PRONTO - Ontology-based Evaluation of Knowledge Based Systems

Trevor J.M. BENCH-CAPON, Dean M. JONES
Department of Computer Science, The University of Liverpool, Liverpool, England, L69 7ZF

Key words: ontology, rule base, conceptualisation, verification, validation

Abstract: In this paper we examine some of the ways in which an ontology can be used to assist in the evaluation of knowledge-based systems. Key elements of the support provided by the ontology relate to attempting to give coherence to the domain conceptualisation; making the role of experts in evaluation more structured and less at the mercy of interpretation; constraining the number of test cases required to give good coverage of the possible cases; and structuring the testing to give better assurance of its efficacy, and provide for a possible basis for greater automation of the testing process. The discussion is focussed on the development of a prototype software tool to support the approach and this is illustrated using a simple, well known, example relating to the identification of animals.

1. INTRODUCTION

In recent years ontologies have received an increasing amount of attention as a means of supporting the design, development and documentation of knowledge based systems (KBSs). An ontology can be seen as an "explicit specification of the conceptualisation of a domain" (Gruber 1995). Interest in them arises from the growing realisation that the clean separation of knowledge about the domain from task and control knowledge, on which many of the original hopes and expectations for KBSs were founded, is really very difficult to achieve in practice. Invariably the knowledge base will be distorted by considerations arising from the task to be performed on the knowledge, the problem solving method used, the form of representation, and the ways in which and the sources from which the knowledge was acquired. See Visser (1995) for a discussion of these problems.

Ontologies can trace their development from domain models. The tools which we describe are in the spirit of previous tools which took such domain models as their basis, such as Vanthienen (1991), which modelled the domain using decision tables. The idea here is that an ontology can provide a description of the domain which is - as far as possible - independent of the

way in which the domain knowledge is to be used, and the task it will be used for. Hitherto, ontologies have been used mainly for knowledge base development, knowledge sharing and knowledge reuse. They do, however, also have considerable potential for use in the verification and validation of KBSs as well. Some preliminary remarks on the role of ontologies in verification and validation were made in Bench-Capon (1998); this paper builds on those remarks and elaborates this role into an implemented prototype.

Throughout the paper we will use as an illustrative example a very simple rule base described in a well known text book on AI (Winston 1992). This rule base, called *ZOOKEEPER*, is concerned with the identification of animals. It is a useful example since everyone has a reasonable familiarity with the domain, and the example is small enough to be presented in a complete form (it is recapitulated in Appendix A). Moreover, since it appears in a text book it represents the sort of rule base which many people see as their first encounter with a KBS, and thus is responsible for many of the ideas people have about such systems.

In section 2, we describe the possibilities that an explicit specification of the conceptualisation of a domain allows for in the evaluation of a KBS. Section 3 describes in more specific terms the way in which we use ontologies for this purpose. In section 4 we outline how we go about developing an ontology for a given rule base and in section 5 we show what this allows. Section 6 is a discussion of ontology-based evaluation in relation to traditional notions of verification and validation and we give some concluding remarks in section 7.

2. USING ONTOLOGIES IN THE EVALUATION OF KNOWLEDGE-BASED SYSTEMS

In the evaluation of KBSs, a clear distinction can be made between evaluation of the internal and external consistency of a rule base. A rule base is internally consistent if it is structurally sound, which can be determined by ensuring that it free of subsumed rules, contradictions, dead end rules and the like. Internal consistency does not guarantee that a rule base will give the correct answer for any valid query, only that the rules are logically coherent. Determining whether or not the identifications produced by the rule-base are correct is the goal of the evaluation of its external consistency. This typically involves supplying sets of typical observations to the system and evaluating the results produced by the system in relation to some external yardstick (commonly the knowledge of a domain expert.) Additionally, we might present the rules to an expert and ask for confirmation of their correctness.

Given that we have a rule base which is both internally and externally consistent, can we say that the rule base is entirely satisfactory? The answer we give here is no, particularly if we are going to take seriously the possibility of extending the system with additional rules to cater for a wider range of cases. What we are suggesting is that evaluation of a rule base should encompass more than ensuring the absence of structural defects and that the correct answer is always given for the current set of cases, i.e. more than ensuring the internal and external consistency as these were defined above. What we want, in addition, is for the representation to have some kind of conceptual coherence, for it to be expressed within some well defined conceptualisation of the domain, which will promote extensibility and robustness.

Firstly, the distinctions that are made in a rule base should be based on a single, consistent conceptualisation of the domain. In our *ZOOKEEPER* example, distinctions were proliferated as and when they were needed in order to discriminate amongst the seven particular animals, and without much regard for distinctions that had already been made. If a system is to be built correctly, it should make principled distinctions, and make them in a justifiable manner. For example, spots are “dark” and stripes are “black”. Do we want a distinction between “dark” and “black”? What other varieties of spots and stripes might there be? Is there really a good difference between being white in colour with black stripes, black in colour with white stripes and whiteandblack in colour? Without a clear conceptualisation to serve as a reference point, it is futile to ask an expert to say whether rules are correct or not. For example, it might be that certain markings resemble rosettes. While one might be prepared to call them “spots” in the absence of an option to call them “rosettes”, assent to a rule using “spots” depends on an assumption as to whether the finer grain distinction is available or not.

