Towards Validation of Rule-Based Systems - The Loop is Closed *

Rainer Knauf and **Ilka Philippow**
Ilmenau Technical University, Faculty of Computer Science and Automation
PO Box 10 05 65, 98684 Ilmenau, Germany
rainer.knauf@theoInf.tu-ilmenau.de

Avelino J. Gonzalez
Dept. of Electrical and Computer Engineering
University of Central Florida
Orlando, FL 32816-2450, USA
ajg@ece.engr.ucf.edu

Klaus P. Jantke
German Research Center
for Artificial Intelligence Ltd.
Stuhlsatzenhausweg 3, 66125 Saarbrücken, Germany
jantke@dfki.de

Abstract

A methodology for the validation of rule-based expert systems is presented as a 5-step process that has three central themes: (1) creation of a minimal set of test inputs that cover the domain, (2) a Turing Test-like methodology that evaluates the system's responses to the test inputs and compares it to the responses of human experts, and (3) use the validation results for system improvement.

This methodology can be performed in loops. The starting point of each cycle is a rule base and the loop ends up in a (hopefully) better rule base.

The first three steps of this process have been published as separate issues in earlier papers by the authors. This paper gives an overview on the entire process and describes the relation between the steps and the system refinement step. In this last step, the rules are modified according to the results of evaluating the test cases. The base of this rule base reconstruction is both a "rule-associated validity" and the existence of a "better rated" human solution.

Introduction

There is abundant evidence of the need for an integrated approach towards validation and verification of complex systems. Actually, the lack of adequate technologies to evaluate these complex systems is a limiting factor to employ them.

Here, we follow the approach of O'Keefe and O'Leary (O'Keefe and O'Leary 1993) who characterize **verification** and **validation** as *building the system right*, and *building the right system*, respectively.

Verification provides a firm basis for the question of whether or not a system meets its specification. In contrast, **validation** asks whether or not a system is considered to be the required one, something that somehow lies in the eye of the beholder. We concentrate on the validation portion, as that is the one more closely related to ensuring appropriate response to inputs.

The heart of the presented methodology is a TURING Test - like technology of a systematic system interrogation, which is composed of the following related steps (Knauf, Abel, Jantke, and Gonzalez 1998):

1. **Test case generation** Generate and optimize a set of test input combinations (test data) that will simulate the inputs to be seen by the system in actual operation.
2. **Test case experimentation** Employ both the system and human expertise to solve the test cases and rate *all* upcoming solutions by the experts anonymously.
3. **Evaluation** Interpret the results of the experimentation and determine validity assessments attributed to the test cases.
4. **Validity assessment** Analyze the results reported above and express them according to the purpose of the statement: (a) validities associated with outputs for expressing the validity to users, (b) validities associated with rules for expressing the validity to system developers, namely knowledge engineers, and (c) validities associated with test cases for system refinement. Of course, the following (machine-supported) system refinement uses (a) and (b) as well as (c).
5. **System refinement** Provide guidance on how to correct the errors detected in the system as a result of the previous 4 steps. This, hopefully, leads to an improved system.

These steps are iterative in nature, where the process can be conducted again after the improvements have been made. Figure 1 illustrates these steps.

Fundamentals

The input-output behavior of a considered domain and of the system to be validated can be formalized as an input set I, and output set O, a target relation (the wanted system's behavior) $\mathcal{R} \subseteq I \times O$, and a (real)

