Proposal of a Formal Verification Framework for the XTT2 Rule Bases*

Agata Ligęza, Grzegorz J. Nalepa

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

Abstract. *In the paper practical verification algorithms for the XTT rules are described. They are implemented as modules for the HalVA framework and allow for the local verification of completeness, determinism, and incosistency. The evaluation of the algorithms is given as well as an application provided.*

1. Introduction Rule-based systems (RBS) [8] are an important class of intelligent systems [4]. Their formal description allows for a formal analysis of important system properties. Therefore it is possible to assuer their quality and safate at the early design stages.

The main focus of the paper is the formal verification of rules [5, 3]. The state-of-the art in the area is given, with some of the most important properties, such as completeness, conciseness or determinism discussed [19]. These properties have to be considered w.r.t. to a given knowledge formalization format. Therefore, the rule formalization for the XTT representation is given [11, 13]. The representation introduces a structured rule base composed of extended decision tables linked in a tree-like structure. The rule formalization is given using the ALSV(FD) logic [8, 12]. Considering the complex nature of the XTT knowledge base structure, the scope of verification has to be considered. The focus of this paper is on the local, table level verification.

The approach presented in the paper allows for an on-line verification of the XTT knowledge base, during the design. The model can be built incrementally.

*The paper is supported by the HeKatE Project funded from 2007–2009 resources for science as a research project.

2. XTT2 Rule Language Concepts The formalization for XTT^2 representation is based on ALSV(FD) logic [8, 12]. The ALSV(FD) provides a much higher expressive power than the propositional calculus, while providing tractable inference. Therefore, a format of rule is more complex. In the general case, the rule, expressed by attributive logic, is represented as (1)

$$\begin{array}{ll}\text{rule}(i): & \psi \ \wedge \\ & A_1 \in t_1 \wedge A_2 \in t_2 \wedge \ldots \wedge A_n \in t_n \\ & \longrightarrow \\ & retract(B_1 = b_1, B_2 = b_2, \ldots, B_b = b_b) \\ & assert(C_1 = c_1, C_2 = c_2, \ldots, C_c = c_c) \\ & H_1 = h_1, H_2 = h_2, \ldots, H_h = h_h \\ & next(j), \ else(k)\end{array} \tag{1}$$

In (1) a formula ψ desribes context. $A_1 \in t_1 \wedge A_2 \in t_2 \wedge \ldots \wedge A_n \in t_n$ is a precondition formula. C_j ($j = 1, \ldots, c$) and B_i ($i = 1, \ldots, b$) correspond to facts to arrert and to retract from the knowledge base. Conclusions (decisions or actions) are represented by $H_1 = h_1, H_2 = h_2, \ldots, H_h = h_h$. There is also a control statement introducing the next or alternative rule.

In the case of XTT^2 representation, the logical rule format is as follows:

$$r\colon (A_1 \propto_1 V_1) \wedge (A_2 \propto_2 V_2) \wedge \ldots \wedge (A_n \propto_n V_n) \longrightarrow RHS, \tag{2}$$

where $\propto_i \in \{=, \neq, \in, \notin\}$ for simple attributes and $\propto_i \in \{=, \neq, \subseteq, \supseteq, \sim, \not\sim\}$ for general attributes. The form of RHS is as in (1).

The representation introduces a structured rule base composed of extended decision tables linked in a tree-like structure. In order to be able to practically implement inference and verification procedures, a textual meta representation for XTT called HMR is discussed [11, 13].

In HMR *a type* of an attribute is described by properities as: name, description, base (numerical or symbolic), order (default no), and domain. For example, the attribute corresponding to days of a week can be represented as:

```
xtype [name: week_days,
     base: symbolic,
     ordered: yes,
     domain: [monday,tuesday,
      wednesday, thursday,
      friday,saturday,sunday]
    ].
```

A definition of an attribute is introdeced with a frame:

```
xattr [name:  STRING,
       class: simple|general
       type:  STRING,
       comm:  in|out|inter|comm,
       desc:  STRING,  %optional
       abbrev: STRING  %optional
      ].
```

The schema of the XTT table in HMR is represented by a relation between attributes. If the *day* is attribute describing a day of a week and *today* represents a work day of a weekend, the relation between the day and today is included in a table:

```
xschm dt: [day] ==> [today].
```

Rules in the table have a form:

```
xrule dt/1:
  [day in [monday,tuesday,
   wednesday,thursday,friday]]
  ==>
  [today set workday].

xrule dt/2:
  [day in [saturday,sunday]]
  ==>
  [today set weekend].
```

To represent a rule in HMR the name of the table and the rule id have to be specified. The representation can be directly interpreted by Prolog-based HeaRT inference engine. It is also used in the verification framework.

