Why Ontologies are not Enough for Knowledge Sharing*

Flávio S. Corrêa da Silva†
Instituto de Matemática e Estatística
Universidade de São Paulo, São Paulo, Brazil
fcs@ime.usp.br

Wamberto Weber Vasconcelos‡
Departamento de Estatística e Computação
Universidade Estadual do Ceará, Ceará, Brazil
wvasconcelos@acm.org

Jaume Agustí
Institut d'Investigació en Intel.ligencia Artificial
Campus UAB 08193 Bellaterra, Barcelona, Spain
agusti@iiia.csic.es

David Robertson
Department of Artificial Intelligence
University of Edinburgh, Scotland, Great Britain
dr@dai.ed.ac.uk

Ana Cristina V. de Melo
Instituto de Matemática e Estatística
Universidade de São Paulo, São Paulo, Brazil
acvm@ime.usp.br

Abstract

Knowledge sharing is difficult. One reason is that it is hard to decide how to describe a domain in a way which suits everyone with an interest in the knowledge. Tackling this problem has been a central theme of the surge in ontological research over recent years. Unfortunately, getting an agreed ontology is not the end of our problems, since the way we represent knowledge is intimately linked to the inferences we expect to perform with it. This paper looks at some of the problems which sharing inference mechanisms may involve. Often these require us to find ways of coping with losses of information or search strategies which are not the ones we had originally intended. We introduce three very simple inference systems and discuss the problems of having knowledge passing through them, which are representative of complex problems we may need to solve for knowledge sharing. These systems are a classical Horn clause logic, a fuzzy inference system and a logic for planning. All systems and problems are presented via motivating practical examples.

1 Introduction

Mainstream research in knowledge sharing among knowledge-based systems concentrates on the mappings between different domain-specific notations while making the assumption that the inference mechanisms of each system are compatible. Sometimes this assumption is explicit: to share knowledge, each system must translate its knowledge into a standard system of inference [Sub, NG, Gra]. At other times the assumption is implicit: a standard knowledge representation language is provided but it is (mostly) left to users of the notation to choose compatible forms of inference [Gua98, UG96].

There is good reason for making this assumption because it is difficult without it to guarantee that the meaning of knowledge expressed in one system is preserved when used by another system. In theory, this guarantee would require us to demonstrate that the models of the world permitted by a system supplying its knowledge include all the models of the world permitted by the system receiving that knowledge. If we allow donors of knowledge also to be recipients then the theoretical constraint becomes even stronger: the models permitted by all systems must coincide precisely. This raises a major practical problem because differences in models of the world often show up in the differing styles of inference which we use to derive consequences from the knowledge we represent. For instance, in an assembly system we might express rules of assembly using the notation of classical logic but our inferences might be resource-bounded (that is, each piece of information is used no more than once in obtaining a solution). If the rules for this

*Work sponsored by the Consortium British Council/CAPES (Brazil), Grant no. 070/98, and the Science Foundation of the State of Ceará (FUNCAP), Brazil.

†Partially sponsored by FAPESP, Grant no. 93/0603-01, and CNPq, Grant no. 300041/93-4.

‡Partially sponsored by CNPq, Grant no. 352850/96-5.

system were used within a "classical" deduction system then the resource limitation would be removed and we might be able to derive solutions which were unobtainable from the donor system.

There is no definitive solution for this problem, other than the impractically restrictive rule that all knowledge based systems must share the same models of the world. Alternatively, we have looked for localised solutions where the relationship between inference systems is close enough to allow sharing of knowledge in limited forms [CdSVR98]. This sometimes involves loss of information but in many cases this loss may be detected, assessed and made tolerable. The purpose of this paper is to present some problems which occur with this approach and suggest ways of solving them. We do this by describing a high-level method for assessing how knowledge of particular kinds may be shared (Section 4). To make our discussion concrete, we use as examples three simple inference systems (introduced in Section 3). Our analysis is based on logic because this allows us to maintain a clear separation between logical axioms, describing domain knowledge, and the inference rules used to manipulate that knowledge. We can also be sure that if problems can be found at the idealised level of logical analysis then they will occur at least as strongly in actual applications. To make our examples easier to follow we use a running example which we introduce in the next section.

