Developing Heuristics for the Argument Based Explanation of Negation in Logic Programs

T.J.M. Bench-Capon and P.H. Leng
Department of Computer Science
The University of Liverpool
Liverpool
England

September 17, 2015

EXTENDED ABSTRACT

1 Introduction

Dialogue games can be used in a variety of ways. [?] uses the notion of a dialogue game to provide a computational model of a particular variety of legal discourse, whilst [?] uses them to form the basis of a CAL system designed to teach people to argue more effectively. Our interest has centered on the possibility they give of improving interaction with knowledge based systems, in particular those based on a logic programming paradigm. In previous work [?], [?], [?], and [?]) the use of dialogues in explaining positive conclusions from the program has been explored and described. In this paper we address the more problematic task of explaining negative conclusions based on the concept of negation as failure as widely employed by logic programs - where the KBS denies a claim because no proof of the claim is possible on the basis of the knowledge available.

The game itself is based on a modification of Toulmin's argument schema [?], as described in [?], and depicted schematically in figure 1. The notion of the game we will use is of a co-operative rather than an adversarial game. [?] lists a number of dimensions of dialogue games, but the co-operation/competition dimension is not among them. It is, however, an important consideration if we want to design a system to play the game, since the strategy to be used, and the style of play, will be very different. Thus our model will be of an activity where the participants work as a team, such as bobsleighing, ice dancing or the childrens' game of pat-a-cake, rather than the more common picture of competitive games such as chess, rugby or boxing. Perhaps the best analogy of all is with a game like bridge, especially in the case of an uncontested auction where the two partners communicate using a set of conventions to reach the most favourable contract. In our view of

interaction with a KBS the participants similarly co-operate using the structure provided by the rules of the game to produce a convincing argument, in support of, or against, a particular claim.

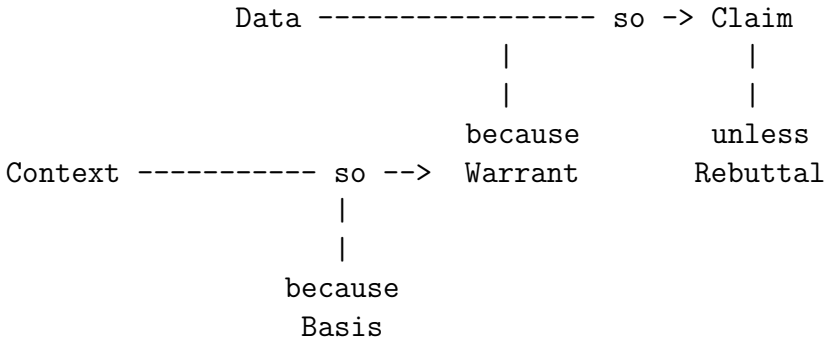

Figure 1: Modification of Toulmin's Argument Schema

2 Disputing a Claim

The nature of the play required when defending a claim is very different from that needed to dispute a claim. In the cases we have described in previous work, the KBS has a proof of the claim, and the purpose of the game is to construct a convincing argument. Here it is the human player's knowledge of what makes a convincing argument that dictates the play: the system is required only to respond to the human player's moves in accordance with the rules of the game. This greatly simplifies the implementation, since the computer player is in a forcing sequence, and so move selection is not a problem. Where, however, we cannot exploit a forcing sequence, move selection - which makes the difference between playing a legal game and playing the game well - becomes a crucial issue. Effective play will thus require that we have some means, such as rules or heuristics, of encapsulating the strategy and tactics needed to play the game well. Very little work has been done here: [**?**] discusses some heuristics for playing MacKenzie's DC, based on empirical observation of human players, but they are not entirely convincing.

When the claim is in dispute no forcing sequence is possible. The system rejects the claim because no proof of the claim exists, but an exhaustive report of all the reasons for failure would not provide a very convincing argument. Rather what is needed is a view as to the reason why the other player might believe the claim, and a demonstration that that reason is not applicable.

Two kinds of move are possible. Some are intended to elicit the reason that the other player believes the claim: once a view has been formed as to that reason, moves designed to show the inapplicability of that reason can be made. Heuristics are thus required to select the appropriate moves, and to determine when to switch from one line of play to the other. One way to discover such heuristics is to examine some simple examples and legal lines of play, so as to discover what principles lead to effective play.

3 Generating An Argument From Annotated Prolog Clauses

As a simple example consider a knowledge base with the following two rules:

```
old(X):-          man(X,class),
                  age(X, A, data),
                  greater_than( A, 70, cond),
                  not( born_in( X, tibet ), qual).

old(X):-          dog(X,class),
                  age(X, A, data),
                  greater_than( A, 17, cond),
                  not( bloodhound( X ), qual).
```

The annotations, which are fully described in [**?**], are intended to indicate the role played by the clause, and to relate this role to a role in the argument schema. Consider the first rule above, for example:

```
annotation    literal          role in argument

claim         old( X )         The head of the clause corresponds
                               to the claim.

class         man( X )         This defines the things for which
                               the warrant is applicable-
                               it corresponds to the context.

data          age( X, A )      This is data used to justify the claim.

cond     greater_than( A, 70)  This is a condition on a datum
                               and is part of the warrant.

qual     not( tibetan( X ) )   This defines an exceptional
                               condition when the clause is not
                               applicable - it corresponds
                               to a rebuttal.
```

The warrant will now be

```
Warrant          old(X) if age(X,A) and greater_than(A,70)
```

Note that this is the clause without class and qualification. This corresponds to our observation that rules are typically expressed in English in terms of a sortal concept "all men are old if they are over 70", not "all things are old if they are men and over 70". The basis will be

```
Basis          old(X) if man(X),age(X,A) and greater_than(A,70)
```

Note that this is the clause without qualification. Rules are normally expressed in a general but defeasible way: the defeasibility only comes under consideration where an exceptional case - represented by the rebuttal clause - is suspected. By executing annotated clauses through the meta-interpreter described in [**?**] we can generate a set of relations describing the argument in terms of our schema.