We also need the required observations to be relatively easy to obtain, if the system is to be able to come up with answers consistently. For example, some of those required by the *ZOOKEEPER* rule base need judgement to be applied - in particular, the requirement be that an albatross flies “well”. This might well raise differences of opinion and interpretation. Others are rather hard to obtain: “lays eggs” is an occasional thing which might be hard to observe (and not observable at all in the case of a male of the species). At the very least we need to be aware of what information is likely to be available so that we resort to the information which is harder to obtain only when it is essential. In the ZOOKEEPER rule base it is essential that an giraffe or a zebra be first classed as an ungulate. Both of the observations required to classify an animal as an ungulate are, however, not always available. If the designer is unaware of this practical problems may arise in that giraffes and zebras may not be identified even though sufficient information is available,

whereas if the designer is aware of this problem, rules identifying zebras and giraffes in terms of more readily available observations can be supplied.

Much of the problem derives from the initial failure to conceptualise the domain in a coherent fashion. The strategy is first to classify an animal as a mammal or a bird, then sub-divide mammals into carnivores and ungulates, and then to discriminate members of these categories in terms of some observable features which are indicative of the particular animals in the collection. The higher level distinctions are theory driven, and the rules are determined by theory: for example Z4 is justified on the grounds that "some mammals fly and some reptiles lay eggs, but no mammal or reptile does both" (Winston, p122). But in the context of use of the system, Z4 is applicable only to the albatross, since the other two birds are flightless, and if it can fly it is an albatross, so its oviparity is neither here nor there. On the other hand, if we were to take the notion of extensibility seriously Z9 would be inadequate since it describes leopards and jaguars as well as cheetahs. As it stands here the rules are defective, with respect to the standards of a well constructed system, because they derive from conflicting conceptualisations of the domain, and conflicting ideas of how the system will be used. Separation of the animals into mammals and birds, and mammals into carnivores and ungulates, is obviously useful for a zoological taxonomy, but is of little practical importance in performing the identifications the system is supposed to supply.

The problems above derive in part from the lack of a clear specification of what the system will be used for as a starting point. Viewed simply from the standpoint of its real use, as an example rule base to illustrate forward and backward chaining, it is adequate. It is only when we project it into standing as a real application that we would need to specify whether it was supposed to identify only seven or an indefinite range of animals; whether it is meant to incorporate a known theory about animal classification, or to restrict itself to what can be seen; what kind of judgements the user of the system can be expected to make, and the like. As they stand the rules represent more the unstructured outpouring of information about the animals, rather than a well thought out plan for identification.

One major problem that has always existed in checking the external consistency of rule bases of a substantial size (and in this context even *ZOOKEEPER* can be considered substantial), is the combinatorial explosion that combining the predicates in test cases gives rise to. In the original *ZOOKEEPER* there were 20 predicates each of which appeared capable of being true or false independently, giving more than a million possible combinations. On this basis exhaustive testing can be considered impossible, and so test data must be selected by using some selected plausible combinations. There is, however, no systematic way of generating these and

so coverage of the important cases is not only not ensured, but there is not even any reliable way of estimating the coverage provided by the test data. In section 4, we will show how this can be improved through constructing an ontology for the rule base. First, we will illustrate how we construct the ontology for the rule base, continuing to base our on discussion on *ZOOKEEPER*.

3. THE DEVELOPMENT OF AN ONTOLOGY FOR RULE BASE EVALUATION

The question we address in this section is how we go about developing an ontology for the evaluation of a specific rule base. A recent survey of ontology development methodologies (Jones *et al.*, 1998) showed that there are two general strategies that are pursued in the construction of a new ontology, the stage-based approach and the evolving prototype approach. It was suggested that where a clear task can be identified and the purpose and requirements of the ontology are evident at the outset, the stage-based approach is most appropriate since this allows the ontology to be assessed in relation to the given requirements at the completion of each stage. In common with this, and since we have a well defined task, we adopted a stage-based strategy in the development of an ontology for the *ZOOKEEPER* rule base. We will now describe each of the phases in turn.

It was also suggested in Jones *et al.* (1998) that the initial phase of the development of a new ontology is commonly concerned with defining the minimal necessary scope of the ontology, as this allows us to ensure that the ontology at least satisfies the requirements of the task. The minimal requirement for our ontology of *ZOOKEEPER* is that it should permit the expression of all the facts and rules about the animals found in the original rules. Consequently, the first scoping exercise is to list the classes in the ontology, the hierarchical relationships between them (shown in Fig. 1) and the attributes that can be identified from the rules that are used to describe the classes. For *ZOOKEEPER*, the user is expected to be able to answer questions about the following predicates (note that the predicates relating to whether the animal is a mammal, bird, carnivore and ungulate are internal to the system; the user is neither asked questions about these predicates, nor sees any information about them.)