3. Formal Analysis of XTT Rules The quality of a rule-based system is dependent on the quality of a knowledge base. What is more important, anomalies in the set of rules could be a result of serious faults in system's responses. Therefore the analysis of knowledge base is significant step during developing rule-based system.

The issue of verification and validation were discussed by many authors. Differences in their approaches to V&V start at the definition level. To unify them, in this paper verification and validation processes are defined as follows:

Verification is a process in the early design phase, aimed at checking if the system meets its constraints and requirements ([18, 1, 15, 16, 17]).

Testing is a process aimed at analysing the system work, by comparing system responses to known responses for special input data ([18]).

Validation is a case of testing, aimed at checking if the system meets user's requirements ([18]).

A summary result of analysis techniques and tools is presented in [20].

The classification of potential errors and deformation of knowledge base was also widely discussed. From the formal point of view anomalies of knowledge base could be divided into three main categories: 1) *incompleteness*, 2) *indeterminism*, and 3) *overdeveloped set of rules*.

Let the knowledge base be described by rules:

$$\begin{array}{lrcl} r_1\colon & \Psi_1 & \rightarrow & h_1 \\ r_2\colon & \Psi_2 & \rightarrow & h_2 \\ & \vdots & & \\ r_n\colon & \Psi_n & \rightarrow & h_n \end{array} \tag{3}$$

Completeness assures that for any input state the system reacts and produces some response (conclusion, decision or action) ([6]). In other words, the system with the set of rules (3) is *logically complete* if a disjunction of preconditions is a tautology:

$$\models \Psi_1 \ \vee \ \Psi_2 \ \vee \ \ldots \ \vee \ \Psi_n$$

The knowledge base is incomplete, when in the set of rules exist *unreachable clauses*, *dead-end clauses* or some rules are *missing*.

The unreachable clause exists if rule:

$$r\colon \ P(x) \rightarrow Q(x)$$

can be found in in the set (3), and $Q(x)$ is not the system response and do not satisfy preconditions of any other rule.

In (3) a formula $\Psi'_1, \Psi'_2, \ldots, \Psi'_n$ is missing, if

$$\models \Psi_1 \vee \Psi_2 \vee \ldots \vee \Psi_m \vee \Psi'_1 \vee \Psi'_2 \vee \ldots \vee \Psi'_n$$

but a formula:

$$(\Psi_1 \vee \Psi_2 \vee \ldots \vee \Psi_n) \wedge \left(\Psi'_1 \vee \Psi'_2 \vee \ldots \vee \Psi'_m\right)$$

is never satisfied.

Determinism guarantees that the system always produces the same reaction for the same input data. In other words for any input state the system finds a unique solution ([6]). From the formal point of view the set (3) is indeterministic "if there exists a state described by formula ψ, such that simultaneously $\phi \models \Psi_1$ and $\phi \models \Psi_2$ and $h_1 \neq h_2$" ([7]).

The system is indeteministic, if there are *contradictory rules* in knowledge base. Rules r_i and r_j are contradicted, if there exists a state ϕ, such that $\phi \models \Psi_i$ and $\phi \models \Psi_j$, but under the considered interpretation $I \not\models_I h_i \wedge h_j$.

Inconsistency also is a cause of indeterminism. "Two rules r_1 and r_2 are inconsitent if there existsa state described by formula ϕ, such that simultaneously $\phi \models \Psi_1$ and $\phi \models \Psi_2$, but $\not\models h_1 \wedge h_2$" ([7]).

Minimal number of rules indicates a set of rules without *redundant*, *subsumed* rules. What is more, the set of rules should produce the same reactions as a overdeveloped set.

The formal classification of reduncancy is presented in [15]. The redundancy in rule chain is the most general form of redundancy. In this case two rules r_i and r_j are redndant, if

$$\begin{aligned} r_i&: \ P(x) \rightarrow Q(x) \rightarrow R(x) \\ r_j&: \ P(x) \rightarrow R(x) \end{aligned}$$

and $Q(x)$ is not satisfied in any other rule.

All of above features – completeness, determinism, minimal number of rules – should be provided to assure reliability, safety and efficiency of the rule-base system ([6]).

In systems built on XTT^2 rules ([11, 13]), verification and validation procedures are simplified. The design process of the system is based on top-down methodology ([9, 10]). Therefore, every step in the process can be analysed. The model can be built incrementally. This solution allows to provide a *incremental verification* and – in a consequence – save computational resources.

The XTT^2 representation introduces a structurization of knowledge base and simplify pointing out *contexts*. Implicitly the context is identified with an extended decision table. An isolation of contexts allows to provide *local analysis* – contexts can be verified separately.

