledge manipulation,
- analogies to the Relational Data Bases data representation scheme,
- direct knowledge representation mapping to PROLOG and rule-based systems.

Now syntactic, visual, and semantic aspects of XTT will be discussed.

4.1. XTT Structure, Representation, and Semantics

The *syntactic structure* of *XTT* includes the following elements:

- attribute,
- cell,
- header,
- row,
- table,
- connection,
- tree.

They all have been formally defined in [10].

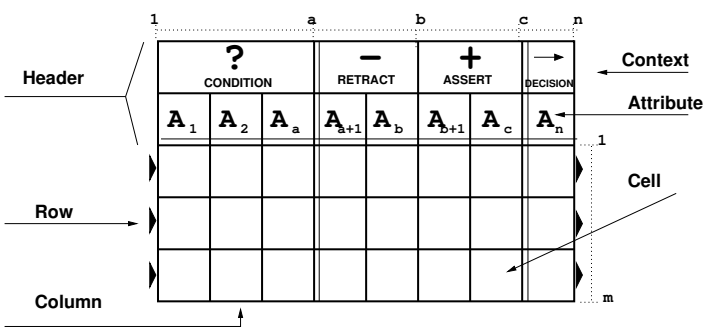

Fig. 1. An example, single XTT Table

The *XTT Visual Representation* is crucial from the rule-based system design point of view. An example of a Table is shown in Figure 1.

A more complex example with several Tables is presented in Figure 2.

The semantic interpretation of *XTT* uses some well-established concepts.

Rule Mapping A *Row* of a *Table* is interpreted as a *production rule*, of a form:

```
IF condition THEN conclusion
```

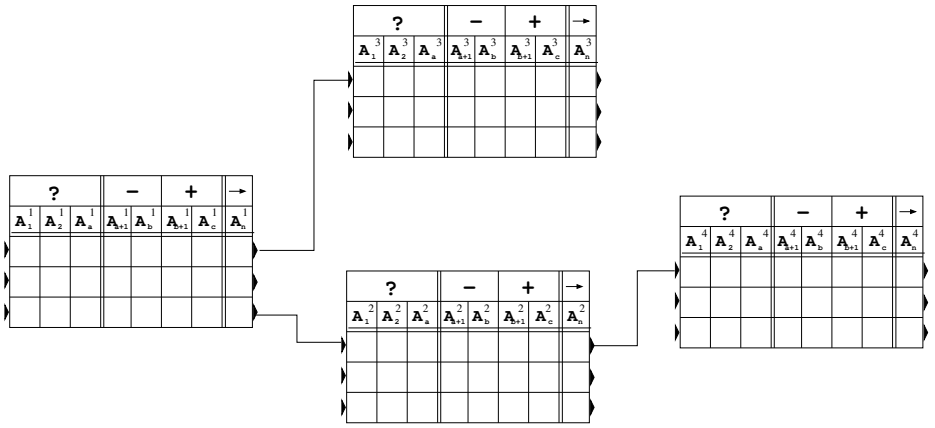

Fig. 2. A simple XTT tree

The `condition` part of the *rule* is mapped to the *Conditional context* of the *Row*. However, the `conclusion` part is mapped to *Assert*, *Retract*, and *Decision contexts* of the *Row*. The use of *Assert*/*Retract* contexts allows for the dynamic modification of the rule-based system knowledge base. So in practice it is an *extended rule*, allowing for *non-monotonic* reasoning, with explicit *control* statements.

Table Mapping A *Table* is simply interpreted as a set of *rules*, where *rule* $j+1$ is processed after *rule* j. However, the *rules* grouped in a *Table* share the same attributes. This concept is similar to *Decision Tables* and to Relational Data Base knowledge representation.

Tree Mapping A concept of *Tree* allows for building a *hierarchy* of *Tables*. Each *Row* x of a *Table* w can have a right *Connection* to another *Row* z in another *Table* y. Such a connection implies logical an `AND` relation in between. Rule processing is then transferred from *Row* j in *Table* x to *Row* k in *Table* y. This concept is similar to *Decision Trees*.