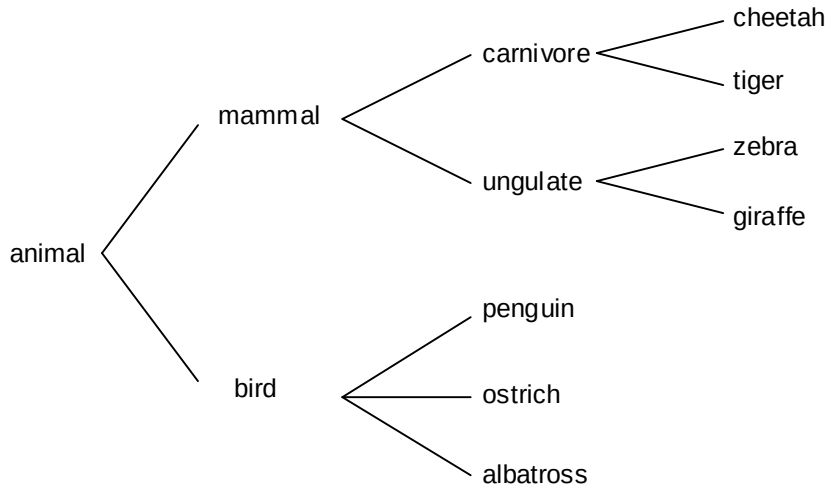

Figure 1: Class Hierarchy for *ZOOKEEPER*

a) has hair
b) gives milk
c) has feathers
d) flies
e) lays eggs
f) eats meat
g) long legs
h) long neck
i) tawny colour
j) dark spots
k) white colour
l) black stripes
m) black and white colour
n) swims
o) flies well
p) pointed teeth
q) claws
r) eyes point forward
s) hoofs
t) chews cud

One problem that can immediately be identified with this list is that, although some of the predicates seem to imply alternatives, only one option is used in the rule base because the alternatives are not needed by the current set of rules. For example, although it is possible to express the observation that an animal has eyes that point forward, the equally valid observation that an animal's eyes point sideways cannot be expressed. As outlined in section 2, the above predicates are not, as they stand, conceptually coherent. The second phase of the development of our ontology must be to organise the predicates into a principled set based on a coherent conceptualisation of the domain. Those predicates that allow for the addition of alternatives are grouped below with suitable values:

teeth{pointed, rounded} [p]
eats{meat, plants, everything}[f]
legs{long, normal} [g]
neck{long, normal} [h]
stripes{black, white} [l]
spots{dark, light} [j]
flies{well,poorly, no} [d,o]
eyes{forward, sideways} [r]

Some of the predicates specify alternative values for the same attribute and these can also be grouped together:

skin covering {hair,feathers} [a,c]
markings{spots,stripes} [j,l]
movesBy{swims,flies} [n,o]

In other cases the only values for the predicate are true or false:

gives milk [b]
lays eggs [e]
chews cud [t]

We can retain the remaining predicates (renamed where appropriate):

colour{white, tawny,black and white} [i,k,m]
feet{hoofs,claws} [q,s]

At the end of the second stage, we have a representation in which no two predicates describe the same attribute in the conceptualisation. However, the predicates do not yet form a collectively coherent set. Some of our predicates, *e.g.* flying and swimming, are not mutually exclusive (consider ducks and swans), so they must be separated. Moreover, flying appears to be a qualitative thing rather than a simple boolean: we could ask whether the same should apply to swimming as well, and indeed whether we want to include some kind of land motion such as running. We can also make the markings and colour situation more coherent by saying that an animal has a basic colour, and markings, which may be lighter or darker than the basic colour. Where we have gaps, because the options do not occur explicitly, these need to be filled. The problem of making the predicates mutually coherent is addressed in the third stage of development of the ontology. The rationalisations that occur during this phase are largely dependent upon the details of the conceptualisation being considered and are consequently difficult to generalise. However, the tasks carried out during this stage include (but are not limited to):

1. separation of predicates that are not mutually exclusive;
2. inclusion of additional alternatives to predicates separated during 1.;
3. decide whether values are boolean, qualitative, etc.;
4. decide which predicates have values that are mutually exclusive

At the completion of the third stage, we arrive at the situation where we can identify a set of attributes, and the possible values they can take. This will provide us with a well defined vocabulary with which to construct a set of rules. The set of conceptually coherent attributes for the *ZOOKEEPER* example is shown in Table 1.

The fourth phase is concerned with ascribing the relevant attribute-value pairs to each of the classes in our ontology, which provides us with a definition for the classes, under the current conceptualisation. Firstly, we

Coat :	Feet {claws,hoofs}
Material {hair,feathers}	Flies {no, poorly, well}
Colour {white, tawny, black}	Eats {meat, plants, both}
Markings :	Size :
Pattern {spots,stripes,irregular,none}	Neck {long, normal}
Shade {light,dark, n/a}	Legs {long, normal}
Facial Features	Gives Milk {true, false}
Eyes {forward,sideways}	Lays Eggs {true, false}
Teeth {pointed,rounded,none}	Chews Cud {true, false}
	Swims {true, false}

Table 1: Attributes and Values for *ZOOKEEPER*

construct a table listing each of the bottom-level classes and the relevant values for each of the attributes, such as that given in Table 2 (Note that some of the answers are conjectural - we are not experts, and are unsure what ostriches in fact eat, for example.) This process is likely to identify additional possibilities for some of the attributes (for example birds do not have teeth, and penguins eat fish), which will force us to extend the ontology accordingly. These attribute-pairs can be generalised to higher-level classes where it is both possible and plausible to do so. Some generalisations are logically possible given the current set of animals but are not realistic, e.g. all carnivores have tawny coats.

