Under consideration for publication in Knowledge and Information Systems

A Quantitative Analysis of the Robustness of Knowledge-Based Systems Through Degradation Studies

Perry Groot, Annette ten Teije, and Frank van Harmelen

Division of Mathematics and Computer Science, Faculty of Sciences, Vrije Universiteit, Amsterdam, The Netherlands

Abstract. The overall aim of this paper is to provide a general setting for quantitative quality measures of Knowledge-Based System behavior which is widely applicable to many Knowledge-Based Systems. We propose a general approach that we call 'degradation studies': an analysis of how system output changes as a function of degrading system input, such as incomplete or incorrect data or knowledge.

To show the feasibility of our approach, we have applied it in a case study. We have taken a large and realistic vegetation-classification system, and have analyzed its behavior under various varieties of incomplete and incorrect input. This case study shows that degradation studies can reveal interesting and surprising properties of the system under study.

Keywords: Quantitative analysis; Knowledge-Based Systems; Robust behavior; Validation

1. Motivation

When asked about the essential differences between Knowledge-Based Systems (KBSs) and 'conventional software', one often hears the claim that KBSs can deal with incomplete, incorrect and uncertain knowledge and data, whereas conventional software is typically very brittle in these respects (see e.g., (Hayes-Roth 1984) for a very early formulation of this claim). Although, nowadays researchers no longer view this distinction as either necessary or sufficient to define a KBS, it is believed that the ability of KBSs to deal with missing or invalid data is an essential dimension of KBS validation.

There has been both practical experience and theoretical analysis over many years to back up the mentioned claim. As an example of practical experience, (Preece et al 1997) reports that in a number of verification exercises, errors were found in the knowledge-base of KBSs which were nevertheless still functioning at acceptable levels. As an example of theoretical analysis, (ten Teije and van Harmelen 1996) and (ten Teije and van Harmelen 1997) prove that for a large class of diagnostic systems the computed set of

diagnoses degrades gracefully and predictably when either the system input (observations) or the knowledge-base degrades in quality.

However, until now, the analysis of the robustness of KBSs in the face of incomplete, incorrect or uncertain knowledge and data has been limited to such practical experience and qualitative analysis. Little or no attempt has been made at a quantitative analysis of the proclaimed robustness of KBSs. A recent special issue of a journal was dedicated to methods for evaluating KBSs (Menzies and van Harmelen 1999). None of the papers in that special issue performed any quantitative analysis on the quality of KBSs. The editorial of this special issue lists only a hand-full of quantitative evaluation studies that have been performed over a decade or more of KBS research. In fact, one paper in that special issue (Shadbolt et al 1999) even seems to suggest that global qualitative evaluations are about as much as we can expect from KBS evaluation projects. Finally, one of the reviewers of this paper even remarked: "for a long time, the KA community has decried the lack of good evaluation metrics to measure the quality of the KA process and of the resulting knowledge bases." We consider this a serious defect in the study of KBSs, particularly since such robustness is often proclaimed as a unique characteristic of KBSs.

The aim of this research is to show that *a quantitative analysis of the robustness of KBSs is both possible and useful.* To argue this claim, we present a case study in which we perform such a quantitative analysis for a particular KBS. In Section 2 we describe our approach to measuring robustness by degradation studies, and we give definitions for the basic notions involved in such degradation studies. In subsequent sections, we apply this approach in a case study. Section 3 describes the KBS which we subjected to a degradation study. Section 4 gives an overview of the degradation studies we performed. Thereafter, Section 5 reports our robustness results with respect to the data input and Section 6 the robustness results with respect to the knowledge base used. The results obtained will be analyzed in these two sections. Finally, Section 7 summarizes the main points of the paper and in Section 8 we look at future steps to be taken.

2. Approach and Foundational Definitions

In this section we describe our approach to measuring robustness by degradation studies, and we give definitions for the basic notions involved in such degradation studies. Our aim is to define a very general set of notions that can be widely used in future degradation studies. We regard this section as the central contribution of this research: the definitions in this section should form the basis of similar analyzes by other researchers and practitioners.

2.1. Robustness and Degradation

The IEEE Standard Glossary of Software Engineering Terminology (IEEE 1990) gives the following definition for robustness:

Informal Definition 2.1 (Robustness). The degree to which a system or component can function correctly in the presence of invalid inputs or stressful environmental conditions.

In other words, robustness of a KBS is concerned with the way in which the quality of the KBS output degrades as a function of a decrease in the quality of the KBS input. This definition immediately leads to the idea of degradation studies:

Informal Definition 2.2 (Degradation Study). In a *degradation study* we gradually decrease the quality of the KBS input, and measure how the KBS output quality changes as a result.

This informal definition contains the concepts of 'KBS input (quality)' and 'output quality' which we will discuss in more detail in the Sections 2.2 and 2.3.

2.2. Output Quality

Of course, we must be more precise about the rather vague notion of 'quality' of the KBS input and output. Concerning the KBS output, we assume that this is always a *set* of answers. In fact, for many typical KBS tasks, this is a realistic assumption: a set of consistent classes in a classification task, a set of likely hypotheses in a diagnostic task, a set of potential designs in a configuration task, etc.[1] More explicitly stated:

Assumption 2.1. For the KBSs that we consider we assume that their output can be interpreted as a discrete set of answers.

Under this assumption, we define two measures for KBS output quality. Let *correct(I)* be the set of all correct answers for a given input I, and *output(I)* be the set of actually computed answers for a given input I.

Definition 2.1 (Recall). The *recall(I)* of a KBS for a given input I is defined as:

$$recall(I) = \frac{|correct(I) \cap output(I)|}{|correct(I)|}.$$

In other words: the recall is the fraction of correct answers that the system actually computes. It can of course happen that $correct(I) = \varnothing$, i.e., there is no correct answer. For example, this happens when the system is presented with a case I for which no correct output exists, such as an inconsistent set of observations for a classification system, or an inconsistent set of requirements for a design system. In this case we define:

$$\text{if } correct(I) = \varnothing \text{ then } recall(I) = 1,$$

as all correct answers are recalled by the system.

Definition 2.2 (Precision). The *precision(I)* of a KBS for a given input I is defined as:

$$precision(I) = \frac{|correct(I) \cap output(I)|}{|output(I)|}.$$

In other words: the precision is the fraction of computed answers that are actually correct. In the case $output(I) = \varnothing$ (i.e., the system returns no output) we define:

$$\text{if } output(I) = \varnothing \text{ then } precision(I) = \begin{cases} 1 & \text{if } correct(I) = \varnothing, \\ 0 & \text{otherwise.} \end{cases}$$

This reflects the intuition that the only correct output in this case is no output at all.

Ideally, both recall and precision would be 1 (i.e., the system returns all and only correct answers), but in practice they are antagonistic: a higher precision (recall) is usually paid for with a lower recall (precision).

[1] Of course, even for KBSs which return a single answer, this assumption still holds since we can interpret such a single answer x simply as the singleton set $\{x\}$.