4 Presenting the Argument Against A Claim

Suppose we have facts

```
man(fred).
man(bert).

born_in(fred,uk).
born_in(bert,tibet).

dog(fido).

beagle(fido).

age(fred,55).
age(bert,87).
age(fido,15).
```

Fred fails the first rule because his age is too low, and the second rule because he is not a dog. Note, however, that if Fred were a dog, he would be an old dog. Thus Fred may be thought old because he is thought to have a different age, because the operative condition in the rule about men is thought to be less stringent, or because he is thought to be a dog.

Bert fails the first rule because he was born in Tibet. Like Fred he fails the second rule because he is not a dog, although if he were he would be an old one. Here a mistake as to his age is of no relevance; he must be thought old either because of ignorance as to his place of birth, or its relevance, or from a belief that he is a dog.

Fido fails the the first rule because he is not a man, but would fail by being too young even if he were a man. He fails the second rule because he is too young. Here then the mistake must be as to Fido's age, or to the threshold appropriate to dogs.

Different refuting behaviour is required in these three different cases. In the case of Fred we need to refine our view as to the nature of the mistake. Thus we should ask the user to provide the data on which he bases his claim. If he overestimates Fred's age, we can cite Fred's age as a refutation. If he is, however, correct as to Fred's age we may demand the warrant, (that is ask why the user believes that a man of 55 is old), and so see whether he is applying the wrong criterion to the correct sort, which can be refuted

by displaying the correct warrant, or the correct criterion to the wrong sort which can be refuted by stating that Fred is a man.

In the case of Bert we may immediately state the rebuttal that Bert is Tibetan, as there is a great likelihood of this being the source of error. If this does not cause the user to withdraw the claim, we know that the user must either be unaware that Bert is a man, or ignorant as to the effect of being born in Tibet. Asking the user to supply the warrant will discriminate the two cases.

In the case of Fido, the sortal has no relevance. We can therefore state Fido's age. If this doesn't lead to withdrawal of the claim, we should cite the warrant relevant to dogs.

One final case which is worth considering is where the information in the knowledge base is incomplete. In particular a claim may fail because the age, or the sortal is unknown. In such cases the appropriate action is to attempt to get this information from the user. Thus if someone claims that Rover is old, the system should find the sort of Rover, and then Rover's age, which may lead either to the claim being conceded, or for the situation to resemble one of the previously discussed cases. That a fact is missing as opposed to its absence designedly representing falsity can be detected, since all clauses annotated as data should always succeed, as their role is to bind a variable internal to the clause, and at least one class condition should succeed, as everything must belong to some type. Thus a claim should only be denied if an applicable warrant can be found (i.e at least one class clause succeeds), and the condition clause for that warrant fails (i.e. that data clauses, succeeds, but with the wrong value). Otherwise the relevant information must be obtained before a judgement on the claim can be pronounced.

5 Conclusion

Thus even in this very simple example we can see that a range of different behaviour is needed, depending on the nature of the claim to be denied, the annotations on the rules, and the rules themselves. Good play of the game will require the ability to exhibit this flexibility of behaviour. The bulk of the paper will begin to develop heuristics, exploiting the annotations on the program which convey extralogical information as to the role and status of the various clauses. We will then show how these heuristics provide this flexibility in the above and more extensive examples, and so point the way towards developing a general set of heuristics for explaining negation in annotated logic programs.

References

[1] Trevor J.M. Bench-Capon, Paul E.S. Dunne and Paul H. Leng, (1991) 'Interacting With Knowledge Based Systems Through Dialogue Games', 11th Annual Conference on Expert Systems and Their Applications, Avignon, 1991, pp123-130.

[2] Bench-Capon, T.J.M., Lowes, D, and McEnery, A.M. (1991), 'Using Toulmin's Argument Schema to Explain Logic Programs'. Knowledge Based Systems, Vol 4, No 3, pp 177-183, 1991.

[3] Bench-Capon, T.J.M., Dunne, P.E., and Leng, P.H., (1992) 'A Dialogue Game For Dialectical Interaction with Expert Systems', Proceedings of the 12th International Conference on Expert Systems and Their Applications, Avignon, 1992.

[4] Bench-Capon, T.J.M., Coenen, F.P., and Orton, P., (1993) 'Argument Based Explanation of the British Nationality Act as a Logic Program', Computers, Law and AI, vol 2 No 1, 1993, pp 53-66.

[5] Gordon, T., (1993) 'The Pleadings Game', PH. D Thesis, Technischen Hochscule Darmstadt.

[6] Moore, David, (1993) 'Dialogue Games and Computer Aided Learning', PhD Thesis, Leeds Metropolitan University.

[7] Toulmin, S., (1958), 'The Uses of Argument', Cambridge University Press, 1958.