Next, we need to add some axioms, stating combinations which are impossible. Some of these combinations will have been identified during the third stage, especially in task (iv) from the list above. Examples include:

A1 Not (eats meat and chews cud)

A2 Not (Material feathers and chews cud)

A3 Not (Pattern none and shade not n/a)

In fact we could supply many more such axioms, but at this stage we need not attempt to be exhaustive. The addition of some axioms allows us to remove attribute values from the definitions of some of the classes, *e.g.* given A1 we do not need to include a value for the predicate `chews_cud/2` for each of the carnivores.

We now have an ontology which we can use to verify and validate a knowledge base built on it. First, however, we need to check the quality of the ontology itself. This is where the expert comes in: the expert should not be shown the encoded rules, but rather the ontology. This changes the role of the expert significantly. The expert no longer examines rules, but instead the vocabulary. With respect to the vocabulary the expert should check:

Predicate	Cheetah	Tiger	Zebra	Giraffe	Ostrich	Penguin	Albatross
Material	hair	hair	hair	hair	feathers	feathers	feathers
Colour	tawny	tawny	white	tawny	black	black	white
Pattern	spots	stripes	stripes	spots	irregular	irregular	none
Shade	dark	dark	dark	dark	light	light	n/a
Eyes	forward	forward	sideways	sideways	sideways	forward	sideways
Teeth	pointed	pointed	rounded	rounded	none	none	none
Feet	claws	claws	hoofs	hoofs	toes	toes	toes
Neck	normal	normal	normal	normal	long	normal	normal
Legs	normal	normal	normal	long	long	normal	normal
Gives Milk	true	true	true	true	false	false	false
Flies	no	no	no	no	no	no	well
Eats	meat	meat	plants	plants	both	meat	meat
Lays Eggs	false	false	false	false	true	true	true

Table 2: Attributes of Animals in *ZOOKEEPER*

1. that the attributes represent sensible distinctions
2. that the values are exclusive
3. that the values are exhaustive

The point about values can be addressed from two standpoints: either from the point of view of the existing collection, or from the point of view of a potentially extended collection. The first will indicate what is needed to test the rule base against its current operation, and the other will provide an indication of its extensibility. Also, to facilitate testing the expert should indicate whether observations are always available, or only sometimes available. Following this process we might modify Table 1 to give Table 3. Here always observable *attributes* are indicated in bold, as are *values* required by the current seven animals. The expert should also examine the table of attributes (Table 2), to confirm that these entries are correct. The table can be further verified by ensuring that it does not conflict with any of the axioms. By concentrating on the ontology rather than the rules, the role of the expert becomes much more well defined, and more systematic so that there is less possibility of interpretation allowing errors to go unnoticed.

4. PRONTO -¸A TOOL FOR ONTOLOGY-BASED EVALUATION OF RULE BASES

Once we are satisfied with the ontology we can proceed to evaluate the rule base. The ontology allows a substantial improvement on the number of test cases that were originally identified as necessary in section 2. The grouping together of attributes in the ontology identifies predicates that are not

Coat :
  Material {**hair**,**feathers**,scales}
  Colour {**white**, **tawny**, **black**, grey, russet}

Markings :
  Pattern {**spots**,**stripes**,**irregular**,**none**}
  Shade {light,**dark**, n/a}

Facial Features
  Eyes {**forward**,**sideways**}
  Teeth {**pointed**,**rounded**,**none**}

Feet {**claws**,**hoofs**,**toes**}
[Comment: feet are hard to observe (Winston)]

Flies {**no**, poorly, **well**}

Eats {**meat**, **plants**, both}
[Comment: meat includes fish]

Size :
  Neck {**long**, **normal**}
  Legs {**long**, **normal**}

Gives Milk {**true**, **false**}
Lays Eggs {**true**, **false**}
Chews Cud {**true**, **false**}
Swims {**true**, **false**}

Table 3: Validated Attributes and Values for *ZOOKEEPER*

independent. If we allow for 5 colours, coverage of these as booleans would require 64 cases. By considering them as they are in the ontology, however, there are only five cases. If we confine ourselves to testing only the attributes which the expert has identified as always available as observations, and only the values actually used by our current collection, we have only 1152 combinations. The useful test data moreover, contains only those cases which conform to the constraints imposed by the axioms. This enables a substantial further pruning. If we are able to identify a good set of axioms, then exhaustive testing becomes a possibility. We can now rigorously specify the minimal set of test cases that are required to exhaustively test the rule base. The ontology provides the essential input for our automated test harness, a prototype of which¸- called PRONTO - has been implemented in Prolog.