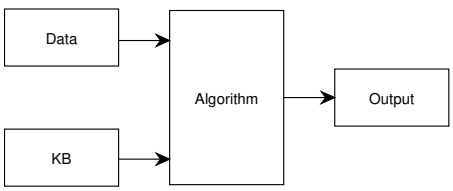


Fig. 1. Knowledge-based system components.

There are two attractive aspects to these definitions for measuring the output quality. First of all, these definitions are well known from the literature on information retrieval (e.g., (Salton and McGill 1983)) and have proven to be useful, informative, and intuitive measures in many studies in that field and elsewhere (see for instance (Schumann and Fischer 1997) for an application of these measures to deduction-based software component retrieval). Secondly, these measures are completely general and make no commitment to either the task or the domain of the KBS that we wish to study. Consequently, the approach proposed can be directly applied to other KBSs, even when they are very different from the one that we happen to have chosen in our own case study.

The above definitions are in fact gradual versions of the classical notions of soundness and completeness: recall corresponds to the degree of completeness of the system, and precision corresponds to the degree of soundness of the system. These measures provide a quantitative angle on earlier work by (van Harmelen and ten Teije 1998) which was strictly qualitative.

2.3. Knowledge-Base System Input

We can represent a KBS graphically as in Figure 1: the data, the knowledge base, the algorithm and the output. By 'KBS input' we mean the data given for example by a user as well as the knowledge base used by the system. The degradation experiments we propose for measuring the robustness of the system will either be with respect to the quality of the data input or with respect to the quality of the knowledge base.

For the input quality measure we give two general aspects to consider for measurement. These aspects are incompleteness or incorrectness of the data or knowledge. Although these aspects are quite general, they may not be applicable to every knowledge base. The aspects proposed also do not cover all possible aspects one might want to measure. More explicitly formulated, with incompleteness and incorrectness we mean the following:

Incomplete: A part of the data input or part of the knowledge used is missing from the KBS input. For example, the data input could be a number of observations. Some of the observations might be to hard or to expensive to obtain. A part of the knowledge might be missing because the knowledge was obtained from human experts who forgot to provide it.

Incorrect: In contrast with incomplete data or knowledge, a certain part of the data or knowledge *is* provided by a user or expert. However, it is incorrectly represented. For example, incorrect data could be caused by a user who has made a wrong observation. Incorrect knowledge could for example be caused by faulty knowledge of an expert, misunderstandings when coding the knowledge into the system, or limitations of the representation used by the system.

These two aspects are important in KBS validation. For example, a knowledge base will most likely be incomplete and often partially incorrect. Hence, to analyze the robustness of a knowledge base with respect to incompleteness or incorrectness is a realistic and important issue.

Notice that we have already taken a step further than (van Harmelen and ten Teije 1998). In that paper, the authors did not commit to any definition of quality on input or output, and only demanded that whatever the definition was, it should respect a partial ordering. In our approach with degradation testing, we commit to a specific definition of output quality, while leaving input quality open to be defined for each specific application.

2.4. Comparing Robustness

The only notion that is still left undefined is some ordering on robustness: when do we call a system more robust or less robust than another? Unlike output quality (where we have given a single widely applicable definition) and input quality (whose definition is deliberately left open to depend on the task-type), we have not been able to determine a good answer to this question.

When input quality decreases, a system that produces an output with a fluctuating quality is much less predictable than a system that produces an output with a monotonically decreasing quality. We therefore demand that any system which is called robust at least produces an output with monotonically decreasing quality as function of decreasing input quality.

Definition 2.3 (Monotonicity). A robust system will show a monotonically decreasing output quality as a function of deteriorating input quality.[2]

Note that this demand corresponds precisely to the usual demand on anytime algorithms that their output quality monotonically increases with increasing run-time (Dean and Boddy 1988). However, this demand is insufficient for ordering various systems according to their robustness. We therefore give additional competing definitions without choosing one over the other.

Definition 2.4 (Quality Value). A system S_1 is more robust than a system S_2 for a set of inputs, if everywhere on that input set the output quality of S_1 is higher than the output quality of S_2.

Definition 2.5 (Rate of Quality Change). A system S_1 is more robust than a system S_2 for a set of inputs, if everywhere on that input set the output quality of S_1 decreases more slowly than the output quality of S_2.

Definition 2.6 (Integral of Quality Value). A system S_1 is more robust than a system S_2 for a set of inputs, if on that input set the integral of the output quality of S_1 is larger than the same integral for S_2.

Formally speaking, Definition 2.4 compares the output quality of two systems (and is concerned with which system produces the *best* output), while Definition 2.5 compares the first derivative of the output quality of the systems (and is therefore concerned with which system produces the *most stable* output). Definition 2.6 compares the overall quality of the output quality over an entire interval even when neither system always

[2] In practise this demand may be a bit too strong. Systems which produce a more or less monotonic output can also be considered robust.

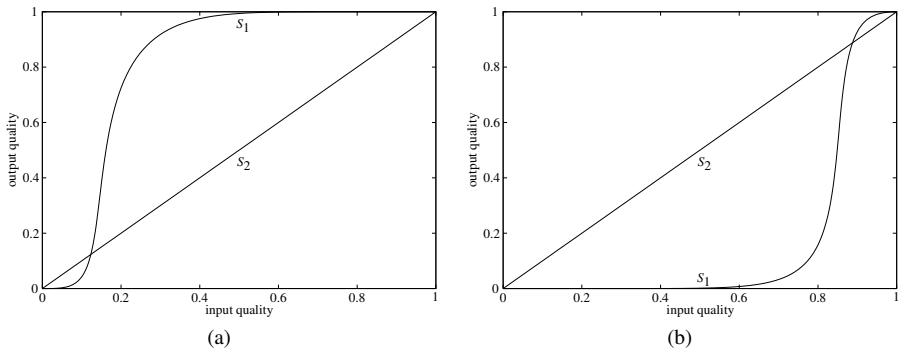


Fig. 2. Comparing robustness of systems.

dominates the other (as required in Definition 2.4). These definitions are illustrated in Figure 2.[3]

Note that the definitions do not have to be applied on the entire input quality range. Instead, they should be applied to an interval of interest. Usually this will be some interval including 1 (i.e., $[x, 1]$).

In Figure 2(a), system S_1 is more robust on for example the interval [0.1, 0.3] according to Definition 2.4, since on that interval its output quality is always higher than that of S_2. However, according to Definition 2.5, S_2 is the more robust of the two, since its output quality decreases more gradually (reading the graph from right to left). Definition 2.6 allows to take a more overall perspective: it takes the size of the area under the output quality graph as measure of the overall quality. Under this definition, S_1 is more robust on the entire interval [0,1], since the value of $\int_0^1 output\ quality\ d(input\ quality)$ is larger for S_1 than for S_2. Note that the situation is rather different in Figure 2(b). Although the output quality again increases monotonically from 0 to 1 over the same interval, the comparisons between S_1 and S_2 on various subintervals are very different. Now, for example on the interval [0.1, 0.3], S_2 is more robust than S_1 when using Definition 2.4, but S_1 is more robust than S_2 when using Definition 2.5.