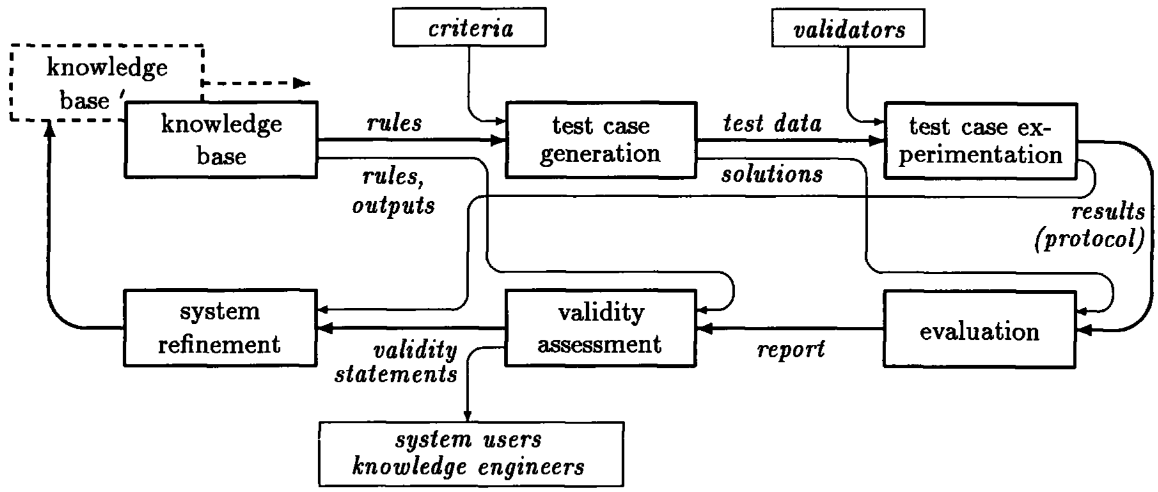

Figure 1: Steps in the Proposed Validation Process

system's behavior $\mathcal{S} \subseteq I \times O$. $\mathcal{R}$ is decomposable into a finite number of convex subsets $\mathcal{R}^o \subseteq I \times \{o\}$. There is at least one $\mathcal{R}^o$ for each output $o \in O$.

In these formal settings the knowledge of a human expert e_i can be expressed as $\mathcal{E}_i \subseteq I \times O$ and should meet *consistency* ($\mathcal{E}_i \subseteq \mathcal{R}$) and *completeness* ($\pi_{inp}(\mathcal{E}_i) = \pi_{inp}(\mathcal{R})$) with respect to $\mathcal{R}$. Ideally, $\mathcal{E}_i$ meets *omniscience* ($\mathcal{E}_i = \mathcal{R}$). An expertise $\mathcal{E}_i$ is said to be *competent*, exactly if it is complete and consistent:

$$competence = consistency + completeness$$

Practically, because of not directly knowing $\mathcal{R}$, we estimate $\mathcal{R}$ by $\bigcup_{i=1}^{n} \mathcal{E}_i$.

Based on these formalisms, we are now able to develop our validation scenario:

- There is assumed a (non-accessible) desired target behavior $\mathcal{R} \subseteq I \times O$.
- There is a team of n experts which is considered to be omniscient as a team, although not necesarily individually.
- There is a system to be validated with an input/output relation $\mathcal{S}$.

Our validation methodology deals with relating the system's behavior to the experts' knowledge. A deeper discussion of the fundamentals can be found in (Knauf, Jantke, Gonzalez, and Philippow 1998).

Generation of Test Cases

The main test case generation approach is to generate a "quasi exhaustive" set of test cases (*QuEST*) (Abel, Knauf, and Gonzalez 1996), to define some validation criteria, and to use these for a reduction of *QuEST*, down to a "reasonable" set of test cases (*ReST*) as described in (Abel and Gonzalez 1997), e.g.

Due to simplification reasons but also because of its practical relevance, we consider rule-based systems with an input I of an m-dimensional "input space", in which each dimension is "atomic", i.e. not compound in any way, and an output O of a set of possible output values.

The generation procedure contains a step of analyzing the dependencies between the inputs and outputs of the system. This is a basis for the reduction procedure, which needs the so called *dependency sets.* These sets describe which output depends on which inputs. A set of so called *critical values* that describe certain values of a single input that are considered a trigger value for the output.

Experimentation and Evaluation

There are two gaps between the (non-formalized) real domain knowledge, and the formalized knowledge of the system: One is between the desired target domain behavior and the experts' knowledge ($\mathcal{R} \rightleftharpoons \mathcal{E}_1, \ldots, \mathcal{E}_n$) and another one is between the experts' knowledge and the system's specification, which is (in case of successful verification) equivalent to $\mathcal{S}$ ($\mathcal{E}_1, \ldots, \mathcal{E}_n \rightleftharpoons \mathcal{S}$).