4. HalVA Verification Framework The analysis of the XTT knowledge base is provided by HalVA Verification Framework. The main purpose of the framework is the local verification.

HalVA consists of Prolog-based predicates. The verification framework was developed as a plug-in of the HeaRT inference engine [14]. HalVA provides the verification of completeness, contradiction and subsumption. What is more, the number of rules can be reduced. In order that the verification is focused on a local level, the schema of the XTT table is considered.

All of the verification procedures presented bellow are based on inference rules for ALSV(FD) introduced in [10]. For state described by ϕ a precondition $(A_i \propto_i V_i)$ (where $\propto_i \in \{=, \neq, \in, \notin\}$ for the simple attribute and $\propto_i \in \{=, \neq, \subseteq, \supseteq, \sim, \not\sim\}$ for general attribute, $i = 1, \ldots, n$) is satisfied, if simultaneously:

- V_i is a value from a domain of A_i,
- ϕ_{A_i} is a value from a domain of A_i, where ϕ_{A_i} is a value of attribute A_i in formula ϕ,
- a clause $(A_i = \phi_{A_i})$ is a logical consequence of a clause $(A_i \propto_i V_i)$.

In practise, those inference rules are implemented in the HeaRT engine by a predicate `alsv_valid/2`. The predicate has a complex form. Therefore, in this paper only base clauses are presented.

For simple attributes one of clauses has form:

```
alsv_valid(Att in [L to U], State) :-
  alsv_attr_class(Att,simple),
  xstat State: [Att,StateValue],
  !,
  alsv_values_check(Att,
                    [StateValue]),
  alsv_values_check(Att,[L]),
  alsv_values_check(Att,[U]),
  normalize(Att,[StateValue],
            [NormStateValue]),
  normalize(Att,[L],[NormL]),
  normalize(Att,[U],[NormU]),
  NormStateValue =< NormU,
```

```
  NormStateValue >= NormL.
```

This clause is adequate for a condition ($A \in V$), where A is a simple attribute and V is a value set. In this case the value set V is introduced by lower and upper bounds. It means, the domain of the attribute A is ordered.

An example of clause for general attributes is:

```
alsv_valid(Att sim Set,State) :-
  alsv_attr_class(Att,general),
  xstat State: [Att,StateValue],
  !,
  alsv_values_check(Att,StateValue),
  alsv_values_check(Att,Set),
  normalize(Att,StateValue,
            NormStateValue),
  normalize(Att,Set,NormSet),
  intersection(NormStateValue,
               NormSet,[_|_]).
```

The clause corresponds to preconditions ($A \sim V$), where A is a general attribute and V is a value set. All clauses for predicate `alsv_valid` are included in HeaRT.

The basic idea in the verification of *completeness* consists in checking all input states. Domains of attributes are taken into consideration. The cartesian product of domains determine all states for the context (table). For an every tuple, corresponded to the input state, the algorithm checks, if preconditions of any rule are satisfied. If there is no rule to execute, the considered state is reported as uncover. Based on all uncovered states, a proposal of a new rule is introduced.

The analysis ends, when all states are checked. Domains of attributes are finite. Therefore, the verification procedure terminates after a finite number of discrete steps. However, cardinalities of domain sets can be outsized. This could result in a combinatorial explosion. To save computational resources, the set of input states is limited. It means, not all but only possible states are considered.

Verification of *contradiction* is based on a pairwise comparison of rules. Two rules, executable in the same time (for the same state), are taken into consideration. The comparison concerns the right-hand side of rules. If conclusions are inconsistent, the conflict is reported. The verification procedure stops when all possible comparisons are done.

The pairwise comparison of rules is also used to verify *subsumption*. However, the analysis concerns both sides of rules. One rule is subsumed by another, if its preconditions are more specific, but simultaneously conclusions are more general. The verification procedure finds two rules in the context (table), executable for the same state. Then checks, whether there is relation between conclusions.

The algorithm provides all comparisons. This strategy allows to detect identical rules. In this case, a rule is reported as subsuming another and the other subsuming the first one.

HalVA allows to reduce an overdeveloped set of rules. The reduction can be done by using the dual resolution ([7]). If rules produce the same conclusions and in the precondition part exists at least one the same clause, the rest of the clauses are joined into one formula. All possible reductions are reported. What is more, proposals of new rules are introduced.

5. Verification Example Let the thermostat control system ([7]) be considered. The table-leveled analysis is provided by HalVA Verification Framework. The HMR representation is used.