2 A Simple Knowledge Sharing Scenario

The purpose of this section is to introduce informally a scenario upon which we shall base a simplified formal example. Suppose that we have the task of combining knowledge-based systems involved in ordering appropriate personal computers. Three systems are involved:

- Our business is international so different computers may be assembled in different countries. We want each computer built in each country to be assembled from parts available in that country. Therefore we have a system giving a catalogue of the types of components which are available in each country and telling us how these components may be combined to produce product specifications for different countries. This is based on a classical logic and is described in Section 3.1.

- Our customers sometimes impose acceptability requirements on their orders – they may want, for example, to trade off cheapness against quality. As a consequence of a separate business analysis effort we have a system which partially describes these requirements. Since our requirements are imprecise, this is based on a fuzzy inference system described in Section 3.2.

- To assist in the control of our production processes we have a system for suggesting how the components we have in stock at each of our factories may be assembled to form the larger scale components of our computers. Since this form of advice is based on the assumption that each individual component is used once, and cannot then be used again, we use a resource-bounded inference system as described in Section 3.3.

We would like to allow these three independently constructed inference systems to be able to share their knowledge. For example, it might be preferable that our catalogue-based system for describing which computers are available sometimes consults directly with the stock inventories at factories, which means it must communicate with our resource-bounded system. Numerous other opportunities for communication exist, as we shall see in Section 4 where we follow through the methods for making appropriate connections. First, we introduce detailed examples of each of the three inference systems.

3 Three Simple Inference Systems

To avoid becoming lost in details it is necessary to keep the examples of this section simple. This is a limitation but not a serious one – the point of our paper is to show how some key problems may be anticipated when attempting to share knowledge so if we find problems in these simple systems we can expect at least these same problems in more complex systems.

3.1 A Classical Inference System

Our product specification system contains definitions of the role of different catalogue entries (*e.g.* $disk(dtype_1)$ says that the catalogue part $dtype_1$ is a disk):

$$
\begin{gathered}
disk(dtype_1)\\
disk(dtype_2)\\
memory(mtype_1)\\
screen(stype_1)\\
keyboard(ktype_1)\\
mouse(xtype_1)\\
button(btype_1)
\end{gathered}
$$

Our product specification for a computer[1] combines the appropriate components:

$$computer(D, M, S, K, X) \leftarrow disk(D) \wedge memory(M) \wedge screen(S) \wedge keyboard(K) \wedge mouse(X)$$

Goals are satisfied in this system in a standard backward-chaining style based on classical logic, in which we attempt to find an instance satisfying the goal and can search for further instances on backtracking. For example, the goal $computer(D, M, S, K, X)$ is satisfied by the instance $computer(dtype_1, mtype_1, stype_1, ktype_1, xtype_1)$ and, upon backtracking, the instance $computer(dtype_2, mtype_1, stype_1, ktype_1, xtype_1)$.

3.2 A Fuzzy Inference System

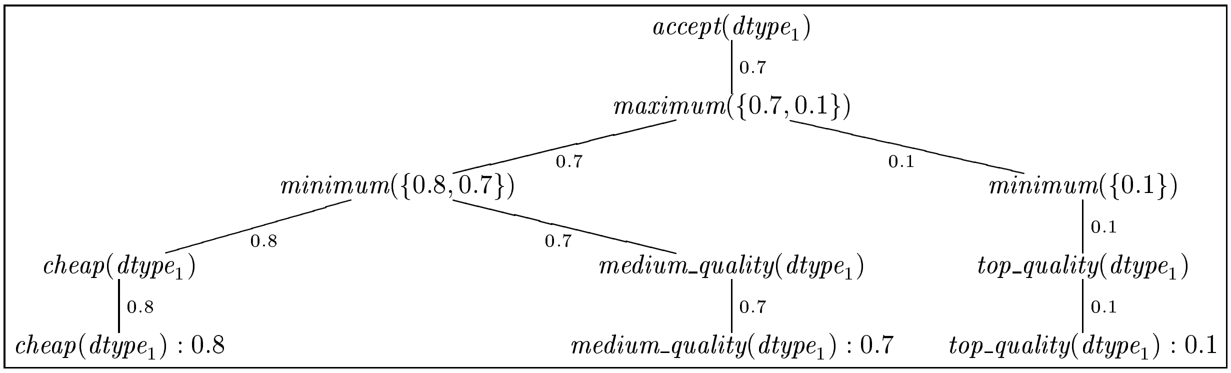
Our system for acceptability requirements uses a fuzzy logic which allows the degree of acceptability of any component type to be related to its properties [AEG+91, CdSC94, CdS91, CdSRH94, Lee72, Sha83] (in our example these are its cheapness and the degree to which it is considered medium or top quality).

