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

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

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

$\text{competence} = \text{consistency} + \text{completeness}$

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

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

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

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 ($R = E_1, \ldots, E_n$) and another one is between the experts' knowledge and the system's specification, which is (in case of successful verification) equivalent to $S$ ($E_1, \ldots, E_n = S$).

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 $E_{n+1}$.

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 $t_j$; (2) the expert’s behavior while rating the test case solutions; 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 $t_j$:

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

$\text{cpt}(e_i, t_j)$ 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

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

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 $T_k = \{|t_j, sol_k| \in \text{ReST} : t_j \in \pi_{\text{inp}}(\text{ReST}), [t_j, sol_k] \in E_{n+1}\}$.

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 $r_i$ is used for:

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

with $T_i = \{|t_j, sol_k| \in \text{ReST} : t_j \in \pi_{\text{inp}}(\text{ReST}), [t_j, sol_k] \in E_{n+1}, r_i \text{ is used for } t_j\}$

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 $v(r_i)$ for each rule.
- There is a set $T^*_r$ containing all test cases with test data parts occurring in $T_i$ 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 $en+1$:

$$T^*_r = \{|t_j, sol(e_i, t_j) : \exists [t_j, sol_k] \in T_i\}$$

- $T^*_r$ is splitted according to the different solution parts $sol_1, \ldots, sol_p, \ldots, sol_m$ of its elements. This leads to $m$ disjunctive subsets $T^*_{r1}, \ldots, T^*_{rp}, \ldots, T^*_{rm}$. 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 $v(r_i, sol_p)$ (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 $T^*_{rp}$ – can be computed:

$$v(r_i, sol_p) = \frac{1}{|T^*_{rp}|} \sum_{[t_j,sol_p] \in T^*_{rp}} \frac{1}{\sum_{i=1}^{n} (cpt(e_i, t_j) \cdot c_{ij})} \sum_{i=1}^{n} (cpt(e_i, t_j) \cdot c_{ij} \cdot r_{ij})$$

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 $sol_p = sol(e_i, t_j)$ of the test cases in $T^*_{rp}$, and $l$ indicates $\text{Since sol}_p = sol(e_i, t_j)$, $p$ occurs in the right hand side of the equation as the combination of $i$ and $j$.  

---

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 $t_j$?
the rule, from which \( T_i^* \) is originated.

- There is an “optimal validity” of rule \( r_i \), which is the maximum of all \( v(r_i, sol_k) \) among all solutions \( sol_k \) occurring in \( T_i^* \). The solution, to which this maximum is associated, is called the optimal solution \( sol_{opt} \) for \( r_i \).

\[
v_{opt}(r_i, sol_{opt}) = \max\{v(r_i, sol_1), \ldots, v(r_i, sol_m)\}
\]

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

In case \( v_{opt}(r_i, sol_{opt}) > v(r_i) \) there is at least one solution within \( T_i^* \), which obtained better marks by the experts than the system’s solution. In this case \( r_i \) 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 \( T_i \) of a guilty rule \( r_i \) 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, \( T_i \) of a “guilty rule” has to be splitted into subsets \( T_i^* \subseteq T_i \) according to their optimal solution \( sol_k \).

The new if-part(s) of the new rule(s) instead of a remaining and compiled “guilty rule” \( r_i \) are expressions \( e_i \in \mathcal{E}^2 \) of a set of \( p \) new alternative rules \( \{r_i^1, r_i^2, \ldots, r_i^p\} \) for each \( T_i^* \) and will be noted as a set of sets

\[
P_i^* = \{(e_1^1, \ldots, e_{p_1}^1), (e_1^2, \ldots, e_{p_2}^2), \ldots, (e_1^p, \ldots, e_{p_p}^p)\}
\]

here. The corresponding rule set of \( P_i^* \) is

\[
r_i^1 : \bigwedge_{i=1}^{p_1} e_i^1 \rightarrow sol_s \quad \ldots \quad r_i^p : \bigwedge_{i=1}^{p_p} e_i^p \rightarrow sol_s
\]

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

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

\[
P_i^* = \{(s_1 = s_1^{ident}, \ldots, s_q = s_q^{ident})\}
\]

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

\[
Pos = \{s_i : \neg \exists(s_i = s_i^{ident}) \in P_i^*\}
\]

Table 1 shows the rules used to reconstruct the remaining “guilty rules”. The reduction terminates, if the situation \( [T_i^*, \emptyset, P_i^*] \) is reached.