Logical Aspect An important feature of XTT is the fact, that besides its visual representation, they have a well-defined, *logical form* which may be formally analyzed. It is accomplished using *XTT to Prolog mapping*. Any subset of an *XTT Tree* hierarchy can be mapped to a corresponding PROLOG code, in which a specific extended rule structure is used. This representation is crucial to the formal verification of *XTT*.

It is worth noting that, while this does not introduce a new semantic interpretation, XTT can be represented in an XML-based *XTTML* (*XTT Markup Language*) suitable for import and export operations as well as translated to XML-based formats such as *RuleML*. It opens up possiblities for machine processing.

Let us now briefly elaborate on how XTT compares to standard knowledge representation methods.

4.2. XTT and Standard Representation Methods

XTT incorporates best ideas found in classic knowledge representation methods. It has several new aspects and aims at creating a *better* design formalism.

It *takes* from:

- *production rules* the basic knowledge representation concept,
- *decision tables* the ability to combine and represent a set of rules related to a common situation, or context,
- *decision trees* the idea of a hierarchy of knowledge modules.

It may be considered better than:

- *production rules* since it offers a more abstract and transparent knowledge representation, along with inference control aspects, useful in design,
- *decision tables* because it allows for building a hierarchical and modular structure of a knowledge base,
- *decision trees* since it offers greater flexibility and density in terms of knowledge specification.

XTT offers an attribute-based and rule-based hierarchical knowledge representation, with intuitive visual representation, and well-defined logic-based semantics. Using an XTT-based design approach, it is possible to offer an *integrated design and implementation process* for rule-based systems.

5. Integrated Design Process

The XTT language plays a key role in the new approach to RBS design and implementation process. The proposed approach [10] follows the structural methodology for design of information systems. It is simultaneously a top-down approach, which allows for incorporating hierarchical design – in fact, any tabular component can be split into a network of more detailed components, and a network of components can be grouped together to form a more abstract component. The approach covers the stages of conceptual, logical and physical design. The principles of the integrated design process are based on selected existing approaches to system design [10, 4].

The approach proposed herein does not aim at covering the whole system life-cycle, as for example the one that can be found inclassical software engineering. However, it does aim at including all phases of the system life-cycle from the design to implementation phase. The following three design phases are identified:

1) *Conceptual design*, in which the basic structure of the system is identified, along with data and control flow, as well as main operating contexts, objects and their attributes; this allows for further defining the headers of XTT tables.
2) *Logical design*, which involves building table rows (corresponding to rules), connecting tables; the XTT structure can be incrementally built, analyzed, and possibly verified and even optimized on-line.

3) *Physical design*, in which a preliminary implementation is done by building a PROLOG code (or any other target language since the approach is of generic character), which can be executed, compiled, debugged and possibly translated to system-specific representation.

This is a top-down approach. The names of design phases are similar to Relational-Data Base design phases. However, the actual stages in each phase are different. One of the most important features of this approach is the separation of logical and physical design, which also allows for a transparent, hierarchical design process. All of the stages discussed above are supported by MIRELLA, an integrated CASE environment. Subsequent design phases are presented in Figure 3.