Incompleteness Let the table *th* be described:

```
xschm th: [today,hour] ==> [operation].
```

The table consists only of two rules:

```
xrule th/1:
  [today eq workday,
   hour  gt 17]
  ==>
  [operation set not_bizhours].
xrule th/2:
  [today eq weekend,
   hour  eq any]
  ==>
  [operation set not_bizhours].
```

The verification of completeness points out uncovered states in the system.

```
>> vcomplete(th).
In th uncover states:
[[today, workday], [hour, 0]]
   [hour, 1]]    [hour, 2]]
   [hour, 3]]    [hour, 4]]
   [hour, 5]]    [hour, 6]]
   [hour, 7]]    [hour, 8]]
   [hour, 9]]    [hour, 10]]
   [hour, 11]]   [hour, 12]]
   [hour, 13]]   [hour, 14]]
```

```
   [hour, 15]]   [hour, 16]]
   [hour, 17]]
New rule: [today in [workday],
hour in [17, 16, 15, 14, 13, 12,
         11, 10, 9, 8, 7, 6, 5,
         4, 3, 2, 1, 0]] ==> _
No more uncovered states by table th
```

A rule marked as a new rule is a suggestion.

Contadiction The table *dt* is considered:

```
xschm dt: [day] ==> [today].
```

Adequate rules are defined as follows:

```
xrule dt/1:
  [day in [monday,tuesday,wednesday,
   thursday,friday,saturday]]
  ==>
  [today set workday].
xrule dt/2:
  [day in [saturday,sunday]]
  ==>
  [today set weekend].
```

The verification detects contradiction:

```
>> vcontradict(dt).
Clause [ today set weekend ] is
 conflicted with [ today set workday ]
In dt conflicted rules: 1 <=> 2
 for state  [[day, 6]]
No more conflicts in table dt
```

For state `[day, 6]` rules 1 and 2 are activated. But they produce conflicting result.

Subsumption Let a rule:

```
xrule th/5:
  [today eq weekend,
   hour  in [9 to 17]]
  ==>
  [operation set not_bizhours].
```

be added to the table *th*. The analysis of the subsumption shows that there are two rules, executable for the same state, but one of them produce more general conclusion.

```
>> vsubsume(th).
In th rule 2 subsumes 5
No more subsumption by table th
```

In this case, added rule is subsumed by rule already existed in the context (table).

Minimal set of rules In the control system of thermostat dependency between the season, hours and the thermostat settings are described by *os* table. The table consists of rules:

```
% (...)
xrule os/4:
  [operation eq during_bizhours,
   season eq spring]
  ==>
  [therm_set set 20].
% (...)
xrule os/8:
  [operation eq during_bizhours,
   season eq autumn]
  ==>
  [therm_set set 20].
```

Both of presented rules produce the same conclusion. Therefore, they could be replaced by one, more general rule.

```
>> vreduce(os).
In os rule 4 can be joined
  with rule 8 to new rule:
  [operation in [during_bizhours],
   season in [1, 3]] ==>
   [therm_set set 20]
In os rule 8 can be joined
  with rule 4 to new rule:
  [operation in [during_bizhours],
   season in [3, 1]] ==>
   [therm_set set 20]
No more reduction  in table os
```

As it is shown above, the HalVA algorithm reports, that rules 4 and 8 could be joined. The proposal of a new rule is introduced.

6. Evaluation and Related Research All of HalVA verification features are focused on the local analysis. The limitation of the scope indicates a verification of some context (implicity a table) only. Therefore, HalVA procedures can point out anomalies in some specific area. Unfortunately, the question of a knowledge base global condition is unsolved.

The other serious issue is computational complexity of provided algorithms. As was mentioned, verification of completeness could cause a combinatorial explosion, if domains of attributes are outsized. However, this approach was introduced in [17, 18, 15].

The other technique – the pairwise comparison of rules – is also dependent on a size of the considered case. Generally, to verify a set of n rules, $\binom{n}{2}$ comparisons need to be done. Nevertheless, the approach is presented in [6, 15, 16, 17] to verify consistency and contradiction and in [16, 17] to verify redundancy and subsumption.

7. Future Work The great challenge is to provide global verification of XTT^2 knowledge bases. The verification of rules using graph theory is taken into consideration. Anomalies of knowledge bases – in graph-oriented representation – can be defined as follows ([2]):

Inconsistency – "there exists a path from vertex x to its exclusive vertex $\neg X$" ([2]).

Contradiction – in graph exist two paths from vertex X to vertexes Y and $\neg Y$.

Redundancy – from vertex X to Y exist at least two diffrent paths.

Circularity – there is a circus in the graph.