At the current point in our research, we simply propose each of these definitions as reasonable, without claiming superiority of any definition in all cases. In fact, we believe that under different pragmatic circumstances, different definitions will be preferable: if steep drops in system performance are to be avoided, the second definition is preferable. If one is interested in upholding output quality as long as possible in the face of declining input, the other two definitions may be preferred.

3. Example Knowledge-Based System

For our case study we have used a classification system for commonly occurring vegetation in Southern Germany. The plant-classification system was created with the D3 Shell-Kit which is a tool for the development of KBSs. We will not discuss this tool here but refer to a number of publications about D3 (Puppe et al 1996, Puppe et al

[3] In our figures, we plot input quality against output quality. When speaking about 'robustness', we are interested in *decreasing* input quality, so the graphs must be read from right to left.

1994). It is also possible to download a demo-version of the software from the URL `http://d3.informatik.uni-wuerzburg.de`.

The plant-classification system that we studied can have 40 different observables as input and has 93 different plant names as output. The knowledge base consists of 7586 rules. Furthermore, with the system we received 150 test cases. Each of these cases consisted of the set of observations for that case (color and shape of flowers, leafs, stem, etc.), together with the (supposedly correct) answer for these observations as given by a human expert. Around 97% could be answered correctly by the system.

The input observations can be entered in a graphical user interface, but the user is not restricted to the ordering in this interface. The observations can be entered in any order, thus the input can be seen as a *set*. This is not entirely true because some observations are dependent on other observations and will only appear when certain input-conditions are met. Because of these dependencies, the maximum number of observations that can be given for one case is 30.

For our degradation tests we translated the plant classification system into Prolog code. This resulted in a knowledge base with 11724 rules all with the following representation:

```
kb(Plant, Observation, Value, Score).
```

Each time a new observation is given to the system, all rules are collected containing the same `Observation` and `Value`. For each of these rules the score of the `Plant` mentioned in the rule is adjusted by adding the `Score` to its current score. When the score crosses a threshold it is outputted by the system.

In fact the rules do not actually contain numerical scores but descriptions, which were used to make it easier for the experts to express their knowledge. The descriptions are however translated to numerical scores whenever they are used by the system. We will therefore use the descriptions as if they are numbers. The descriptions range from P_1 through P_6 for positive scores and N_1 through N_6 for the negative scores. Furthermore, $|Pi| = |Ni|$ for $i = 1,\ldots,6$ and $P_{i+1} = 2 * P_i$ for $i = 1,\ldots,5$.

4. The Degradation Studies

In the following sections we will report our robustness analysis of the plant classification system. In Section 5 we will analyze the robustness with respect to the data input, while in Section 6 we will analyze the robustness with respect to the knowledge base used.

Before discussing the results in more detail we emphasize that the case study only serves to illustrate our proposal to analyze the robustness of KBSs through degradation studies. The important aspects of this case study are the quantities measured and how they were analyzed, not the obtained robustness results of the plant classification system.

Before we discuss the results, a final remark must be made about the possible values of recall and precision in this case study. Since for every case there is at most one correct answer (namely the name of the actual plant on which the observations were made), we have for any case I, $|correct(I)| = 1$ iff the case is in the knowledge base or $|correct(I)| = 0$ iff the case is not in the knowledge base. As a result, the only values that $recall(I)$ can assume are either 0 or 1. For the same reason, $precision(I)$ is either 0 or $1/|output(I)|$. However, we are not interested in the specific behavior of the system for a particular case, but in the average behavior of the system. Hence, we are interested in the average recall (i.e., the sum of all recall values divided by the number of cases used) and the average precision. We will represent these averages in our figures, but will use for example the term ‘recall’ when in fact we mean ‘average recall’.

5. Results of the Degradation Study – Data Input

In this section we present the robustness results with respect to the data input that we have obtained in empirical experiments with the plant-classification system. First, we will define the input quality measure we will use in Section 5.1. Thereafter we will give results using the ordering on the observables found in the test cases (Section 5.2). We analyze the effect of other orderings in Section 5.3. We give some conclusions in Section 5.4.

5.1. Which Input Quality Measure to use?

According to the definitions from Section 2 we must still decide on what to use as a measure on the input quality. In this case study we choose the *completeness of the input* as the measure of input quality. In our classification system, completeness of the input can be directly translated as the *number of available observations*.

There are two reasons why this choice is reasonable and attractive:

Robustness: in many practical classification settings, the input observations are not completely available. It then becomes an interesting question how robust the system functions under such incomplete input.

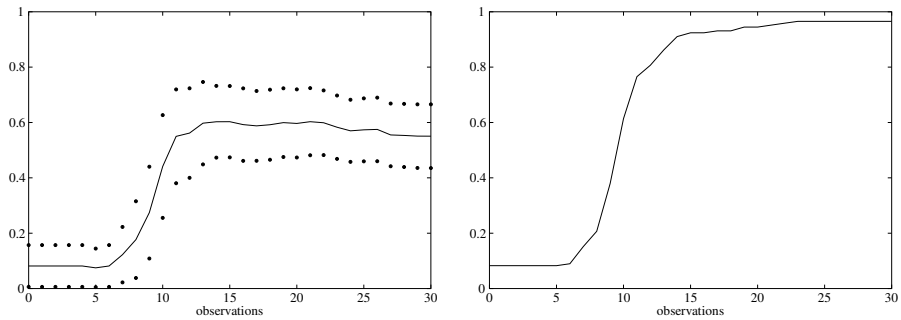
Anytime behavior: Even when all observations are present, there are practical settings where insufficient run-time is available to process all the observations: some output from the system is required before a given real-time deadline, and not all observations can be processed before this deadline. Those observations that could not be processed before the deadline can be regarded as 'missing from the input'.

As a result of this second reason, the degradation results that we present in this section can also be seen as *anytime performance profiles* for the plant-classification system. Performance profiles are a basic tool in the study of anytime algorithms (Dean and Boddy 1988). They plot the output quality as a function of available run-time. Since available run-time can be interpreted as one aspect of 'input quality', such performance profiles are simply a special case of our more general proposal: performance profiles only study output degradation as a function of decreased run-time, whereas our approach is applicable to any aspect of input quality that one chooses to model.

In the following we will present a number of graphs analyzing the robustness of the plant-classification system. Each of these graphs plot output-quality (measured by either recall or precision) against input-quality (measured by the number of observations that were available to the system). If one is interested in anytime behavior, these graphs can be read from left to right: "what happens when the system has time to process more and more of the inputs?" If one is interested in robustness, these graphs should be read from right to left: "what happens when the system is provided with fewer and fewer of the inputs?"