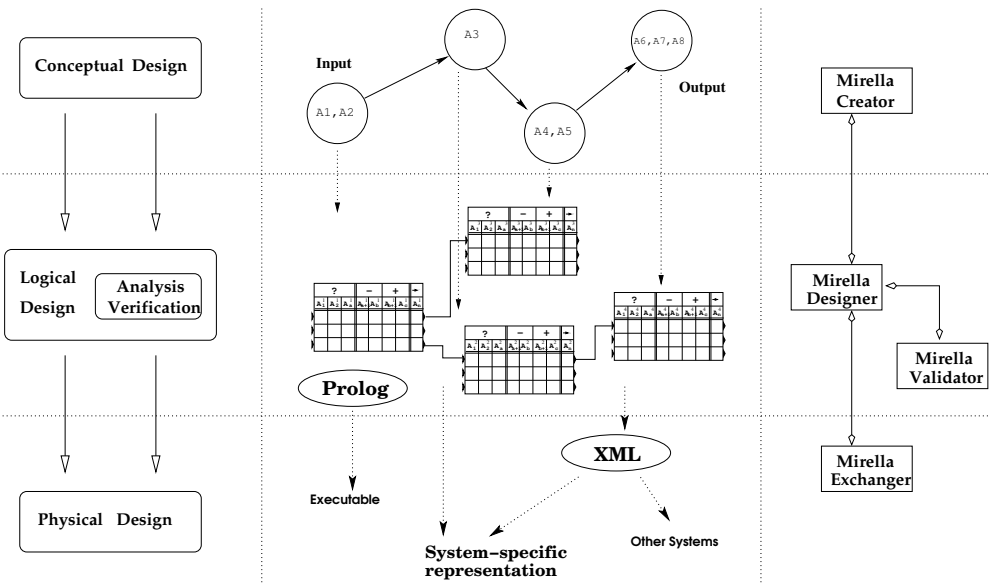

Fig. 3. Phases of the integrated design process

6. Mirella

MIRELLA [10, 15] is an intelligent visual design tool supporting on-line verification of rule-based systems, based of the XTT knowledge representation. It is oriented towards designing reliable and safe rule-based systems in general. The main goal of the system is to move the design procedure to a more abstract, logical and graphical level, where knowledge specification is based on use of abstract rule representation. The designed graphical specification is automatically translated into a predefined XML (XTTML) knowledge format, so the designer can focus on logical specification of safety and reliability; simultaneously, practical code can be generated form a wide class of systems. On the other hand, formal aspects such as completeness, determinism,

etc. may be automatically verified on-line during the design, using PROLOG-based representation, so that it verifiable characteristics are preserved.

6.1. System Architecture

The MIRELLA architecture is shown in Figure 4. It consists of several modules and layers. There are four main system layers: *environmental* provides ways of importing, exporting and presenting knowledge base in both human and machine readable formats, *visual* allows for visual design of knowledge base, *logical* provides means of formal verification of knowledge base, and *semantical* it helps in adding domain-specific semantics to the knowledge base. The *knowledge base* itself is a repository for XTT based system description.

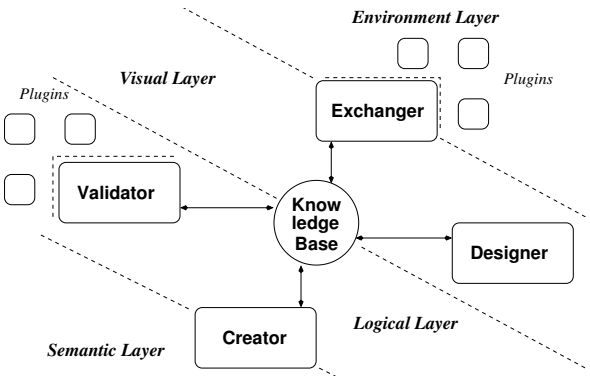

Fig. 4. Architecture of MIRELLA environment

There are four principal modules of the system:

1) *Creator*, which supports defining of system attributes with specific constraints, and semantic information,
2) *Designer*, which allows for visual design of system knowledge base using XTT schema,
3) *Validator*, which performs on-line analysis of system formal properties via an integrated PROLOG compiler,
4) *Exchanger*, which consists of number of import and exports plugins, exchanging data with other knowledge representation languages, and domain specific system prototypes.

6.2. Development Platform

The development platform for the environment has been chosen with flexibility, portability and efficiency in mind. The platform consists of several middleware layers supporting GUI, canvas management, PROLOG integration and XML handling. This architecture is object-oriented. However, it is implemented in pure ANSI C language, and is able to run on multiple platforms. While the MIRELLA environment was imple-

mented in the GNU/Linux environment, it can be easily ported to Sun Solaris, and possibly Windows platform.

6.3. Prolog Integration

The SWI-Prolog, a modern ISO-compliant portable Prolog compiler, has been chosen for the Prolog integration module. It was selected due to its: flexibility, portability, availability, and possibility of integration with other software. It offers a bi-directional programming interface to and from ANSI C language allowing for easy integration with other code. It provides a built-in XML/RDF parser, suitable for direct XML processing and applications in the field of Semantic Web.