Once the minimal set of test cases for a given rule base has been identified, testing can begin with the evaluation of the results being provided by an expert. Incorrect output falls into one of the three possibilities that are outlined below with the potential causes of each type:

1. no answer; this can arise in several situations:
   a) the case represents an impossible combination, so an axiom should be added to the ontology to exclude such cases;
   b) the case is possible but these animals are not in the collection. The rule base is correct, and the combination should not be observed in practice. We can therefore either add a new rule, extending the coverage to animals outside the current collection, or simply disregard it; or
   c) the case is possible, but identification is reliant on some not always available feature.
2. a single incorrect answer. This indicates either:

a) the case is possible and an offending rule requires amendment, or
b) the case is impossible and an axiom should be added to the ontology is in order to exclude it;

3. multiple answers; here there are further possibilities;
   a) if none of the answers are correct, we have a similar situation to 2., *i.e.* either
      i) the case is possible and more than one rule requires amendment, or
      ii) the case is impossible and we need to add an axiom in order to exclude it.
   b) if the solution contains a single correct answer with at least one incorrect result, the ontology¸ is sufficient to discriminate the correct result but at least one rule needs to be made more specific (possibly using not always available features).
   c) if there is more than one correct solution the current ontology is inadequate and requires another predicate to discriminate the cases. The rule base should then be amended to include this new predicate. Additional modifications to the rule base may be required if the solution also includes erroneous results.

We should be careful in modifying the rule base not to introduce new problems. For example, a test case that produces a solution of type (1c) may encourage us to remove the antecedent that relies on the hard to observe feature. This, in turn, may result in the same case producing solutions of type (2) or (3). If so, we may have to reconcile ourselves to a certain incompleteness, or find some always available discriminating observation. Results of type (2) might lead us to introduce antecedents relating to intermittently observable features, whereas case (3) may motivate us to remove them.

This classification of the types of erroneous results leads us to the development of the testing process as represented diagrammatically in Fig. 1. Note that process 1, *generate case*, uses the ontology; the decision *violates axioms?*, uses the output from process 1 together with the axioms from the ontology; process 2, *execute cases*, uses the rule base and the mappings from the ontology predicates into the rule base predicates; *answer correct?* and *animal in collection?* requires an extensional description of actual animals such as is provided by Table 2; processes 3 and 4, *add axiom* and *modify ontology* modify the ontology; processes 5 and 6, *modify rules* and *add rule*, modify the rule base; and *case possible?* and requires input from the expert. Adding axioms will prune the cases subsequently generated and additional and modified rules are tested before a new case is generated.

The ontology cannot, of course, work magic: the testing effort required even with this test harness is substantial and non-trivial, requiring as it does considerable expert input. It does, however, supply the discipline and

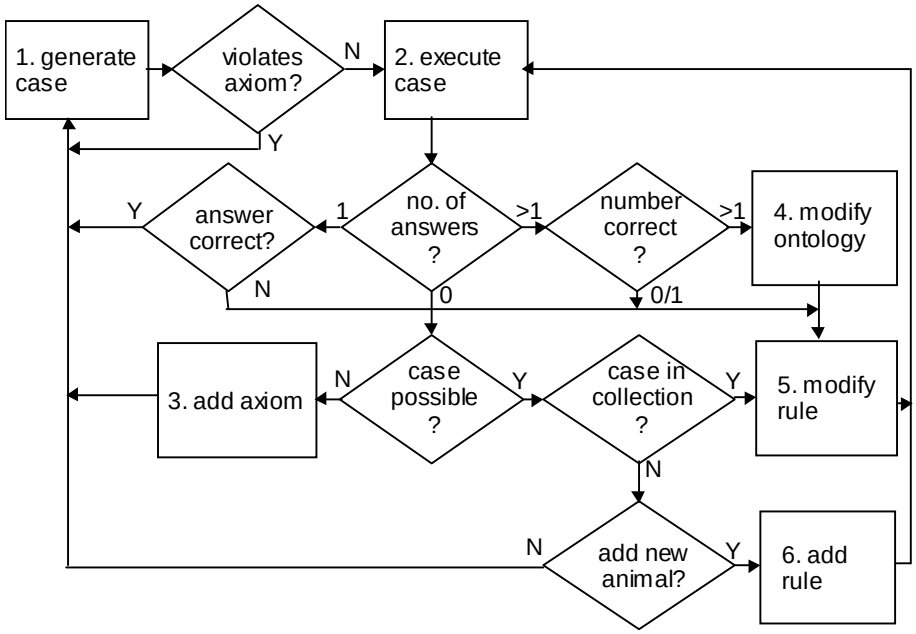

Figure 2: Schematic of Testing Process

structure necessary for testing a system, and in any event testing is always for any system an important and lengthy task, typically consuming anything between 25% and 30% of the development time of a software project. Moreover, the time spent in getting the initial ontology right, particularly with respect to a complete specification of the necessary axioms, is handsomely repaid by savings in testing required. In addition to these possibilities for evaluation against the ontology, normal structural checks should, of course, be applied. The quasi-random testing is, however, unnecessary given the more structured approach permitted by the ontology.