$$
\begin{aligned}
accept(X) &\leftarrow cheap(X) \wedge medium\_quality(X)\\
accept(X) &\leftarrow top\_quality(X)
\end{aligned}
$$

We then associate the acceptability-related properties with fuzzy truth values. For example:

$$cheap(dtype_1) : 0.8 \quad medium\_quality(dtype_1) : 0.7 \quad top\_quality(dtype_1) : 0.1 \quad top\_quality(dtype_2) : 0.6$$

Our fuzzy inference mechanism will operate by finding the maximum obtainable truth value for a given goal, with the truth value for any conjunction of goals taken as that of the least preferred conjunct. For instance, the goal $accept(dtype_1)$ is solved as follows:



The tree above shows the goals we had to solve in order to satisfy the initial query. The fuzzy values obtained are shown as labels associated to branches of the tree, conveying the fact that that value was passed on to the parent node as part of the inference mechanism. For instance, fuzzy value 0.8 obtained in the left-most leaf node $cheap(dtype_1)$ is passed on to its parent node which, on its turn, passes it on to node *minimum*; the *minimum* node then calculates the minimum among its nodes and passes the result, 0.7, to the *maximum* node; the *maximum* node performs a similar calculation, but chooses the maximum of the values associated to its branches, passing this value to its parent node, the initial query. We use the same format later when we represent proofs which share knowledge, using bracketing to indicate when we have used knowledge from a different system. Deducted information (unifications or fuzzy values) is presented as labels of branches

[1] This is obviously an incomplete specification but enough for the purposes of our example.

3.3 A Resource-Bounded Inference System

Our assembly system contains definitions which explain how to construct components from subcomponents. For example, a two-button mouse (where m_2 denotes the type of mouse) needs to have two button subcomponents:

$$mouse\_component(m_2, B_1, B_2) \leftarrow button\_component(B_1) \wedge button\_component(B_2)$$

The assembly system also records which individual components are in stock. For instance, we might have only two buttons left:

$$button\_component(b_1) \quad button\_component(b_2)$$

The inference mechanism for this system is the same as for the specification system of Section 3.1 with the additional constraints that each fact can be used no more than once, and in the proposed order as occurring in the clauses. Thus, the goal $mouse\_component(X,\ B_1,\ B_2)$ would succeed, with the instance $mouse\_component(m_2, b_1, b_2)$ but there would be no solution for the goal $(mouse\_component(X_1, B_1, B_2) \wedge mouse\_component(X_2, B_3, B_4))$ because only two buttons are available in our database. This characterises a resource-bounded logic, similar to those found in *e.g.* [SSCdSR96, BK92, ADP96, Gir87]

4 Method of Analysis

Our aim is to use examples of knowledge sharing between simple representatives of common forms of inference system to suggest problems which may occur. Our representative systems are those of Section 3. We first consider (in Section 4.1) whether knowledge sharing is needed at all. We then examine solutions to knowledge sharing which rely on systems who donate knowledge being viewed as oracles (Section 4.2). Finally, we explore solutions which rely on the knowledge of one system playing the role of a surrogate knowledge base for another (Section 4.3). In both these sections we consider sharing in each direction between classical and fuzzy systems and between classical and resource bounded systems[2]. This may be visualised as shown below:

$$\boxed{\text{Fuzzy}} \Longleftrightarrow \boxed{\text{Classical}} \Longleftrightarrow \boxed{\text{Resource-bounded}}$$

Each example of knowledge sharing raises a general question which is summarised in the sentences labelled **Question 1** to **9**. Although derived from logic, these questions apply to any sort of system with the properties described.

4.1 How Much Can Be Done Locally?

The first question we need to ask is whether we need to have three different communicating systems at all. Perhaps it would be easier simply to merge the systems into one large, heterogeneous system. The easy part of this would be merging the knowledge bases, since their syntax is similar and there is no obvious ontological conflict (when we use different names we have different concepts in mind). The difficulty comes in merging the inference mechanisms. We cannot simply combine them because it is not clear what it would mean to have an inference mechanism which was *simultaneously* classical, fuzzy and resource bounded. On the other hand, using each inference mechanism independently does not allow us to share knowledge between systems any more than when they were physically separate. To do that, we need to agree on the correspondences between the terms which we wish to share and assess the extent to which the conclusions derived by one inference system can be trusted by appropriate recipient systems.