Unfortunately, earthly creatures like humans are not capable of bridging the first gap. A technology to bridge the second gap is the subject of this section.

The idea of the TURING test methodology, as described in (Jantke, Knauf, Abel 1997) e.g., is divided into four steps: (1) solving of the test cases by the expert validation panel as well as by the system, (2) randomly mixing the test case solutions and removing their authorship, (3) rating all (anonymous) test case solutions, and (4) evaluating the ratings. In the experimentation, the system is considered the "expert" e_{n+1}, i.e. its expertise is $\mathcal{E}_{n+1}$.

To come up with a validity assessment for the system we consider the expert's assessments of the system solution, but each assessment is weighted with a "local competence" of the rating expert for the considered test case.

This "local competence" of an expert e_i for a test case t_j is estimated by considering (1) the expert's behavior while solving the test case t_j[1], (2) the expert's behavior while rating the test case solutions[2], and (3) the other experts' assessment of the solution of the expert e_i.

The TURING test methodology leads to a validity $v_{sys}(t_j)$ (ranging between 0 and 1) for each of the test case t_j:

$$v_{sys}(t_j) = \frac{1}{\sum_{i=1}^{n} \left(cpt(e_i,t_j) \cdot c_{ij(n+1)}\right)} \cdot \sum_{i=1}^{n} \left(cpt(e_i,t_j) \cdot c_{ij(n+1)} \cdot r_{ij(n+1)}\right)$$

$cpt(e_i, t_j)$ is the competence estimation of an expert e_i for a test case t_j, $r_{ij(n+1)}$ is the rating of the system's solution for the test case t_j given by the expert i, and $c_{ij(n+1)}$ is the certainty of the expert e_i while rating this solution. All these variables range between 0 and 1 (inclusive).

A deeper discussion of the experimentation and evaluation, the competence assessment, and the test case associated validities can be found in (Jantke, Knauf, Abel 1997).

Expressing Validity

There are three different ways to express a system's validity according to the purpose of the statement:

1. A system user might be interested in knowing an average global validity of the system, which can be estimated by

$$v_{sys} = \frac{1}{|ReST|} \sum_{j=1}^{|ReST|} v_{sys}(t_j)$$

   or validities associated with the particular outputs (system solutions), which can be estimated by the average validity of all test cases referring to this system solution:

$$v_{sys}(sol_k) = \frac{1}{|T_k|} \sum_{[t_j,sol_k]\in T_k} v_{sys}(t_j)$$

   with $T_k = \{[t_j, sol_k] \in ReST : t_j \in \pi_{inp}(ReST), [t_j, sol_k] \in \mathcal{E}_{n+1}\}$.

2. A system developer, namely a knowledge engineer, may be interested in knowing validities associated with rules. They can be estimated by the average validity of those test cases the considered rule r_l is used for:

$$v(r_l) = \frac{1}{|T_l|} \sum_{[t_j,sol_k]\in T_l} v_{sys}(t_j)$$

   with $T_l = \{[t_j, sol_k] \in ReST : t_j \in \pi_{inp}(ReST), [t_j, sol_k] \in \mathcal{E}_{n+1}, r_l \text{ is used for } t_j\}$

3. For formal system refinement (cf. next section) besides these validity statements we use the validities associated with test cases $v_{sys}(t_j)$ as described in the previous section.

System Refinement

Here, we introduce a newer part of our research: the reconstruction of the rule base according to the results of validation. This step closes the loop of figure 1. The main idea is to find rules, which are "guilty" in the system's invalidity and to replace them by better ones. A rule is better, if it leads to a solution which got better marks then the system's solution. We use the idea of a rule-based reduction system to construct these "better rules" systematically.