Unreachability – in the graph can be found a vertex, which is not the begin of a path to any output node and is not the end of a path from any input node.

Therefore, the principal idea of global verification consists in representing XTT^2 rules as a directed hipergraph. All of clauses are transformed into vertexes and appropriate hyperarcs are determined. This restructuring allows to provide a verification using graph algorithms.

References.

[1] E. P. Andert. Integrated Knowledge-Based System Design and Validation for Solving Problems in Uncertain Environments. *International Journal of Man-Machine Studies*, 36(2):357–373, 1992.

[2] N. Arman, D. Richards, and D. Rine. Structural and Syntactic Fault Correction Algorithms in Rule-Based Systems. *International Journal od Computing and Information Sciences*, 2(1), 2004.

[3] Joseph Giarratano and Gary Riley. *Expert Systems. Principles and Programming*. Thomson Course Technology, Boston, MA, United States, fourth edition edition, 2005. ISBN 0-534-38447-1.

[4] Adrain A. Hopgood. *Intelligent Systems for Engineers and Scientists*. CRC Press, Boca Raton London New York Washington, D.C., 2nd edition, 2001.

[5] Jay Liebowitz, editor. *The Handbook of Applied Expert Systems*. CRC Press, Boca Raton, 1998.

[6] A. Ligęza. Logical Support for Design of Rule-Based Systems. Reliability and Quality Issues. *ECAI'96 Workshop on Validation, Verification and Refinement of Knowledge-Based Systems*, 1996.

[7] A. Ligęza. *Logical Foundations for Rule-Based Systems*. Uczelniane wydawnictwa Naukowo-Dydaktyczne AGH w Krakowie, 2005.

[8] Antoni Ligęza. *Logical Foundations for Rule-Based Systems*. Springer-Verlag, Berlin, Heidelberg, 2006.

[9] G. J. Nalepa and A. Ligęza. Prolog-Based Analysis of Tabular Rule-Based Systems with XTT Approach. In *FLAIRS Conference*, pages 426–431, 2006.

[10] G. J. Nalepa and A. Ligęza. HeKatE Methodology, Hybrid Engineering of Intelligent Systems. *Int. J. Appl. Math. Comput. Sci.*, 18(3):1–15, 2009.

[11] Grzegorz J. Nalepa and Antoni Ligęza. A graphical tabular model for rule-based logic programming and verification. *Systems Science*, 31(2):89–95, 2005.

[12] Grzegorz J. Nalepa and Antoni Ligęza. Xtt+ rule design using the alsv(fd). In Adrian Giurca, Anastasia Analyti, and Gerd Wagner, editors, *ECAI 2008: 18th European Conference on Artificial Intelligence: 2nd East European Workshop on Rule-based applications, RuleApps2008: Patras, 22 July 2008*, pages 11–15, Patras, 2008. University of Patras.

[13] Grzegorz J. Nalepa and Antoni Ligęza. Hekate methodology, hybrid engineering of intelligent systems. *International Journal of Applied Mathematics and Computer Science*, 2009. accepted for publication.

[14] Grzegorz J. Nalepa, Antoni Ligęza, Krzysztof Kaczor, and Weronika T. Furmańska. Hekate rule runtime and design framework. In Gerd Wagner Adrian Giurca, Grzegorz J. Nalepa, editor, *Proceedings of the 3rd East European Workshop on Rule-Based Applications (RuleApps 2009) Cottbus, Germany, September 21, 2009*, pages 21–30, Cottbus, Germany, 2009.

[15] D. L. Nazareth. Issues in the Verification of Knowledge in Rule-Based Systems. *International Journal of Man-Machine Studies*, 30(3):255–271, 1989.

[16] T. A. Nguyen, W. A. Perkins, T. J. Laffey, and D. Pecora. Checking an Expert Systems Knowledge Base for Consistency and Completeness. In *IJCAI*, pages 375–378, 1985.

[17] Motoi Suwa, Carlisle A. Scott, and Edward H. Shortliffe. *Completeness and consistency in rule-based expert system*, pages 159–170. Addison-Wesley, Reading, Massachusetts, 1985.

[18] J. Tepandi. Verification, testing, and validation of rule-based expert systems. *Proceedings of the 11-th IFAC World Congress*, pages 162–167, 1990.

[19] A. Vermesan and F. Coenen, editors. *Validation and Verification of Knowledge Based Systems. Theory, Tools and Practice*. Kluwer Academic Publisher, Boston, 1999.

[20] J. A. Wentworth, R. Knaus, and H. Aougab. Verification, Validation and Evaluation of Expert Systems. World Wide Web electronic publication: http://www.tfhrc.gov/advanc/vve/cover.htm.