Question 1 *Can the inference systems and knowledge bases be merged cost-effectively, avoiding the need for knowledge sharing?*

4.2 Can Donor Systems be Oracles?

Perhaps the simplest way in which systems can share knowledge is for the recipient to view the donor as an oracle, which always provides all the right answers. Whether we can maintain this level of trust depends on which sorts of system need to communicate.

[2]We could also have considered the interaction between resource bounded and fuzzy but this does not add significantly to the analysis.

4.2.1 Oracle: Classical Specification Receives Fuzzy Requirements

If we wish to have our model of acceptability influence the specification of our computers then we could do this by defining new axioms within the specification system (Section 3.1) which relate the fuzzy acceptability constraints to the specification. One axiom which we might add to Section 3.1 is:

$$\begin{array}{rcl} accept\_comp(D,M,S,K,X) & \leftarrow & computer(D,M,S,K,X)\wedge \\ & & economical\_to\_accept(D)\wedge \\ & & economical\_to\_accept(M)\wedge \\ & & economical\_to\_accept(S)\wedge \\ & & economical\_to\_accept(K)\wedge \\ & & economical\_to\_accept(X)\wedge \end{array}$$

The *economical_to_accept* conditions are not defined within the specification system. Instead, we link them to the fuzzy definition of *accept* in the requirements system of Section 3.2. This requires us to provide translations, along the lines of those described in [CdSVR98], between the predicate names and also from the fuzzy values of the requirement system to the boolean truth values of the specification system [AEG+91, ZO89, REPG98]. We lose information this way because we are compressing the shades of meaning available in the fuzzy system into a true or false decision. The most obvious way to do this is by agreeing that whenever a fuzzy value exceeds a given threshold value it is translated into "true" in our classical system (shown as $\top^C$ below). Our translations are therefore:

Expression in donor	Expression in recipient
$accept(X)$	$economical\_to\_accept(X)$
Fuzzy value (μ), where $\mu \geq$ threshold value	Boolean truth ($\top^C$)

This narrow channel of communication allows us to bring the acceptability requirements within the specification system but the results we obtain are only as reliable as the assumption we made in choosing the threshold value.

Question 2 *What information may be lost if we use a system with complex truth values as an oracle for a system with simpler truth values?*

4.2.2 Oracle: Fuzzy Requirements System Receives Classical Specification

All the requirements of Section 3.2 refer to subcomponents but not to assembled computers. We might like to include information from the specification system of Section 3.1 in order to talk about requirements on possible computer systems. For instance, a computer system might be considered acceptable if its disk, memory and screen are acceptable; which we might express by adding the following axiom to the requirements system[3]:

$$\begin{array}{rcl} accept(computer(D,M,S,K,X)) & \leftarrow & available\_computer(D,M,S,K,X)\wedge \\ & & accept(D)\wedge accept(M)\wedge accept(S) \end{array}$$

The need for translation here is similar to the previous example but in the opposite direction. Again, we must decide on the mapping between boolean and fuzzy truth values, this time associating classical truth with our fuzzy maximum of 1.

Expression in donor	Expression in recipient
$computer(D,M,S,K,X)$	$available\_computer(D,M,S,K,X)$
Boolean truth ($\top^C$)	Fuzzy value (1)

The assumptions made in sharing this knowledge are similar to the previous case. Although in our example the mapping of truth values seems obvious, since we either have a specification or not, we could imagine circumstances when we might set a lower fuzzy value (for example, if we had reason to believe that specifications were not always reliable). The mapping above between the classical truth value $\top^C$ and the fuzzy value 1 is correct [AEG+91, AEG+94, REPG98], since we use an appropriate minimum combinator. If different combinators had been used, the morphism between these logics might have been destroyed .

[3] Here we have overloaded the name *computer* by using it as the functor in a structured term as well as a predicate name but there is no clash in our use of these names and the overloading here seems natural.

Question 3 *How do we approximate when using a system with simple truth values as an oracle for a system with complex truth values?*

4.2.3 Oracle: Resource-Bounded Assembler Receives Classical Specification

Our resource-bounded system is only allowed to use the supply of buttons which it has in stock but on some occasions we might want to guess which components we are able to assemble if we ordered sub-components of the appropriate type from our catalogue of part types. This catalogue resides in the specification system of Section 3.1. To connect to it, we must translate the appropriate component names. Translating the truth values of both logics, which lost information in our previous example, is safe here because the notion of truth in both logics is similar (although the consequences obtainable from the resource bounded logic are a subset of those for the classical system):