Finding "guilty rules"

All rules having a conclusion part, which is a final solution sol_k are subject of the following considerations:

- There is a rule-associated validity $v(r_l)$ for each rule.
- There is a set T_l^* containing all test cases with test data parts occurring in T_l and all solution parts which came up in the experimentation, regardless of whether the solution is given by a human expert $e_i (1 \leq i \leq n)$ or the system e_{n+1}:
  $$T_l^* = \{[t_j, sol(e_i,t_j)] : \exists [t_j, sol_k] \in T_l\}$$
- T_l^* is splitted acoording to the different solution parts $sol_1, \ldots, sol_p, \ldots, sol_m$ of its elements. This leads to m disjunctive subsets $T_{l1}^*, \ldots, T_{lp}^*, \ldots, T_{lm}^*$. One of the subsets is the one collecting the test cases with the system's solution sol_k.
- Analogously to the computation of the system solution's validity v_{sys}, a validity $v(r_l, sol_p)$ $(1 \leq p \leq m)$ of each solution $sol_1, \ldots, sol_p, \ldots sol_m$ – but only based on the test cases of the corresponding T_{lp}^* – can be computed:

$$v(r_l, sol_p) = \frac{1}{|T_{lp}^*|} \sum_{[t_j,sol_p]\in T_{lp}^*} \frac{1}{\sum_{i=1}^{n} (cpt(e_i,t_j)c_{ijq})} \cdot \sum_{i=1}^{n} (cpt(e_i,t_j) \cdot c_{ijq} \cdot r_{ijq})$$

  Here, i indicates the rating expert, j indicates the test data, q indicates the solver (who might be one of the experts or the system, but doesn't matter here), p indicates the (common) solution $sol_p = sol(e_i, t_j)$[3] of the test cases in T_{lp}^*, and l indicates

[1] Did he/she admit some incompetence by giving the solution "*I don't know!*"?

[2] Did he/she give his/her own solution bad marks after knowing the colleagues' solutions? How often did he/she express certainty while rating the solutions for t_j ?

[3] Since $sol_p = sol(e_i, t_j)$, p occures in the right hand side of the equation as the combination of i and j.

the rule, from which T_{lp}^* is originated.

- There is an "optimal validity" of rule r_l, which is the maximum of all $v(r_l, sol_p)$ among all solutions sol_p occurring in T_l^*. The solution, to which this maximum is associated, is called the optimal solution sol_{opt} for r_l.

  $$v_{opt}(r_l, sol_{opt}) = max(\{v(r_l, sol_1), \dots, v(r_l, sol_m)\}$$

  $v_{opt}(r_l)$ can be considered an upper limit of the reachable rule-associated validity for rule r_l.

  In case $v_{opt}(r_l, sol_{opt}) > v(r_l)$ there is at least one solution within T_l^*, which obtained better marks by the experts than the system's solution. In this case r_l is guilty and has to be modified.

Reduction of the set of "guilty rules"

First, a simple case will be considered: If all test cases within T_l of a guilty rule r_l have the same optimal solution sol_k, the conclusion-part of this rule is substituted by sol_k. Thus, the considered rule won't be "guilty" any longer.

Replacing the if-part of a "guilty rule"

First, T_l of a "guilty rule" has to be splitted into subsets $T_l^s \subset T_l$ according to their optimal solution sol_s.

The new if-part(s) of the new rule(s) instead of a remaining and compiled "guilty rule" r_l are expressions $e_i \in E$[4] of a set of p new alternative rules $\{r_l^1, r_l^2, \dots, r_l^p\}$ for each T_l^s and will be noted as a set of sets

$$P_l^s = \{\{e_1^1, \dots, e_{p_1}^1\}, \{e_1^2, \dots, e_{p_2}^2\}, \dots \{e_1^p, \dots, e_{p_p}^p\}\}$$

here. The corresponding rule set of P_l^s is

$$r_l^1 : \bigwedge_{i=1}^{p_1} e_i^1 \rightarrow sol_s \qquad \dots \qquad r_l^p : \bigwedge_{i=1}^{p_p} e_i^p \rightarrow sol_s$$

Pos is the set of Positions (dimensions of the input space), at which the $t_j \in T_l^k$ are **not** identical.