**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:

\[
\exists r_i : (if-part_1 \rightarrow int_1)
\]

\[
\exists r_j : (if-part_1 \land if-part_2 \rightarrow int-or-sol) \Rightarrow
\]

\[
r_j : (if-part_1 \land if-part_2 \rightarrow \text{int-or-sol}) \leftarrow
\]

\[
(int_1 \land if-part_2 \rightarrow \text{int-or-sol})
\]

**Removing unused rules**

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

\[
\exists r_i : (if-part_1 \rightarrow int_1)
\]

\[
\neg \exists r_j : (int_1 \land if-part_2 \rightarrow \text{int-or-sol}) \Rightarrow
\]

\[
r_i : (if-part_1 \rightarrow int_1) \leftarrow \emptyset
\]

**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

<table>
<thead>
<tr>
<th>Reduction rules</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>R1</strong></td>
</tr>
<tr>
<td>• pos ∈ Pos, s_{pos} has a finite value set with no well-defined ≤ relation</td>
</tr>
<tr>
<td>• there is no ordering relation ≤ among the values of s_{pos}</td>
</tr>
<tr>
<td>• {s_{pos}^{1}, ..., s_{pos}^{m}} are the values of s_{pos} occurring in T'_i ⇒</td>
</tr>
<tr>
<td>[ T'_i, Pos, {p_1, ..., p_n} ] ←</td>
</tr>
<tr>
<td>1. [ T'<em>{1,1} \setminus {t_j \in T'<em>i : \text{s</em>{pos} \neq s</em>{pos}^{1}} }, Pos \setminus {pos}, \bigcup_{j=1}^{m} \bigcup_{i=1}^{n} (p_i \cup {(s_{pos} = s_{pos}^{1})}) ]</td>
</tr>
<tr>
<td>2. [ T'<em>{1,2} \setminus {t_j \in T'<em>i : \text{s</em>{pos} \neq s</em>{pos}^{2}} }, Pos \setminus {pos}, \bigcup_{j=1}^{m} \bigcup_{i=1}^{n} (p_i \cup {(s_{pos} = s_{pos}^{2})}) ]</td>
</tr>
<tr>
<td>• • •</td>
</tr>
<tr>
<td>m. [ T'<em>{1,m} \setminus {t_j \in T'<em>i : \text{s</em>{pos} \neq s</em>{pos}^{m}} }, Pos \setminus {pos}, \bigcup_{j=1}^{m} \bigcup_{i=1}^{n} (p_i \cup {(s_{pos} = s_{pos}^{m})}) ]</td>
</tr>
</tbody>
</table>

Continue with each T'_{1,i} (1 ≤ i ≤ m) separately.

| **R2**          |
| • pos ∈ Pos, s_{pos} has a value set with a well-defined ≤-relation |
| • s_{pos}^{min} is the smallest value of s_{pos} within T'_i |
| • s_{pos}^{max} is the largest value of s_{pos} within T'_i ⇒ |
| \[ T'_i, Pos, \{p_1, ..., p_n\} \] ← |
| \[ T'_i, Pos \setminus \{pos\}, \bigcup_{i=1}^{n} (p_i \cup \{(s_{pos} \geq s_{pos}^{min}), (s_{pos} \leq s_{pos}^{max})\}) \cup \{s_{pos} \notin \text{S_{exc}}\} \] |

S_{exc} is the set of excluded values for s_{pos}, which have to be mapped to a solution different from sol_k because of belonging to some other T'_u with v ≠ k:

\[ S_{exc} = \{ (s_{pos} \neq s_{pos}^{1}) : \exists [t_j, sol_k] \in T'_i \exists [t_m, sol_v] \in T'_u (v \neq s) \] with \[ \forall p \neq pos(s_{pos} = s_{pos}^{1}) \] and \[ s_{pos}^{min} < s_{pos} < s_{pos}^{max} \]