5.2. Using the Input Sequence from the Test-Cases

Now that we have established that the number of available observations will be the input-aspect that we will degrade in our studies, we have to decide in which order observations will be made available to the system. Our first choice is simply based on the order in which the observations appeared in the test-cases for the plant-classification system. Each such test-case consisted of a list of observables and their values for that



(a) Precision (b) Recall

Fig. 3. Measures with test-case order.

case. Figure 3(a) shows how the precision of the answers from the system (as defined in Definition 2.2) increases when longer initial-sequences of the test-cases were given to the system. The first surprise that this graph has in store for us is its monotonic growth:

Surprise 5.1. Both average precision and average recall (see Figure 3) grow monotonically (or: almost monotonically in the case of precision) when adding more observations. This is somewhat surprising since this is not true for individual cases.[4]

The classification algorithm of the plant-classification system assigns both positive and negative scores. This means that the answer-set can both grow and shrink when adding more observations. In fact, only 58% of the test-cases has a monotonically growing answer-set. As mentioned above, such monotonic behavior is desirable from both a robustness and from an anytime perspective, so on a case-by-case basis, the plant-classification system does not score very well on this. Surprisingly, the average-case behavior of the system is apparently much better.

A second observation to make is that (as to be expected) initially the system is not able to make any sensible guess at likely solutions (this holds up to about 6 observations). For higher number of available observations, the graph is surprising for two reasons:

Surprise 5.2. After about 12 observations, adding more observations does not increase the precision. This is surprising since most cases contain as much as 19-30 observations. Figure 3(a) suggests that 12 observations is sufficient to obtain the maximally achievable precision on average.

Surprise 5.3. The region in which additional observations actually contribute to an increase in precision is surprisingly small, namely between the 6 and 12 observations. Of the 19-30 observations per case, all observable changes to the output seem to be in this small segment of observations!

Figure 3(b) shows similar results for the other dimension of output quality, namely the recall from definition 2.1.

The dotted lines in Figure 3(a) indicate the variance of the precision, and this variance is rather significant. It shows that the distribution of the actual precision-values

[4] This result is not completely surprising because it is well known that the average of several variables can indeed show a different distribution than the individual variables.

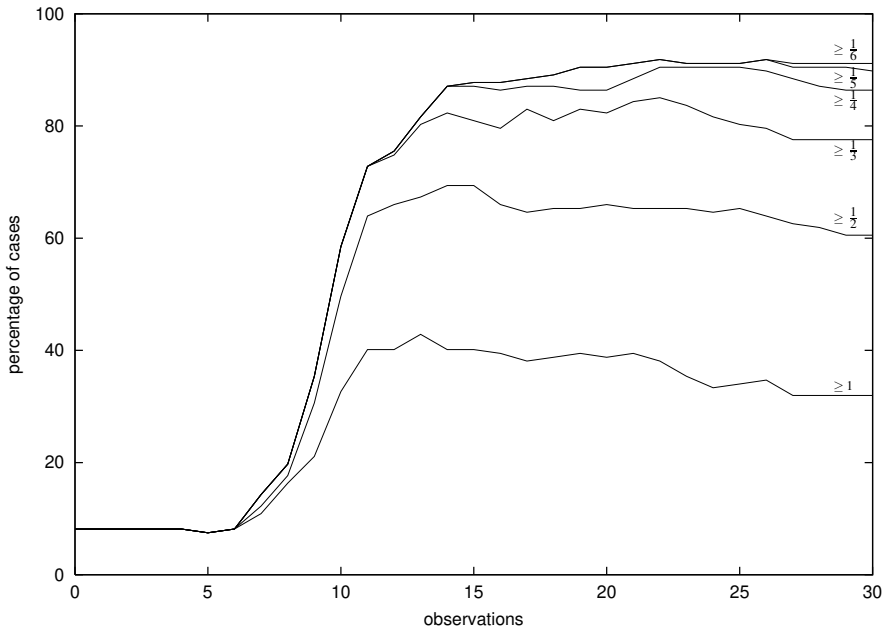


Fig. 4. Precision with multiple levels.

that were obtained for the different cases are actually spread rather widely around the average.[5]

Figure 4 gives more insight in the distribution of the precision than the simple average from figure 3(a). Each line in this figure shows the percentage of cases that achieved a precision of at least a certain value after the given number of observations. The lowest line shows that after 12 observations, 40% of the cases have already reached the maximum precision (namely 1). Furthermore, and more surprisingly, this percentage then stops growing! This means that:

Surprise 5.4. When aiming for the maximum precision of 1, there is no need to use any more than 12 observations (out of a maximum of 30!). If the maximum precision has not been reached after the first 12 observations, adding further observations will not help.

This is actually a more precise version of surprise 5.2 above. There we claimed that extending beyond 12 observations was not useful *on average*. Here we see that for harder cases, a few more observations do actually help, although not more than 20 observations in total.

This is because at the other end of the scale (the top line in figure 4) , we see that the percentage of cases with a precision of at least 0.2 continues to increase during a longer interval. Apparently, harder cases (those that ultimately achieve a lower precision) benefit more from additional observations than easy cases (those that achieve precision 1). Nevertheless, even there we see that no increase is gained after about 20 observations:

[5] No variance was plotted for the recall since, as explained above, in our application the recall is either 0 or 1. Because of this, the variance for recall is not a meaningful notion.

Surprise 5.5. Whatever the final precision that is ultimately obtained by the system, this level of precision is already obtained after at most 20 observations. It seems that asking for any more then 20 observations will not improve the output quality any further. This is surprising since many cases (in fact 98% of the test set) contain more than 20 observations.

Looking at the initial segment of observations, we see another surprise: although we may expect that a low number of observations leads to a low average precision, it is surprising that the lines for the different precision-levels all *coincide* until the 6th observation:

Surprise 5.6. No increase in precision can be gained from the first 6 observations.

This means that in an anytime setting, interrupting the system before the 6th observation is completely useless, since no increase in precision will have been obtained yet.

Figure 4 is particularly interesting from an anytime perspective: it tells us for each partially processed input what the chance is that the system has already obtained a certain precision in its output: for instance, after having fed the system 10 observations, there is a 30% chance that it has already obtained the maximum precision of 1, a 45% chance that it has already obtained a precision of at least 0.5, and a 60% chance that it has already obtained a precision of at least 0.3.[6] This information can be used by the user to determine if it is useful to continue feeding the system more input, or if a sufficiently high precision has already been obtained for the purposes of the user, so that processing (and acquiring potentially expensive observations) can be stopped. Our graph (when interpreted as a performance profile) contains much more information than the usual performance profiles presented in the literature (e.g., (Zilberstein S 1996)). These graphs typically give only a *single* expected value for the output quality at any point in time (compare our Figure 3(a)), whereas we give a probability distribution of the expected output value, which is much more informative.

Note that the ideas behind Figure 4 can in principle also be applied to the recall. We omitted this because in our case study the recall is either 0 or 1, thus the resulting figure would be the same as Figure 3(b).

5.3. Exploring Other Input Sequences

In the profiles sofar, we have degraded the input by removing observations in the order in which they were listed in each test-case. Similar degradation experiments were performed using different orderings which are reported in Figure 5.

For reference, the dotted line shows the recall-profile from Figure 3(b). The left-most line shows the theoretically optimal average-recall profile: at each step in each case, we computed which next observation would contribute maximally to an increase in recall. This computation can only be done theoretically for test-cases where all observations are already present. The right-most line shows the slowest average-recall profile. Note that every other recall profile must lie between these two lines. Figure 5 also shows a bundle of lines. Each line corresponds with a random ordering. Finally, we also performed experiments with the ordering shown by the graphical user interface (GUI), which is represented by the line close to the dotted line. These experiments lead us to the following observation:

[6] Note that these chances are cumulative, which is why they add up to more than 100%.