7. Application Example

Mirella can significantly improve knowledge engineering process by supporting the design and implementation of RBS. In [10] a complete description of several real-life RBS examples designed with Mirella can be found.

Let us consider a simple but illustrative rule-based control system for setting the required temperature in a room, depending on the day, season, hours, etc. The example is based on Thermostat example found in [17]. The goal of the system is to set a temperature at a certain *set point*, which is the output of the system. The input is the current time and date. The temperature is set depending on the particular part of the week, season, and working hours. Most important parts of the Mirella-based design are presented in the following paragraphs; the full solution may be found in [10].

7.1. Conceptual Design

During conceptual design phase system input, state and output has been considered. A resulting formal attribute specification, conforming to XTT method is shown in Table 1.

Table 1

XTT Thermostat attributes specification

Name	Subset	Type	Constraints
day	input	enumerative, symbolic	{`monday, tuesday, wednesday, thursday, friday, saturday, sunday`}
time	input	integer	⟨ `0,24` ⟩
month	input	enumerative, symbolic	{`january, february, march, april, may, june, july, august, september, october, november, december`}
today	middle	enumerative, symbolic	{ `workday, weekend` }
season	middle	enumerative, symbolic	{ `spring, summer, autumn, winter` }
operation	middle	boolean	–
thermostat	output	enumerative, integer	{`14, 15, 16, 18, 20, 24, 27`}

Having the attributes specified, and considering the original rule base, the XTT design can be introduced.

7.2. Logical Design

The rules in the original rule-base can be divided in a natural way into groups producing the same kind of decision, where the precise decision depends on precise preconditions. Furthermore, in the groups the preconditions of the rules employ the same linguistic variables, but different values.

In fact, there are four different groups of rules defining decisions whether *today* is `workday` or `weekend`, whether the *time* is during business hours or not, deciding what season do we have, and finally indicating the setting of the thermostat.

For each group a separate XTT table can be built – this seems to be a natural and efficient approach. Since the rules uses the same variables, the specification of attributes in the table is just given once and all the columns are necessary for any rule in a table.

This results in creating four XTT tables:

1) **today** table infers the value of *today* attribute, it is a *root table*,
2) **operation** table checks the time of the day,
3) **season** table infers the current *season*,
4) **temperature** table makes the decision about thermostat temperature.

The XTT structure for the thermostat system has been designed using Mirella Designer. All of the tables and connections between them are shown in Figure 5.

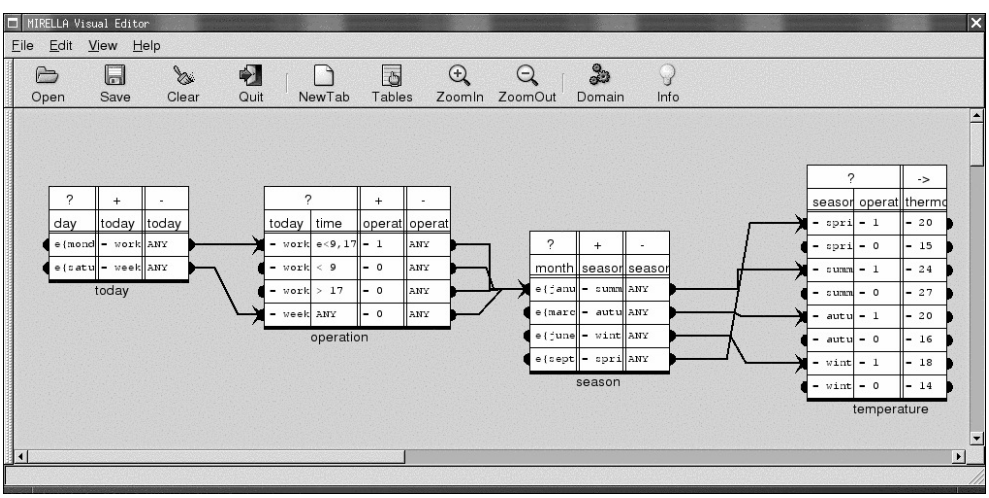

Fig. 5. Thermostat system design in Mirella

7.3. Physical Design

During the physical design a complete system description in PROLOG has been genereted by MIRELLA. An example of the PROLOG code describing the `today` table follows:

```
%%%Table: today
rule(1,1,[f(aDD,set,sWD)],[f(aTD,set,_)],[f(aTD,atomic,wd)], [],2,3).
rule(1,2,[f(aDD,set,sWK)],[f(aTD,set,_)],[f(aTD,atomic,wk)], [],2,6).
```