5. DEMONSTRATION OF PRONTO

Once we have defined the ontology for a rule base, we need to map the predicates found in the rule base onto the terms in the ontology. We can do this as a set of Prolog rules. A possible set of mappings is given in Table 4. Note that no mapping rules are given for `teeth/2`, `eats/2`, `eyes/2`, `colour/2`, `legs/2` or `neck/2` since those appear both in the rule base and the ontology. Note also that not always observable attributes are randomly included in test cases to reflect that they are sometimes available.

```
M01: hair(X) :- coat_material(X,hair).
M02: gives_milk(X) :- gives_milk(X,true).
M03: feathers(X) :- coat_material(X,feathers).
M04: flies(X) :- flies(X,true).
M05: flies(X,well) :- flies(X,true).
M06: lays_eggs(X) :- lays_eggs(X,true).
M07: has(X,Y) :- feet(X,Y).
M08: spots(X,dark) :- markings_pattern(X,spots),
     markings_shade(X,dark).
M09: spots(X,white) :- markings_pattern(X,spots),
      markings_shade(X,light).
M10: stripes(X,black) :- markings_pattern(X,stripes),
        markings_shade(X,dark).
M11: stripes(X,white) :- markings_pattern(X,stripes),
        markings_shade(X,light).
```

Table 4: Mappings between *ZOOKEEPER* Rule Base and Ontology

Our first example of the use of PRONTO illustrated in Appendix B1, the rule base identifies the test case of a swimming animal as a giraffe. Although this case could be eliminated from the evaluation process by adding an axiom, this would be the wrong option since the conditions are not impossible. It is simply a matter of fact that giraffes cannot swim, although some other long necked, spotted, tawny ungulate might be able to. In the classification given in section 4, this result is of type (2a) - a single incorrect result which is theoretically plausible. The correct option in this scenario is to add the condition `swims(X,false)` to rule Z11 to indicate that giraffes cannot swim. In our second example, given in Appendix B2, however an animal with claws is identified as a giraffe. If we look at the original rule base, all of the conditions in Z11 are satisfied. Since, according to our ontology, all ungulates have toes, we know that the case is impossible and we have a scenario of type (2b). The solution here, as can be observed from the example, is to add an axiom to rules out cases where we have both `chews_cud(X,true)` and `feet(X,claws)`.

For our third example, consider rules Z13 and Z14. On the basis of the guaranteed observable predicates, which, recall, do not include swimming and flying, cases will allow both these rules to fire, identifying the animal as both a penguin and an ostrich. Such a situation is shown in Appendix B3. Examination of this case reveals that ostrich is the right answer, since the cases contain `legs(X,long)` and `neck(X,long)`. This situation is classified as type (3b) - our ontology is sufficient to discriminate the animal that the case should be identified as but an additional erroneous solution is also included. We could rectify this situation without recourse to the

intermittently observable predicates by adding `legs(X,normal)` as an extra condition to Z14. Discriminating between a cheetah and a leopard would, however, require an extension to the ontology, since in terms of what is currently in the ontology the two are identical. We would need to extend the ontology to allow for a condition such as having the ability to climb trees, or having retractable claws.

6. DISCUSSION

Here we begin with Boehm's well known distinction between verification and validation (Boehm 1981):

verification: are we building the product right?
validation: are we building the right product?

The purpose of verification is to determine whether or not the implemented system correctly fulfils its design while the process of validation aims to ensure that the functionality embodied in the design meets the user's actual requirements. Verification is usually performed in one of two ways:

1. domain-independent analysis of relationships between the rules (this correlates to what we termed the assessment of internal consistency in section 2);
2. comparison of the behaviour of the system to a (more or less) formal design specification.

Validation, according to Boehm's definition, should be performed by comparing the requirements specification (which makes the user's requirements explicit) with the formal design specification. However, for KBSs formal requirements specification and design specification documents are rarely available. In theory, validation of a KBS should only be based on the results of test cases if the system has already been verified. For this reason, in the evaluation of KBS systems verification and valuation are often not distinguished and we find descriptions of V&V techniques, rather than separate discussions of methods of verification or of validation. When Boehm's definitions are applied to the evaluation KBSs, the clear distinction becomes somewhat blurred.

The ontology used in PRONTO specifies the conceptualisation underlying the rule base rather than the domain and can be taken to form part of the design specification. Now, whether erroneous results require us to change the rule base or the ontology indicates whether we are performing verification or validation. When the ontology is used to determine that the rule base should be changed, the system is being verified since it does not match the design as embodied by the ontology. On the other hand, when we are required to modify the ontology we are performing validation as (this part of) the

	1a	1b	1c	2a	2b	3a(i)	3a(ii)	3b	3c
verification			✔	✔		✔		✔	
validation	✔	✔			✔		✔		✔

Table 5: Case Types as Verification or Validation

specification does not fulfil the requirements of the user. Recall the different types of erroneous results that were distinguished in section 4; where the rule base requires modification (types 2a, 3a(i) and 3b) the system does not satisfy the ontology and we can say that making these types of changes is verification of the system. However, where the ontology needs refinement (types 1a, 1b, 1c, 2b, 3a(ii) and 3c) we can say that the specification does not match users requirements and modifying the ontology is validation of the system.

The situation is slightly more complex, however, because if we actually revisit the descriptions given in section 4, we see that in case 1b the ontology and the rule base are in fact correct and should only be modified if we want to extend the coverage of the rule base (thereby extending the user's requirements. This is, therefore, neither verification nor validation. Also, results of type (3c) require that we modify both the ontology and the rule base. However, as the initial rule base matches the specification in the form of the ontology, here we are performing validation only. This is summarised in Table 5.