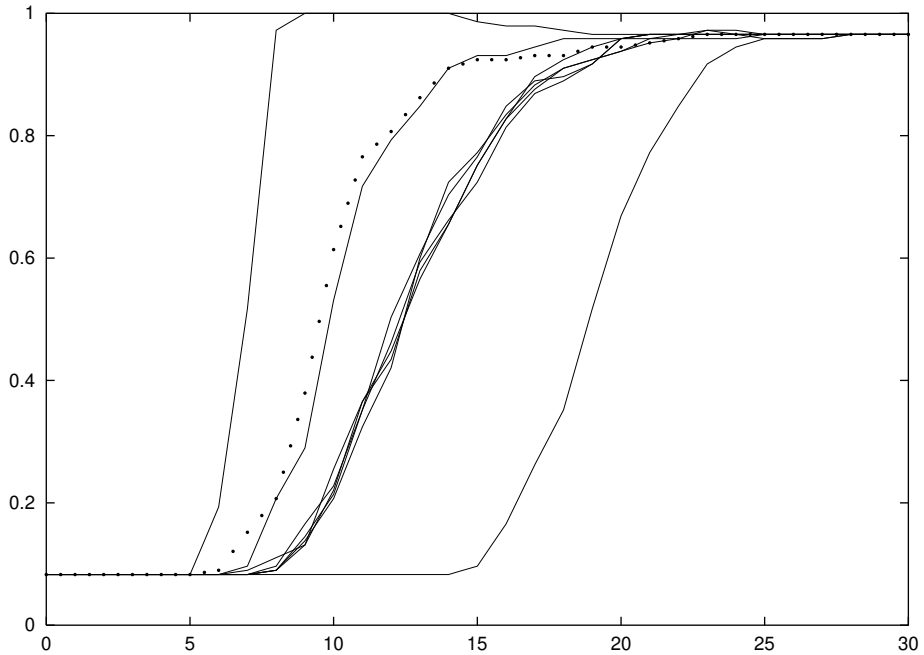


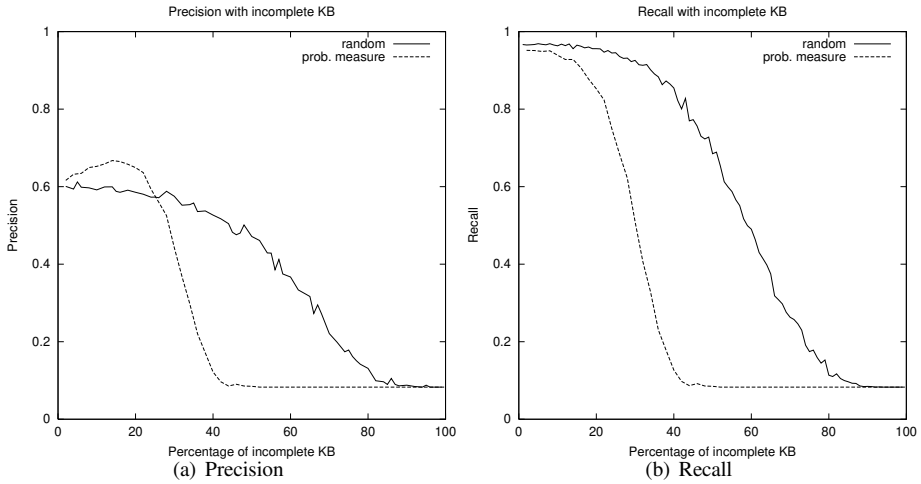
Fig. 5. Recall with various orderings.

Surprise 5.7. The degradation sequence taken from the test-cases and GUI is surprisingly effective in obtaining a high recall after only a few observations. In fact, it is much closer to the theoretically optimal sequence than the randomly generated (information-free) sequences.

5.4. Preliminary Conclusions on Robustness

From the first experiment in which we used the test-case ordering its clear that the robustness of the plant classification system in not very uniform across the distribution of input quality. When degrading the input quality, the system first appears extremely stable agains missing obervations as no quality loss occurs at all. This holds until about 12-15 observation. At that point the robustness of the system is very low and the input quality drops dramatically.

The experiments with different orderings showed that the plant classification system is very sensitive to the specific order in which the observations are presented to the system, which had effect on the robustness profile obtained. This type of robustness is not covered by our definitions in Section 2.4. The definitions there are all concerned with comparing the behavior of different systems on the same degrading input. The phenomenon observed in Figure 5 concerns the behavior of a single system on different ways of degrading the input. We leave it for further research how to include this type of robustness in our approach.



(a) Precision (b) Recall

Fig. 6. Recall and precision with incomplete knowledge base.

6. Results of the Degradation Study – Knowledge Base

The degradation experiments on data input already yielded interesting insights in the behavior of a realistic KBS. In this section we will perform the same kind of analysis with respect to the knowledge base used. For the input quality we will measure the incompleteness and incorrectness of the knowledge base. In Section 6.1 we will analyze robustness with respect to an incomplete knowledge base, whereas in Section 6.2 we will analyze robustness with respect to an incorrect knowledge base.

6.1. Incomplete Knowledge

Our experiment is as follows: we select some percentage of rules at random from the knowledge base, remove the selected rules, and compute the recall and precision value for each case with the modified knowledge base. The average recall (i.e., the sum of all recall values divided by the number of cases) and the average precision will indicate the robustness of the system with respect to the percentage of rules removed. Of course, the experiment has to be repeated multiple times to obtain the robustness of the system on average. Note that as we are no experts ourselves in plant classification and therefore unable to check if some rules are missing from the knowledge base, we will call the provided plant classification system complete, although in reality it may not be.

The results of the experiments are reported in Figure 6(b) for the recall and in Figure 6(a) for the precision. For each percentage, the average recall and average precision was obtained over 10 different runs. Besides average values, the minimal and maximal obtained values are also plotted in both graphs.

Both graphs show an almost smooth decrease of our measured values. The graphs are surprising in that the measures start to decrease around 40% in both graphs.

Surprise 6.1. In the plant classification system almost 40% of the knowledge base can be removed at random without severe quality loss.

A possible cause for this decrease at 40% and not at some earlier point is that the average recall and precision are computed over 150 cases. When a rule is removed from

the knowledge base it will only have effect on a small number of cases and therefore only a small effect on the average recall and precision. When more rules are removed, more cases will be effected which results in a decreasing recall and precision.

This form of degradation experiment can lead to interesting insights into the robustness behavior of the system analyzed. Nevertheless, the experiment performed has a drawback. It is not realistic to assume that each element in the knowledge base has the same chance of being forgotten. It would be more realistic to remove parts of the knowledge base based on some probability measure. Several measures should be compared when it is not obvious which measure should be used.

In another experiment we used a probability measure instead of removing rules at random from the knowledge base. Our results with this probability measure are also in Figure 6. Obviously the results have changed dramatically. Whereas our previous results (also in Figure 6) showed that 40% of the knowledge base could be removed without severe quality loss, the new results show that when 40% of the knowledge base is removed in a biased way the system becomes useless. Furthermore, the drop in output quality already starts at 15% and is much steeper than before.