The generation of the if-parts P_l^s is managed by a *Reduction System*, which is applied in cycles to Triples $[T_l^s, Pos, P_l^s]$ until Pos becomes the empty set $\emptyset$. The starting point of the reduction is $[T_l^s, Pos, P_l^s]$ with

$$P_l^s = \{\{(s_1 = s_1^{ident}), \dots, (s_q = s_q^{ident})\}\}$$

$s_1, \dots, s_q$ are those positions, where all test data $t_j \in T_l^k$ have the same (identical) value s_i^{ident} and Pos is the set of the remaining positions

$$Pos = \{s_i : \neg\exists(s_i = s_i^{ident}) \in P_l^s\}$$

Table 1 shows the rules used to reconstruct the remaining "guilty rules". The reduction terminates, if the situation $[T_l^k, \emptyset, P_l^k]$ is reached.

[4]Formally, $E \subset \{[t_1, r, t_2] : t_1 \in S, t_2 \in (S \cup \Re), r \in \{<, \leq, =, \neq, \geq, >\}\}$ is a set of single *expressions* about input data within the "input space" formed by the dimensions $S = \{s_1, s_2, \dots, s_m\}$.

Recompiling the new rules

In case the *if*-part of a new rule contains a subset of expressions that is the *if*-part of another rule having an intermediate solution as its *then*-part, this subset has to be replaced by the corresponding intermediate solution:

$$\begin{array}{l}
\exists r_i : (if\text{-part}_1 \rightarrow int_1) \\
\exists r_j : (if\text{-part}_1 \wedge if\text{-part}_2 \rightarrow int\text{-}or\text{-}sol) \qquad \Rightarrow \\
\quad r_j : (if\text{-part}_1 \wedge if\text{-part}_2 \ \rightarrow \ int\text{-}or\text{-}sol) \hookrightarrow \\
\qquad\quad (int_1 \wedge if\text{-part}_2 \ \rightarrow \ int\text{-}or\text{-}sol)
\end{array}$$

Removing unused rules

Lastly, we remove rules that have an intermediate hypothesis as its *then*-part, which is not used in any *if*-part:

$$\begin{array}{l}
\exists r_i : (if\text{-part}_1 \rightarrow int_1) \\
\neg\exists r_j : (int_1 \wedge if\text{-part}_2 \rightarrow int\text{-}or\text{-}sol) \qquad \Rightarrow \\
\qquad r_i : (if\text{-part}_1 \rightarrow int_1) \ \hookrightarrow \ \emptyset
\end{array}$$

Summary

The main difficulty in validation of AI systems is that the target domain is neither directly accessible nor there is a commonly accepted formal description of it. Thus, the only way to validate these systems is to confront the system with representative test cases and to compare the system's answers with the answers of human experts.

The heart of the methodology is a TURING test-like technique that systematically interrogates the system through test data. The present paper outlines ideas of

1. generating useful test cases,
2. a TURING Test experimentation,
3. evaluating the experimentation results,
4. expressing validity according to its purpose, and
5. system refinement.

These ideas refer to rule-based systems, which is the most commonly used kind of AI system in real world applications.

Besides providing an overview on the entire methodology and some hints for where to find more detailed descriptions of the "old steps", which are published in earlier papers by the authors, the new items here are (1) the different ways to express a system's validity according to the purpose of the validity statement and (2) the formal system refinement, which leads to a more valid system after passing the "validation loop" of figure 1. Thus, we can proclaim, that the loop is closed.

References

Abel, T.; Knauf, R.; Gonzalez, A.J. 1996. Generation of a minimal set of test cases that is functionally

Table 1: Reduction rules to construct "better rules" systematically

Reduction rules

R1
- $pos \in Pos$, s_{pos} has a finite value set with no well-defined $\leq$ relation
- there is no ordering relation $\leq$ among the values of s_{pos}
- $\{s^1_{pos}, \ldots, s^m_{pos}\}$ are the values of s_{pos} occuring in T^s_l $\Rightarrow$

$$[T^s_l, Pos, \{p_1, \ldots, p_n\}] \hookrightarrow$$

$$\textbf{1.}\ [T^{k,1}_l \setminus \{t_j \in T^s_l : s_{pos} \neq s^1_{pos}\}, Pos \setminus \{pos\}, \bigcup_{j=1}^{m} \bigcup_{i=1}^{n} (p_i \cup \{(s_{pos} = s^1_{pos})\})]$$

$$\textbf{2.}\ [T^{k,2}_l \setminus \{t_j \in T^s_l : s_{pos} \neq s^2_{pos}\}, Pos \setminus \{pos\}, \bigcup_{j=1}^{m} \bigcup_{i=1}^{n} (p_i \cup \{(s_{pos} = s^2_{pos})\})]$$