The question is, do we need to worry about maintaining the distinction? That is, does our ability to assess whether a particular evaluation technique is verification or validation help in producing better KBSs? We would say that the answer is no, since the purpose of the distinction was originally to help determine whether the problem lies with the implementation or the design. If knowing whether we are involved in verification or validation requires us to know whether a test case indicates a problem with the rule base or with the ontology, we have already addressed the original problem and provided we recognise that there are two separate forms of difficulty, the terminology used is not important

7. CONCLUSIONS

In this paper we have shown how an ontology can be used to aid verification and validation, illustrated by a simple example, and a discussion of an implemented prototype system. The main conclusions are:

– having an ontology provides an objective point of reference for verification and validation activities;
– much of the interaction with the expert can be done in terms of the ontology. This means that the role of the expert is better defined, and it is not necessary to judge rules which may depend for their meaning on implicit assumptions, and the context within which they will be used. Essentially the expert can focus on the conceptualisation, free from implementation details;
– testing can be structured by the ontology;
– when test fails, the failures can be classified so as to determine the appropriate response;
– some test results will result in a modification of the program and others in a modification of the ontology. As was discussed in section 6 this provides a useful distinction between failures resulting from the way the conceptualisation has been implemented to the program and failures resulting from inadequacies in the conceptualisation itself.

For future work we would like to extend the Pronto system to incorporate existing work in the evaluation of KBSs. In particular, we aim to incorporate a facility that will assist in the identification the rules which need modification (Coenen and Bench-Capon, 1993).

REFERENCES

Bench-Capon, T.J.M. (1998) "The Role of Ontologies in the Verification and Validation of Knowledge Based Systems", *Proceedings of the Ninth International Workshop on Database and Expert Systems*, IEEE Press, Los Alamitos, pp64-69.

Coenen, F.P. and T.J.M. Bench-Capon (1993) *Maintenance of Knowledge Based Systems: Theory, Tools and Techniques*, Academic Press, London.

Boehm, B.W. (1981) *Software Engineering Economics*, Prentice Hall.

Gruber, T.R. (1995) "Towards Principles for the Design of Ontologies Used for Knowledge Sharing", *Int. J. Human-Computer Interaction*, **43**, 907-928.

Jones, D.M., T.J.M. Bench-Capon, and P.R.S. Visser (1998) "Methodologies for Ontology Development", *Proc. IT&KNOWS Conference of the 15th IFIP World Computer Congress*, Budapest, Chapman-Hall.

Vanthienen, J. (1991) "Knowledge Acquisition and Validation Using a Decision Table Engineering Workbench", 1st World Congress on Expert Systems, Pergamon Press, pp1861-8.

Visser, P.R.S. (1995) "Knowledge Specification for Multiple Legal Tasks", Kluwer.

Winston, P.H. (1992) *Artificial Intelligence*, Third Edition. Addison Wesley, Reading, Mass.

APPENDIX A - THE *ZOOKEEPER* RULE BASE

The rulebase for *ZOOKEEPER* is given in Winston (1992), page 121-4. It is explicitly limited to the identification of seven animals: a cheetah, tiger, zebra, giraffe, ostrich, penguin and an albatross. It has15 rules, enabling identification of these seven animals, often in several ways, to allow for some observations being unobtainable. The rules (expressed here in Prolog form) are:

```
Z1: mammal(X) :- hair(X).
Z2: mammal(X) :- givesMilk(X).
Z3: bird(X) :- feathers(X).
Z4: bird(X) :- flies(X),
              laysEggs(X).
Z5: carnivore(X) :- mammal(X),
      eats(             X,meat).
Z6: carnivore(X) :- mammal(X),
      teeth(             X,pointed),
      has(             X,claws),
      eyes(             X,forwardPointing).
Z7: ungulate(X) :- mammal(X),
     has(             X,hoofs).
Z8: ungulate(X) :- mammal(X),
                   chewsCud(X).
Z9: cheetah(X) :- carnivore(X),
    colour(             X,tawny),
    spots(             X,dark).
Z10: tiger(X) :- carnivore(X),
   colour(             X,tawny),
   stripes(             X,black).
Z11: giraffe(X) :- ungulate(X),
     legs(             X,long),
     neck(             X,long),
     colour(             X,tawny),
     spots(             X,dark).
Z12: zebra(X) :- ungulate(X),
   colour(             X,white),
   stripes(             X,black).
Z13: ostrich(X) :- bird(X),
     not flies(X),
     legs(             X,long),
     neck(             X,long),
     colour(             X,blackandwhite).
Z14: penguin(X) :- bird(X),
     swims(X),
     not flies(X),
     colour(             X,blackandwhite).
Z15: albatross(X) :- bird(X),
       flies(             X,well).
```