The table is mapped into two rules, facts in XTT cells are represented by `f/3` term. A fact includes attribute name, type (atomic, nonatomic, such as set of values), and value. Rules shown above use some explicitly predefined sets of values:

```
set(sWD,[monday,tuesday,wednesday,thursday,friday]).
set(sWK,[saturday,sunday]).
```

During the physical design phase it is also possible to conduct an interactive simulation using the PROLOG-based system prototype.

7.4. Analysis

The specification of the system with the use of the XTT method seems to be both concise and easy to analyze. The original Thermostat example is specified in such way that the system has basic properties such as determinism, completeness and lack of subsumption preserved.

However, the example reduction plugin, provided with MIRELLA, is able to detect a possible *reduction*:

```
?- vpr(4).
*** Rule: 4.11 may be glued with rule: 4.15
    reduced fact: f(aSE, set, [spr, aut])
*** Rule: 4.15 may be glued with rule: 4.11
    reduced fact: f(aSE, set, [aut, spr])
No more reduction of rules in table 4
```

The following rules:

```
rule(4,11,[f(aSE,atomic,spr),f(aOP,atomic,yes)],
          [],[],[f(aTHS,atomic,20)],0,_).
rule(4,15,[f(aSE,atomic,aut),f(aOP,atomic,yes)],
          [],[],[f(aTHS,atomic,20)],0,_).
```

could be substituted by a single rule with use of a *non-atomic* value:

```
rule(4,11,[f(aSE,set,[aut, spr]),f(aOP,atomic,yes)],
          [],[],[f(aTHS,atomic,20)],0,_).
```

The XTT approach allowed for dividing the system knowledge base into four interconnected modules. Non-atomic attribute values proved to be useful in specifying precondition attribute values and in table reduction.

8. Concluding Remarks

The paper presents a new, integrated approach to rule-based systems design and implementation. It introduces the XTT knowledge representation language, which can be used as a modern knowledge acquisition tool. The main novel features of the proposed approach include: highly transparent and intuitive visual knowledge representation and manipulation, assuring integrated design process through incorporating the verification stage into design as an intrinsic, available on-line feature, and incorporation of inference control mechanism into design of the rule-base.

The MIRELLA CASE tool, introduced in the paper has both scientific and engineering applications [10]. The application of RBS design methods in the field of computer security (firewall systems) was first proposed in [12]. However since then this idea was extended, and presented in more detail in [10]. Possible application of MIRELLA in the field of computer security systems is considered an important direction for future work. Another effort aims at combining some of MIRELLA features with the Adder project `http://adder.ia.agh.edu.pl` developed as a part of KBN Research Project No 4 T11C 035 24. The project deals with application of colored Petri nets as a formal method for requirements specification of real-time systems, and as an algebraic and graphical language for design of executable models of such systems [16].

Through integrating design and analysis within a single methodology, supported by a CASE tool, the work presented in the paper opens new possibilities in the domain of architecture of expert systems design. MIRELLA home page, hosting information about current status of the project, is located at `http://mirella.ia.agh.edu.pl`. The MIRELLA project was developed as a part of KBN Research Project No 4 T11C 027 24.