These new results should warn people before using the degradation studies for analyzing the robustness of KBSs. The experimental setup should be as realistic as possible and care has to be taken when interpreting the results as these can be sensitive to the chosen setup as we demonstrated with our case study.

6.2. Incorrect Knowledge

In our case study, the knowledge base consists of rules. Each rule consists of some observations, a name of a plant, and a score. To test the robustness with respect to incorrect knowledge the rules need to be modified in some way. However, we like to do this in a realistic setting. Modifying the observations at random is therefore no option. First of all some observations are usually answered right while others are prone to errors. Secondly, if an observation is wrong it is usually not done at random. Some answers lookalike while others are clearly distinct. Modifying the observations of the rules realistically therefore requires domain knowledge which we lack. However, modifying the score of a rule in our case study does not require any domain knowledge.

In our experiments with incorrect knowledge we address the question: “How robust is the system for incorrectly entered scores?”. Our experiment is therefore as follows: we select some percentage of rules at random from the knowledge base, modify the score of the selected rules, and compute the recall and precision value for each case with the modified knowledge base. The average recall (i.e., the sum of all recall values divided by the number of cases) and the average precision will indicate the robustness of the system with respect to the percentage of rules modified. Of course, the experiment has to be repeated multiple times to obtain the robustness of the system on average.

Only one question remains: “In what way do we modify the score of a rule?” We identified the following parameters:

Direction: Scores can be changed to a higher or lower value. Most likely scores are damaged in both directions. Nevertheless, we also investigate the effect of a biased expert, i.e., an expert who always assesses something too high (or too low). We will use the word ‘positive’ when scores are only changed to a higher value, and the word ‘negative’ when scores are only changed to a lower value. When both words do not occur, it means scores are changed in both directions.

Size: The scores of rules are changed from one class to another. Most likely the new

class is only one higher or one lower than the old class. We will call this kind of damage a 'near miss'. To be complete we also investigate the effect when the difference of the new class and old class is two classes. We will call this kind of damage an 'error'.

The parameters lead to six different experiments, but before discussing the actual results let us first consider what results are to be expected.

6.2.1. Hypotheses

The plant classification system we are analyzing uses a threshold and only gives plants with a score higher than this threshold as output. When we change the scores of rules to a higher value it follows that the final score of the plants can only increase. As the recall only measures if the correct answer is given as output it follows that the recall can never decrease. The recall will probably increase as more plants (including the correct answer) are now more likely to cross the threshold and be given as output.

Hypothesis 6.1. When the scores of rules are increased, the recall of the system will increase.

The converse also holds. When we change the scores of the rules to a lower value it follows that the recall can never increase. When the scores are decreased sufficiently the recall will decrease as outputted plants which are correct will drop below the threshold.

Hypothesis 6.2. When the scores of rules are decreased, the recall of the system will decrease.

Some hypotheses for the precision can also be given, but the argument is more complex. Considering the formula for the precision (Definition 2.2) we can identify two causes for an increase in precision:

I1. The score of the correct plant is increased and thereby crosses the threshold. In this case the precision changes from 0 to some positive value.

I2. The scores of one or more incorrect plants decrease and thereby drop below the threshold. If the correct answer remains in the output this will increase the precision from some positive value to a higher positive value. For example, let A and B be two plants given as output. Let A be the correct plant. In this case the system will have a precision of $1/2$. When the scores are changed such that B drops below the threshold while A remains above the threshold the precision will change to 1.

In a similar way we can identify two causes for a decrease in precision:

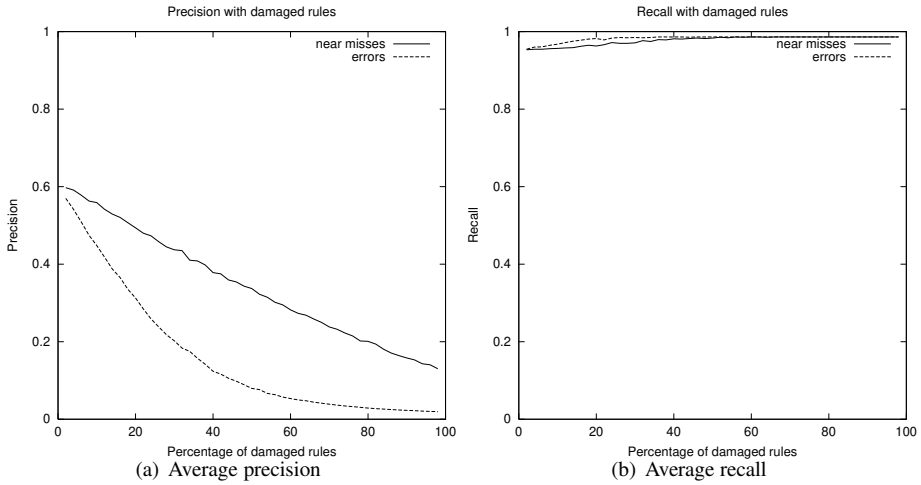
D1. The score of the correct plant drops below the threshold. The precision changes from a positive value to 0.

D2. The score of one or more incorrect plants increase and cross the threshold.

When we change the scores of rules to a higher value both I1 or D2 can occur. However, I1 is unlikely as previous experiments showed that the recall of the system is already close to the optimal value of 1. It follows that we expect a decrease in precision.

Hypothesis 6.3. When the scores of rules are increased, the precision of the system will decrease.

When we change the scores of rules to a lower value both I2 or D1 can occur. As these causes change the precision into two different directions it is not obvious to predict the outcome of the precision.



(a) Average precision (b) Average recall

Fig. 7. Positive near misses and positive errors.

The outcome is even harder to predict when we change the scores of rules both to lower values and to higher values as all four causes for the change in precision can occur. Also the recall can now both increase and decrease and is therefore harder to predict. Nevertheless, for the recall we know that the performance profile has to lie somewhere in between the results for the strictly positive and negative changes.

Hypothesis 6.4. When some rules are changed positively while other rules are changed negatively, the recall of the system will be between the recall values of the strictly positive and negative changes.

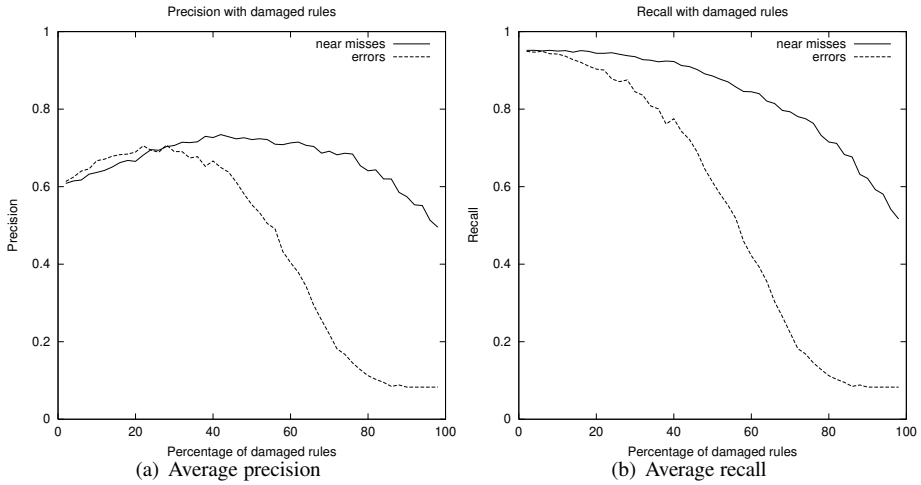
Besides direction of changes we will also look into the effect of the size of the changes. We expect that a larger chance will have a larger effect on the average values. For example, when the scores are changed positively and the recall increases with near misses, we expect the recall to increase even more when we experiment with positive errors.