APPENDIX B1

```
Testing case:
  swims: true
   lays_eggs: false
   gives_milk: true
   legs: long
   neck: long
   feet: hoofs
   teeth: rounded
   eyes: forward
   markings_pattern: spots
   markings_shade: dark
   coat_colour: tawny
   coat_material: hair
One result: giraffe.
Is the answer correct (y/n/q)?
|: n.
You need to modify a rule.
Enter head of rule to modify.
|: animalis.
Enter arity of rule to modify.
|: 2.
Current listing for rule is:
Clause 1:
animalis(A, cheetah) :-
¸ carnivore(A),
        colour(A, tawny),
        spots(A, dark).
Clause 2:
animalis(A, tiger) :-
¸ carnivore(A),
¸ colour(A, tawny),
¸ stripes(A, black).
Clause 3:
animalis(A, giraffe) :-
¸ ungulate(A),
¸ legs(A, long),
¸ neck(A, long),
¸ colour(A, tawny),
¸ spots(A, dark).
Clause 4:
animalis(A, zebra) :-
¸ ungulate(A),
¸ colour(A, white),
¸ stripes(A, black).
Clause 5:
animalis(A, ostrich) :-
¸ bird(A),
¸ not flies(A),
¸ legs(A, long),
¸ neck(A, long),
¸ colour(A, black),
¸ spots(A, white).
Clause 6:
animalis(A, penguin) :-
¸ bird(A),
¸ swims(A),
¸ not flies(A),
¸ colour(A, black),
¸ spots(A, white).
Clause 7:
animalis(A, albatross) :-
¸ bird(A),
¸ flies(A, well).
Select option:
1. Add clause.
2. Add condition to clause.
3. Remove clause.
4.   Remove   condition   from
clause.
|: 2.
Enter  clause  number  to  add
condition
for animalis/2.
|: 3.
Current clause.
animalis(A, giraffe) :-
¸ ungulate(A),
¸ legs(A, long),
¸ neck(A, long),
¸ colour(A, tawny),
¸ spots(A, dark).
Enter new condition.
|: swims(A,false).
New clause.
animalis(A, giraffe) :-
¸ ungulate(A),
¸ legs(A, long),
¸ neck(A, long),
¸ colour(A, tawny),
¸ spots(A, dark),
¸ swims(A, false).
Re-testing case:
   swims: true
   lays_eggs: false
   gives_milk: true
   legs: long
   neck: long
   feet: hoofs
   teeth: rounded
   eyes: forward
   markings_pattern: spots
   markings_shade: dark
   coat_colour: tawny
   coat_material: hair
No answers.
```

APPENDIX B2

```
Testing case :
   lays_eggs: false
   gives_milk: true
   chews_cud: true
   legs: long
   neck: long
   feet: claws
   teeth: rounded
   eyes: forward
   markings_pattern: spots
   markings_shade: dark
   coat_colour: tawny
   coat_material: hair
One result: giraffe.
Is the answer correct (y/n/q)?
|: n.
Is the case possible (y/n)?
|: n.
You need to add an axiom.
Enter attribute and value pairs
for
new axiom.
Enter attribute.
Possibilities:
[coat_material,coat_colour,
markings_pattern,
markings_shade,   eyes,   teeth,
feet, flies, eats, neck, legs,
gives_milk,lays_eggs,
chews_cud, swims].
|: chews_cud.
Enter value for attribute
'chews_cud'.
Possibilities: [true, false].
|: true.
Enter attribute.
Possibilities: [coat_material,
coat_colour, markings_pattern,
markings_shade,   eyes,   teeth,
feet,
flies, eats, neck, legs,
gives_milk,lays_eggs,
chews_cud,
swims].
|: feet.
Enter   value   for   attribute
'feet'.
Possibilities: [claws, hoofs,
toes].
|: claws
```

APPENDIX B3

```
Testing case:
   swims: true
   gives_milk: false
   legs: long
   neck: long
   flies: false
   feet: claws
   teeth: pointed
   eyes: forward
   markings_shade: light
   markings_pattern: spots
   coat_colour: black
   coat_material: feathers
More than one result: [penguin,
ostrich].
Can you modify a rule (y/n/q)?
|: y.
Enter head of rule to modify.
|: animalis.
Enter arity of rule animalis to
modify.
|: 2.
Current   listing   for   rule
animalis/2
is:
Clause 1:
animalis(A, cheetah) :-
¸ carnivore(A),
¸ colour(A, tawny),
¸ spots(A, dark).
Clause 2:
animalis(A, tiger) :-
¸ carnivore(A),
¸ colour(A, tawny),
¸ stripes(A, black).
Clause 3:
animalis(A, giraffe) :-
¸ ungulate(A),
¸ legs(A, long),
¸ neck(A, long),
¸ colour(A, tawny),
   ¸ spots(A, dark).
Clause 4:
animalis(A, zebra) :-
¸ ungulate(A),
¸ colour(A, white),
¸ stripes(A, black).
Clause 5:
animalis(A, ostrich) :-
¸ bird(A),
¸ not flies(A),
¸ legs(A, long),
¸ neck(A, long),
¸ colour(A, black),
¸ spots(A, white).
Clause 6:
animalis(A, penguin) :-
¸ bird(A),
¸ swims(A),
¸ not flies(A),
¸ colour(A, black),
¸ spots(A, white).
Clause 7:
animalis(A, albatross) :-
¸ bird(A),
¸ flies(A, well).
Select option:
1. Add clause.
2. Add condition to clause.
3. Remove clause.
4.   Remove   condition   from
clause.
|: 2.
Enter clause number to add
condition for animalis/2.
|: 6.
Enter new condition.
|: legs(A,normal).
Re-testing case:
   swims: true
   gives_milk: false
   legs: long
   neck: long
   flies: false
   feet: claws
   teeth: pointed
   eyes: forward
   markings_shade: light
   markings_pattern: spots
   coat_colour: black
   coat_material: feathers
One result: ostrich.
```