Expression in donor	Expression in recipient
$button(B)$	$button\_component(B)$
Boolean truth ($\top^C$)	Boolean truth ($\top^R$)

This allows the assembly system to satisfy goals which it was previously unable to solve. For instance, in Section 3.3 we were unable to find a solution for $mouse\_component(X_1, B_1, B_2) \wedge mouse\_component(X_2, B_3, B_4)$ but now we can use the specification system to obtain the solution $mouse\_component(m_2, b_1, b_2) \wedge mouse\_component(m_2, btype_1, btype_1)$. The two instances of $btype_1$ come from our specification system and we must understand these not as instances of particular buttons (like b_1 and b_2) but as "promises" that we could obtain buttons of the type *btype*1 from our catalogue. Consequently, we are changing the interpretation of *mouse_component* in the resource-bounded system - making it no longer resource bounded (since we can get as many instances of $btype_1$ from the classical system as we need) and having it generate both objects in stock and markers from the catalogue.

Question 4 *If we use as an oracle a system with fewer constraints on acceptable solutions than the system receiving its advice, can we check that the additional solutions it may generate will be helpful?*

4.2.4 Oracle: Classical Specification Receives Resource Bounded Assembly

We might decide that for some components our specification system should, instead of looking up the appropriate part type in the catalogue, use the assembly system to generate a description of it. For instance, we might remove the information about *mouse* from the axioms of Section 3.1 and connect to the resource-bounded system of Section 3.3 by allowing *mouse* to succeed in the specification system if *mouse_component* succeeds in our assembly system. Our translations in this case are similar to those of the previous example but the direction is reversed:

Expression in donor	Expression in recipient
$mouse\_component(M, B_1, B_2)$	$mouse(M)$
Boolean truth ($\top^R$)	Boolean truth ($\top^C$)

As in the previous example, we must be aware when choosing this sort of connection that we are also changing the meaning of predicates in the classical system. In particular, we are limiting the number of solutions we can find for $mouse(M)$. Previously we would have obtained a solution to the goal $(mouse(M_1) \wedge mouse(M_2))$ with both M_1 and M_2 bound to $xtype_1$. Now this goal would fail because our resource-bounded system (which now supplies the definition of *mouse*) uses up its supply of buttons when providing an instance for M_1 and is incapable of supplying an instance for M_2.

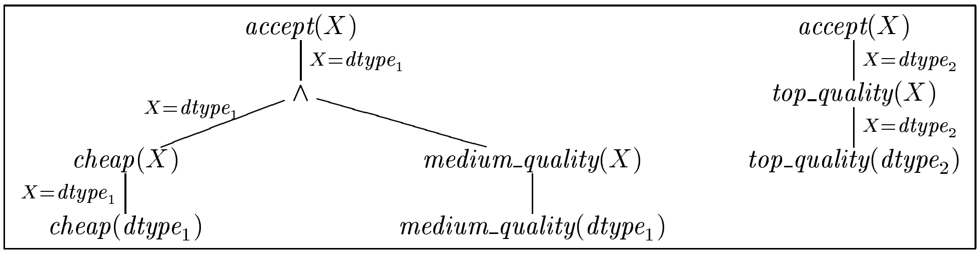
Question 5 *If we use as an oracle a system with more constraints on acceptable solutions than the system receiving its advice, are these all the solutions we need?*

4.3 Can One Inference System be Used as a Surrogate for Another?

In Section 4.2 we used the donor system as an oracle and therefore we had no opportunity to ensure that the inference it performed was consistent with the inference patterns known to the recipient. This meant that we had to tolerate various forms of potential discrepancy. If we are unable to tolerate these sorts of discrepancy then we may consider whether we can continue to use the inference method of the recipient system but with the donor supplying only a surrogate knowledge base, replacing the knowledge of the recipient at appropriate points during inference.

4.3.1 Surrogacy: Classical Specification Receives Fuzzy Requirements

Recall that in Section 4.2.1 we used our fuzzy system as an oracle to tell us whether components used in the specification system were acceptable. This meant that information coming from the fuzzy system was determined according to fuzzy inference rules, rather than the classical inference rules of the specification system. An alternative is to use the classical inference method alone but use appropriate fuzzy knowledge as a surrogate for missing classical knowledge. The translation required to do this is identical to that of Section 4.2.1 but the results we obtain are different because the fuzzy inference method requires exhaustive search of appropriate parts of its knowledge base while the classical inference method may only search part of it. For instance, if we assume that we set the threshold for μ in Section 4.2.1 to 0.5 and use the classical inference mechanism on the knowledge base from Section 3.2 to satisfy the goal $accept(X)$ then we can find the two proofs given below. Notice that these no longer contain fuzzy values because our translation mechanism converts these to a boolean truth value for the classical system:



By contrast, the fuzzy logic inference mechanism applied to the same goal gives us the single proof shown in Section 3.2. The advantage of using the classical inference method is that each individual proof may be shorter because it does not require exploration of all relevant knowledge. The disadvantage is that we no longer produce a single "most likely" result – instead we get all results which exceed the 0.5 translation threshold.