Hypothesis 6.5. When comparing experiments in which we make errors instead of near misses without changing the direction of the changes, the effect observed in the near miss experiment will be made stronger.

6.2.2. Results

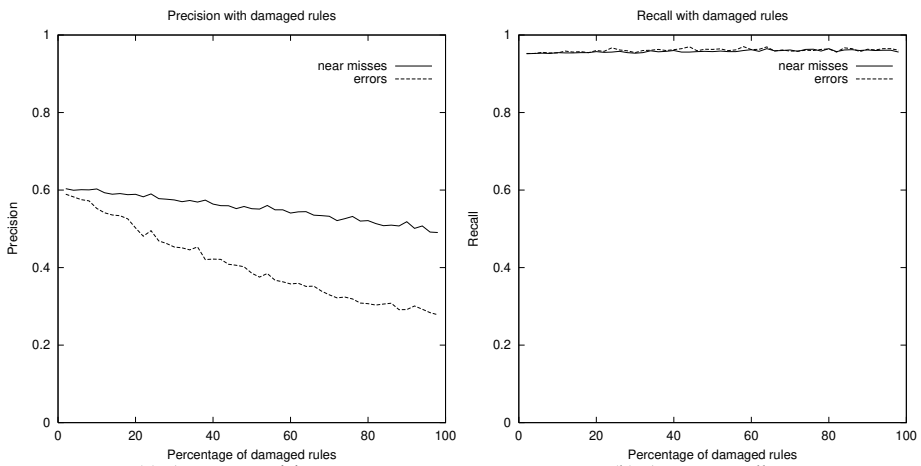
Now we will discuss the results which are shown in the Figures 7 through 9.

First note that our hypotheses hold in the shown figures. In case of positive changes the recall increases as is shown by Figure 7(b) (Hypothesis 6.1). In case of negative changes the recall decreases as is shown by Figure 8(b) (Hypothesis 6.2). In case of positive changes the precision decreases as is shown by Figures 7(a) (Hypothesis 6.3). In case of near misses, the recall (Figure 9(b)) will lie between the values found in the strictly positive experiment (Figure 7(b)) and the strictly negative experiment (Figure 8(b)). The same holds for the experiments with errors (Hypothesis 6.4). In case of positive changes, we find that the recall in the experiment with errors lies everywhere above the recall in the experiment with near misses (Figure 7(b)). In case of negative changes,



(a) Average precision (b) Average recall

Fig. 8. Negative near misses and negative errors.



(a) Average precision (b) Average recall

Fig. 9. Near misses and errors.

we find that the recall in the experiment with errors lies everywhere below the recall in the experiment with near misses (Figure 8(b)) (Hypothesis 6.5).

Other points of interest are the specific curves we obtained in our experiments. For example in case of positive near misses (Figure 7(a)) we find a *linear* decrease in precision whereas the decrease is *non-linear* when we increase the size of the changes.

In the case of negative near misses we could not predict the behavior of the precision as there are two causes that could change the precision in opposite directions (causes I2 and D2 discussed in Section 6.2.1). Figure 8(a) tells us that I2 happens more often than D1 in the first segment of the figure as the precision increases. Hence, in most cases first some incorrect plants drop below the threshold before the correct plant drops below the threshold. It follows that in most cases the correct plant has outscored most of the incorrect plants, which is desired behavior of the system.

The same curve as in negative near misses also occurs in the figure for negative errors (Figure 8), however, the curve of the negative near misses is 'compressed' into the first halve of the figure because the size of the changes has increased. Note that this holds for both the precision and the recall.

Besides interesting behavior we are of course interested in the robustness of the system. Probably the most realistic setting is the experiment with near misses (Figure 9). Both precision and recall show a very stable curve indicating a robust knowledge base for 'near misses'. Although the knowledge base appears to be robust on the entire range of the x-axis, from a practical point of view usually only the first part of the figure is important.

7. Conclusions

In this paper, we have argued for the need for *quantitative analysis* of the quality of KBSs. In particular, we have shown how *robust behavior* in the light of incomplete system-input as well as an incomplete and incorrect knowledge base is amenable to such quantitative analysis. Our quantitative analysis is based on the idea of *degradation studies*: analyze how the quality of the output changes as a function of degrading input. We have proposed a set of *general definitions* which are general enough that they can be used in similar degradation experiments by others, even if the systems concerned are of a very different nature than the one in our case study. We have shown the practicality of our approach by applying it to a particular *case study*. This yielded a number of surprising insights into the behavior of the system under study.

We believe that the following issues are the most important in our proposed approach of degradation studies for measuring the robustness of KBSs:

Generality: Degradation studies are general enough to be applicable to many KBSs. The output quality is measured using the recall and precision, which are well known in information retrieval. Furthermore, the general concepts of incompleteness and incorrectness can usually be used as measures for the quality of the input.

Insights: Degradation studies provide insights into the behavior of the system analyzed. As we demonstrated in our case study this includes its anytime performance with respect to incomplete data or its robustness with respect to incomplete or incorrect data or knowledge.

Performability: The degradation study we performed could quite easily be performed. We were able to do this because we had easy and quick access to the KBS (i.e., giving input to the system and reading its output by an external program). Furthermore, programs could be used to modify the input and knowledge base in a controlled way. These conditions should not be too hard to realize for many other KBSs.

Setup: Before performing a degradation study on a KBS, one should carefully consider the experimental setup. The way data or knowledge is removed or modified may greatly influence the outcome of the degradation studies.

The ultimate suggestion that follows from this work is that any KBS should upon delivery come accompanied with a set of degradation statistics such as discussed in this paper as a quantitative way of measuring interesting and important aspects of the systems quality. This would contribute to a more empirical and quantitative analysis of AI systems in general and of KBSs in particular, very much in the spirit of (Cohen 1995).

8. Future work

Although the results above already yield interesting insights in the behavior of a realistic KBS, many other aspects could still be uncovered using further degradation studies. We discuss some of these extensions in this section:

The informal definition for robustness that we used as a starting point in Section 2 has not been carried through entirely in the paper. Functioning correctly in the presence of invalid inputs has not been evaluated in the case study and should be included in future research.