References

[1] Andert E.P.: *Integrated knowledge-based system design and validation for solving problems in uncertain environments.* Int. J. of Man-Machine Studies, 36, 1992, p. 357–373

[2] Hopgood A. A.: *Intelligent Systems for Engineers and Scientists.* 2nd ed. Boca Raton London, New York Washington, CRC Press, D.C. 2001, ISBN 0-849-30456-3

[3] Jackson P.: *Introduction to Expert Systems.* 3rd ed., Addison–Wesley, 1999, ISBN 0-201-87686-8

[4] Liebowitz J. (Ed.): *The Handbook of Applied Expert Systems.* Boca Raton, CRC Press 1998, ISBN 0-8493-3106-4

[5] Ligęza A.: *Towards design of complete rule-based control systems.* In Kocijan R. K. J. (Ed.), IFAC/IMACS International Workshop on Artificial Intelligence in Real-Time Control, Bled, IFAC, Slovenia, 1995, p. 189–194

[6] Ligęza A.: *Logical support for design of rule-based systems. reliability and quality issues*. In M. Rousset, editor, *ECAI-96 Workshop on Validation, Verification and Refinment of Knowledge-based Systems*, volume W2, pages 28–34. ECAI'96, Budapest, 1996.

[7] Ligęza A.: *Intelligent data and knowledge analysis and verification; towards a taxonomy of specific problems*. Validation and Verification of Knowledge Based Systems: Theory, Tools and Practice, pages 313–325, 1999.

[8] Ligęza A.: *Logical Foundations for Rule-Based Systems*. Volume KU 0146, Scientific Publishers of AGH-UST, Cracow 2005, ISBN 83-89388-37-5

[9] Ligęza A., Wojnicki I., Nalepa G.: *Tab-trees: a case tool for design of extended tabular systems*. In H. M. et al., editor, Database and Expert Systems Applications, volume 2113 of Lecture Notes in Computer Sciences, Springer-Verlag, Berlin, 2001, p. 422–431

[10] Nalepa G. J.: *Meta-Level Approach to Integrated Process of Design and Implementation of Rule-Based Systems*. Ph.D. Thesis, AGH University of Science and Technology, AGH Institute of Automatics, Cracow, Poland, September 2004.

[11] Nalepa G. J., Ligęza A.: *Graphical case tools for integrated design and verification of rule-based systems*. In Burczyński T., Cholewa W., Moczulski W. (Ed.), Methods of Artificial Intelligence, Proceedings of the Symposium on Methods of Artificial Intelligence AI-METH 2002, Gliwice, Poland, Silesian University of Technology, 2002, p. 307–3013

[12] Nalepa G. J., Ligęza A.: *Designing reliable web security systems using rule-based systems approach*. In E. Menasalvas, J. Segovia, and P. S. Szczepaniak, editors, Advances in Web Intelligence. First International Atlantic Web Intelligence Conference AWIC 2003, Madrid, Spain, May 5-6, 2003, volume LNAI 2663 of Lecture Notes in Artificial Intelligence, Berlin, Heidelberg, New York, Springer-Verlag, 2003, p. 124–133

[13] Nalepa G. J., Ligęza A.: *Integrated design environment for formal verification of rule-based systems*. In Z. Bubnicki and A. Grzech, editors, Inżynieria Wiedzy i Systemy Ekspertowe, Wrocław, 2003. Oficyna Wydawnicza Politechniki Wrocławskiej, volume II, p. 30–37,

[14] Nalepa G. J., Ligęza A.: *A graphical tabular model for rule-based logic programming and verification*. In Z. Bubnicki and A. Grzech, editors, Proceedings of 15th International Conference on Systems Science, Wrocław, Wrocław University of Technology, Oficyna Wydawnicza Politechniki Wrocławskiej 2004

[15] Nalepa G. J., Ligęza A.: *A visual edition tool for design and verification of knowledge in rule-based systems*. In Bubnicki Z. Grzech A. (Ed.), Proceedings of 15th International Conference on Systems Science, Wrocław, Wrocław University of Technology, Oficyna Wydawnicza Politechniki Wrocławskiej, 2004

[16] Nalepa G. J., Szpyrka M.: *On two formal approaches to design and verification of embedded rule-based systems*. In Proceedings of WRTP2004, submitted to 28th IFAC/IFIP Workshop on Real-Time Programming, 2004

[17] Negnevitsky M.: *Artificial Intelligence. A Guide to Intelligent Systems*. Harlow, England; London; New York, Addison-Wesley 2002, ISBN 0-201-71159-1

[18] van Harmelen F.: *Applying rule-based anomalies to kads inference structures*. ECAI'96 Workshop on Validation, Verification and Refinement of Knowledge-Based Systems, 1996, p. 41–46