Question 6 *Will the amount of search performed by your inference strategy be sufficient when applied to a surrogate knowledge base designed for more extensive search?*

4.3.2 Surrogacy: Fuzzy Requirements System Receives Classical Specification

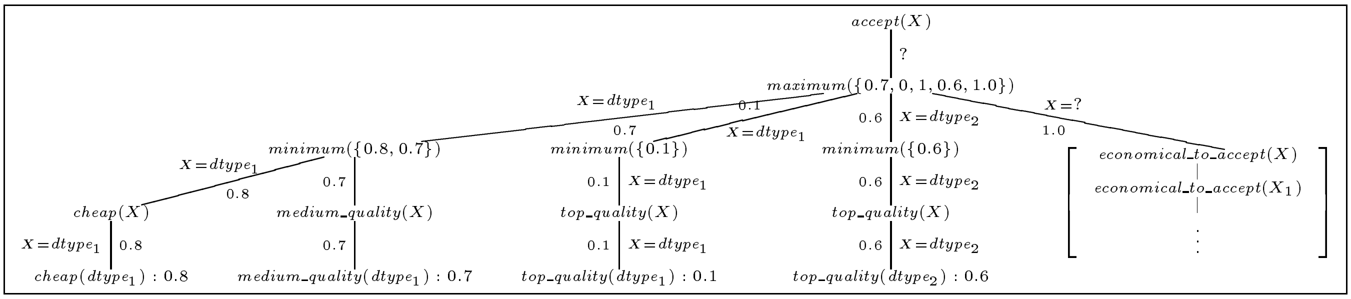
Using a classical inference method in the previous example allowed us to generate solutions based on partial searches of the available fuzzy knowledge. An alternative strategy if we wish to preserve the fuzzy system's exhaustive search is to use the fuzzy inference method with the classical knowledge. For example, we may wish to define acceptability requirements which are not fuzzy and are therefore more naturally expressed in a classical style. An example might be that we will accept a component which is similar to another one which we know is acceptable. Formally, we might represent this within the classical system as follows:

$$economical\_to\_accept(X) \leftarrow economical\_to\_accept(Y) \wedge similar(X, Y)$$

To allow this to be used by our fuzzy inference method we connect the *economical_to_accept* predicate to the *accept* predicate of the fuzzy system and translate from boolean to fuzzy truth values as before:

Expression in donor	Expression in recipient
$economical\_to\_accept(X)$	$accept(X)$
Bolean truth ($\top^C$)	Fuzzy value (1)

Unfortunately, this approach can create problems, as we can demonstrate by attempting a proof of *accept*(X). This is similar, at the beginning, to the proof in the stand-alone fuzzy system of Section 3.2. The important difference is when the connection between *accept* (in the fuzzy system) and *economical_to_accept* (in the classical system) is used to include the requirement given above in the fuzzy logic search (the part belonging to the classical system being enclosed in square brackets below). The problem with the new definition is that it is recursive, so our fuzzy logic inference mechanism will get stuck in an infinite loop when using it:



Question 7 *Will your inference strategy explore uncharted areas of the search space if applied to a surrogate knowledge base designed for less exhaustive search?*