• • •

$$\textbf{m.}\ [T^{k,m}_l \setminus \{t_j \in T^s_l : s_{pos} \neq s^m_{pos}\}, Pos \setminus \{pos\}, \bigcup_{j=1}^{m} \bigcup_{i=1}^{n} (p_i \cup \{(s_{pos} = s^m_{pos})\})]$$

Contniue with each $T^{k,i}_l$ $(1 \leq i \leq m)$ separately.

R2
- $pos \in Pos$, s_{pos} has a value set with a well-defined $\leq$-relation
- s^{min}_{pos} is the smallest value of s_{pos} within T^s_l
- s^{max}_{pos} is the largest value of s_{pos} within T^s_l $\Rightarrow$

$$[T^s_l, Pos, \{p_1, \ldots, p_n\}] \hookrightarrow$$

$$[T^s_l, Pos \setminus \{pos\}, \bigcup_{i=1}^{n} (p_i \cup \{(s_{pos} \geq s^{min}_{pos}), (s_{pos} \leq s^{max}_{pos})\} \cup S_{excl})]$$

S_{excl} is the set of excluded values for s_{pos}, which have to mapped to a solution different from sol_k because of belonging to some other T^v_u with $v \neq k$:

$$S_{excl} = \{ \ (s_{pos} \neq s^j_{pos}) : \exists [t_j, sol_s] \in T^s_l \exists [t_m, sol_v] \in T^v_u (v \neq s) \text{ with } \forall p \neq pos(s^j_p = s^m_p) \ \text{ and } \ s^{min}_{pos} < s^m_{pos} < s^{max}_{pos} \ \}$$

equivalent to an exhaustive set, for use in knowledge-based system validation. In: *Proc. 9th International Florida Artificial Intelligence Research Symposium (FLAIRS-96)*, Key West, FL, USA, May 1996, pp. 280-284. Florida AI Research Society, 1996.

Abel, T.; Gonzalez, A.J. 1997. Utilizing criteria to reduce a set of test cases for expert system validation In: *Proc. 10th International Florida Artificial Intelligence Research Society Conference (FLAIRS-97)*, Daytona Beach, FL, USA, May 1997, pp. 402-406. Florida AI Research Society, 1997.

Jantke, K.P.; Abel, T.; Knauf, R. 1997. Fundamentals of a turing test approach to validation. Intelligent Meme Report MEME-IMP-1/1997. Sapporo, Japan, Hokkaido University, 1997.

Jantke, K.P.; Knauf, R.; Abel, T. 1997. The Turing test approach to validation. In: *Proc. Workshop on Validation, Verification & Refinement of AI Systems and Subsystems (W32)* of the *Int. Joint Conference on Artificial Intelligence (IJCAI-97)*, Nagoya, Japan, Aug. 1997, pp. 35-45. 1997.

Knauf, R.; Abel, T.; Jantke, K.P.; Gonzalez, A.J. 1998. A framework for validation of knowledge-based systems. In: *Proc. Int. Workshop on Aspects of Intelligent Systems Validation*, Ilmenau, Germany, Jan. 1998. Technical Report MEME-MMM-98-2, Sapporo, Japan, 1998, pp. 1-19. Hokkaido University, 1998.

Knauf, R.; Jantke, K.P.; Gonzalez, A.J.; Philippow, I. 1998. Fundamental considerations of competence assessment for validation. In: *Proc. 11th International Florida Artificial Intelligence Society Conference (FLAIRS-98)*, Sanibel Island, FL, USA, May 1998, pp. 457-461. AAAI Press, 1998.

O'Keefe, R.M.; O'Leary, D.E. 1993. Expert system verification and validation: A survey and tutorial. In: Artificial Intelligence Review. 7(1993):3-42