(Zilberstein S 1996) suggests a category of measures on output quality which is not yet covered by the recall and precision that we have used in the above, namely 'specificity of a solution'. This is intended to represent the degree of detail in the systems' answer. An example would be a system which can compute names of ever finer grained plant-families instead of only individual species (as above). This property was irrelevant in our case study, since the plant-classification system only deals with a flat list of candidates, not with a hierarchically organized space of candidates. We have therefore ignored this potential third dimension of KBS output quality, but we expect that good measures can be devised for this just as well as for the other two dimensions which we did handle.

The output quality measures we used (precision and recall) are geared towards systems with a discrete output (a set of answers). Some KBS applications return real-valued answers (e.g., ratings). We must study how these systems can also be subjected to degradation studies using acceptable measures. In fact, the plant-classification system not only returns a set of candidates, but indicates a numeric score for each candidate. Our current output-measures completely ignore this score. A further step would be to also include this score in the quality measures.

As mentioned in Section 5.2, Figure 4 can be interpreted as a prediction for the expected quality of the output after a given number of observations. In effect, Figure 4 is the result of *learning the anytime performance profile* through the test-cases. As with any learning task, we can apply cross-validation to the set of test-cases (Cohen 1995): use a subset of the test-cases to 'learn' profiles as in Figure 4, and use the remaining cases to check the accuracy of the predicted performance levels.

Our measures for output quality (recall and precision) can only be computed for cases where the correct answer is actually known. This is not as obvious as it may sound. In many applications (e.g., computing the best solution to a design problem) the correct (i.e., best) answer is not known to any human expert. In such cases, one must either resort to known approximations of the correct answer, or fundamentally different quality measures must be defined.

Our proposed definitions for output quality (precision and recall) focus on the correctness of the answers computed by a system. Of course, there are many more aspects to the 'quality' of a KBS, such as the quality of its explanation, its computational efficiency, its interaction with its environment (be it users or other systems), etc. It is an open issue to us whether the same 'degradation study' approach can be taken to quantifying any of these other aspects of the systems quality.

Acknowledgements. We like to thank Frank Puppe for the help he has given us. This paper was inspired by a discussion with him during the Banff'98 KAW workshop. He gave us access to the plant-classification software and explained in detail its working. He was also always available to answer any question by e-mail.

References

Cohen P R (1995) Empirical methods for artificial intelligence. MIT Press.

Dean T, Boddy M (1988) An analysis of time-dependent planning. In Proceedings of the 7th National conference on artificial intelligence. Morgan Kaufmann, Saint Paul, Minnesota, august 1988, pp 49–54

Hayes-Roth F (1984) Knowledge-based expert systems – the state of the art in the US. In Fox J (ed). Expert Systems: state of the art report. Pergamon Infotech, Oxford.

IEEE (1990) IEEE standard glossary of software engineering terminology IEEE standard 610.12-1990, ISBN 1-55937-067-X

Menzies T, van Harmelen F (1990) Evaluating knowledge-engineering techniques. *International journal of human-computer studies* 51(4):715–727

Preece A D, Talbot S, Vignollet L (1997) Evaluation of verification tools for knowledge-based systems. *International journal of human-computer studies* 47(5):629–658

Puppe F, Poeck K, Gappa U, et al (1994) Wiederverwendbare bausteine für eine konfigurierbare Diagnostikshell. *Künstliche Intelligenz* 94(2):13–18

Puppe F, Gappa U, Poeck K, et al (1996) Wissenbasierte Diagnose- und Informationssysteme: mit anwendungen des expertensystem-shell-baukastens D3. Springer-Verlag, Berlin Heidelberg, Germany.

Salton G, McGill M J (1983) Introduction to Modern Information Retrieval. McGraw-Hill, New York

Schumann J, Fischer B (1997) NORA/HAMMR: Making deduction-based software component retrieval practical. In Proceedings of the 12th international conference on automated software engineering. IEEE Press, Lake Tahoe, CA, november 1997, pp 246–254

Shadbolt N R, O'Hara K, Crow L (1999) The experimental evaluation of knowledge acquisition techniques and methods: history, problems and new directions. *International journal of human-computer studies* 51(4):729–755

ten Teije A, van Harmelen F (1996) Computing approximate diagnoses by using approximate entailment. In Aiello G and Doyle J (eds). Proceedings of the 5th international conference on principles of knowledge representation and reasoning. Morgan Kaufmann, Cambridge, Massachusetts, november 1996, pp 256–265

ten Teije A, van Harmelen F (1997) Exploiting domain knowledge for approximate diagnosis (1997) In Pollack M E (ed). Proceedings of the 15th international joint conference on artificial intelligence. Morgan Kaufmann, Nogoya, Japan, august 1997, pp 454–459

van Harmelen F, ten Teije A (1998) Characterising approximate problem-solving by partial pre- and postconditions. In Prade H (ed). Proceedings of the 13th European conference on artificial intelligence. John Wiley & Sons, Ltd., Brighton, UK, august 1998, pp 78–82

Zilberstein S (1996) Using anytime algorithms in intelligent systems. *Artificial Intelligence Magazine* 17(3):73–83

Author Biographies

insert photo

Perry Groot received his degrees in 1998 in Computer Science (Artificial Intelligence) and Mathematics (Topology). He is currently a PhD student at the Department of AI, Vrije Universiteit Amsterdam. His main research interests are theoretical aspects of AI, e.g., approximate reasoning, reasoning with incomplete and incorrect knowledge and data, and logical formalisms. Other interests include abstract mathematics, e.g., Topology, Algebra, and Computational complexity. He can be contacted at `perry@cs.vu.nl` or via WWW on `www.cs.vu.nl/~perry`.

insert photo

Annette ten Teije holds a lectureship at the Vrije Universiteit Amsterdam. Her interests include approximate reasoning, formalisation of medical knowledge, configuration of reasoning methods, and diagnosis. She obtained a degree in Computer Engineering in 1991 and spent a year in the computing industry before starting work on her PhD. She earned a PhD (1997) from the University of Amsterdam (SWI) for her thesis entitled "Automated configuration of problem solving methods in diagnosis". She had been working as lecturer at the University of Utrecht. She can be contacted as `annette@cs.vu.nl` or via WWW on `http://www.cs.vu.nl/~annette`.

insert photo

Frank van Harmelen is a professor of Knowledge Representation and Reasoning at the Vrije Universiteit in Amsterdam. His research interests includes approximate reasoning within symbolic knowledge representation, exploiting Knowledge Representation for the Semantic Web, and using Knowledge Representation for medical applications, in particular medical protocols. He has contributed to the KADS project on formal and conceptual models for knowledge-based systems, and is currently active in W3C's working group on a standard ontology-language for the Semantic Web. He has widely published, amounting to over 100 research papers and three books.

Contact him at the Dept. of AI, Faculty of Sciences, Vrije Universiteit Amsterdam de Boelelaan 1081a, 1081HV Amsterdam, the Netherlands;`frank.van.harmelen@cs.vu.nl`; `www.cs.vu.nl/~frankh`.

Correspondence and offprint requests to: Perry Groot, Division of Mathematics and Computer Science, Faculty of Sciences, Vrije Universiteit, Amsterdam, The Netherlands. E-mail: perry@cs.vu.nl