4.3.3 Surrogacy: Resource-Bounded Assembler Receives Classical Specification

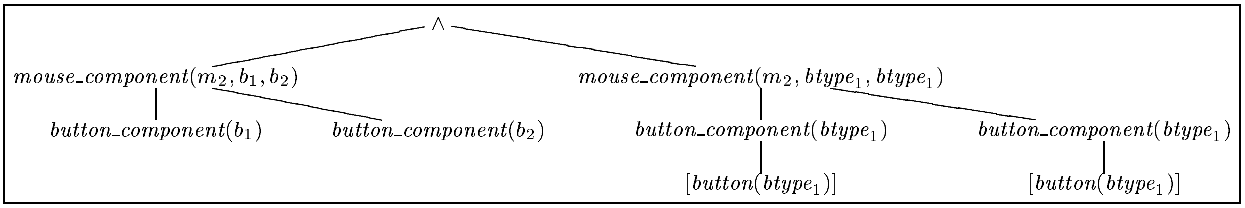
When using our classical specification as an oracle for the resource bounded system (see Section 4.2.3) we had the difficulty that the classical system could generate from its catalogue a limitless supply of (identically named) buttons. It could do this because its inference is not resource constrained. If we wish to impose resource constraints uniformly in this situation we can use the classical knowledge base as a surrogate for the resource-bounded inference mechanism. In this case, the translations would be identical to those of Section 4.2.3. However, the results of inference are different because uniform application of the resource constraint would prevent goals which succeed in the oracle version, such as $(mouse\_component(m_2, b_1, b_2) \wedge mouse\_component(m_2, btype_1, btype_1))$, from succeeding now because two uses of $button(btype_1)$ are no longer allowed.

Although easy to describe, this sort of application of proof constraints is not always easy to do. Three possible strategies are:

- We could simply import the rules from the donor system, translating them into the relevant format for the recipient's inference mechanisms. This is straightforward for our example because the notations of each system are similar. It is more difficult for dissimilar logics and, even if the translation is straightforward, may require us to import larger amounts of knowledge than we would like.

- We could be strict about how often we allow the recipient inference mechanism to access the donor knowledge base. For instance, in our example we could impose the resource constraint by never attempting the same goal more than once using the classical system. The problem with this strategy is that it loses information. For example, if the classical system contains the two identical facts $p(a)$ and $p(a)$ then a normal resource-bounded system should use both of them but our solution would, after succeeding with the first, disallow the goal which might succeed with the second.

- We could require that each donor system returns the proof tree describing how it found the conclusion to each given goal. This proof tree could then be checked retrospectively for compatability with the recipient inference strategy. For example, without the constraint the proof tree for constructing a mouse in the oracle style of interaction given in Section 4.2.3 is:



We can see that the last two steps in the proof, involving goals translated into the classical system (enclosed in square brackets) are duplicated and we would reject the proof from the point of view of the resource-bounded system. This approach is more expensive than the previous one because the donor system must record the proof and may have gone to a great deal of effort only to be rejected. However, it gives more information about the capabilities of the donor system, making it more likely that the recipient could make an informed choice.

Question 8 *If your inference mechanism places resource limits on proofs, how will these be applied to its surrogate knowledge base?*

4.3.4 Surrogacy: Classical Specification Receives Resource Bounded Assembly

In Section 4.2.4 we saw that using the resource-bounded system as an oracle supplying information to the classical specification system gave an unsatisfactory mixture of metaphors when constructing possible computer specifications. All components were treated as if they were constantly available except for the mouse which (because it was obtained from the resource-bounded system) was in limited supply. One way of lifting this restriction is by using the classical inference mechanism with the knowledge base of the resource-bounded system. The translations between notations remain the same as in Section 4.2.4 but we can now obtain as many copies of mouse components of type m_2 as we need - so the stock system is behaving like the catalogue system in this respect.

Question 9 *If your inference mechanism is not resource limited but its surrogate knowledge base is, does it matter that this restriction has been lifted?*

5 Conclusion

Often when we write a knowledge base we have in mind some inference strategy which will harness that knowledge to solve problems. This influences the way we describe the knowledge, so that our representation is not entirely neutral with respect to inference. Consequently, if we share knowledge bases without considering the interaction between the inference methods of the donor and recipient systems then we may produce results which are unexpected. This paper examines this sort of interaction with the aid of three simple logical systems and uses these to generate a series of generic questions which aspiring knowledge sharers might ask themselves if they plan to share knowledge between systems with differing inference methods.

Although we used logic to support our analysis, the generic questions are appropriate for knowledge based systems constructed by more conventional means. We chose logic because it gives a "best case" scenario, in which knowledge is declaratively expressed using a highly standardised notation. In other words, if we need to ask these questions for our simple, logical examples then you will be very likely to need to ask similar questions for your more complex systems.

Our paper raises questions but does not provide a computational architecture for solving them. One promising avenue of investigation, which we are beginning to study, is that the full semantics of systems of inference can often be divided into two components: the part which is responsible for connecting to the syntax of the knowledge base and the part which determines the control strategy used in problem solving. By distinguishing these, it may be possible to make control strategies more "mobile" between knowledge based systems and hence be more clear about when we are diverging from the inference strategy anticipated for a given knowledge base. Preliminary implementation of a knowledge sharing architecture which uses meta-interpretation to automate this method is underway.

References

[ADP96] J. Andrews, V. Dahl, and F. Popowich. Characterizing Logic Grammars: a Substructural Logic Approach. *Journal of Logic Programming*, 26:235–283, 1996.

[AEG+91] J. Agusti, F. Esteva, P. Garcia, L. Godo, and C. Sierra. Combining Multiple-valued Logics in Modular Expert Systems. In *VII Conference on Uncertainty in Artificial Intelligence*, 1991.

[AEG+94] J. Agusti, F. Esteva, P. Garcia, L. Godo, R. Lopez de Mantaras, and C. Sierra. Local Multi-Valued Logics in Modular Expert Systems. *Journal of Experimental & Theoretical Artificial Intelligence*, 6(3):303–321, 1994.

[BK92] A. J. Bonner and M Kifer. Transaction Logic Programming. Technical Report CSRI 270, University of Toronto, 1992.

[CdS91] F. S. Correa da Silva. Automated Reasoning About an Uncertain Domain. Technical report, University of Edinburgh, Department of Artificial Intelligence, 1991. A preliminary version of this work was presented at the European Conference on Symbolic and Quantitative Aspects of Uncertainty, under the same title, authored by Correa da Silva, F. S., Robertson, D., and Chung, P.

[CdSC94] F. S. Correa da Silva and D. V. Carbogim. A System for Reasoning with Fuzzy Predicates. Technical Report RT-MAC-9413, Instituto de Matematica e Estatistica, Universidade de Sao Paulo, 1994.

[CdSRH94] F. S. Correa da Silva, D. S. Robertson, and J. Hesketh. Automated Reasoning with Uncertainties. In M. Masuch and L. Pawlos, editors, *Knowledge Representation and Uncertainty*, chapter 5. Springer Verlag LNAI 808, 1994. Preliminary version presented at the Applied Logic Conference (Logic at Work), 1992.

[CdSVR98] F. S. Correa da Silva, W. W. Vasconcelos, and D. S. Robertson. Cooperation Between Knowledge-Based Systems. In *Proceedings of the IV World Congress on Expert Systems*, pages 819–825, Mexico City, Mexico, 1998.

[Gir87] J. Y. Girard. Linear Logic. *Theoretical Computer Science*, 50:1–102, 1987.

[Gra] P. (principal investigator) Gray. KRAFT – Knowledge Reuse and Fusion/Transformation. http:// www.csd.abdn.ac.uk/ apreece/ Research/ KRAFT/ KRAFTinfo.html.

[Gua98] N. Guarino, editor. *Formal Ontology in Information Systems.* IOS Press, 1998. ISBN 90-519-399-4.

[Lee72] R. C. T. Lee. Fuzzy Logic and the Resolution Principle. *Journal of the ACM*, 19:109–119, 1972.

[NG] R. Neches and D. Gunning. The Knowledge Sharing Effort. http://www-ksl.stanford.edu/knowledge-sharing/papers/kse-overview.html.

[REPG98] J. A. Reyes, F. Esteva, and J. Puyol-Gruart. Defining and Combining Multiple-Valued Logics for Knowledge-Based Systems. In *Proceedings of the First Catalan Congress on Artificial Intelligence*, pages 30–39, IIIA ,Campus UAB , 08193 Bellaterra, 1998.

[Sha83] E. Y. Shapiro. Logic Programming with Uncertainties - a Tool for Implementing Rule-based Systems. In *Proceedings of the 8^{th} International Joint Conference on Artificial Intelligence*, 1983.

[SSCdSR96] M. V. T. Santos, P. E. Santos, F. S. Correa da Silva, and M. Rillo. Actions as PROLOG Programs. In *IEEE Proceedings of Joint Symposia on Intelligence and Systems*, 1996.

[Sub] V. S. (project director) Subrahmanian. Hermes – a Heterogeneous Reasoning and Mediator System. http:// www.cs.umd.edu/ projects/ hermes/index.html.

[UG96] M. Uschold and M. Gruninger. Ontologies: Principles, Methods and Applications. *Knowledge Engineering Review*, 11(2):93–136, 1996.

[ZO89] C. Zhang and M. Orlowska. Homomorphic Transformation Among Inexact Reasoning Models in Distributed Expert Systems. Technical Report 135, Dept. of Computer Science, University of Queensland, 1989.