PLAID - Proactive Legal Assistance

T.J.M. Bench-Capon and G.Staniford

Department of Computer Science
The University of Liverpool
Liverpool
England

ABSTRACT

This paper describes PLAID, a system designed to support the preparation of a brief concerning a legal question. The system provides a multi agent framework in which ideas from argument based explanation, dialogue games, document modelling and conceptual retrieval are integrated into a tool capable of drawing on a variety of sources to produce a coherent argument for a particular point of view. A prototype, illustrating the functionality and components of the system and implemented in PROLOG, is described.

INTRODUCTION

One very common task, in many walks of life, is the preparation of a brief. A brief is required when a senior person in an organisation must pronounce on some question. Because that person will not have the time available to do all the necessary research for that question, a junior assistant is required to prepare a discussion of the issues, which can then be used, or adapted, by the senior. Typically the preparation of a brief requires examination of a variety of different source materials which bear upon the question, identification of the issues involved, and the construction of one or more arguments which support a view of these questions. Essentially, then, a brief is a coherent argument for a position with respect to some question.

The importance of this activity in law is well known. Much of the work of Rissland and her collaborators has been directed towards just this task (Skalak and Rissland 1991, and Rissland et al 1993 may be taken as representative). Also well known is the need for the reasoning from a legal knowledge based system to be presented in the form of an argument, rather than as a bare answer, or as a standard "how" type expert systems explanation (see, for example, Bench-Capon et al 1993). The aim of the PLAID project is to explore the task further. Finally the interactive nature of the process whereby the argument is constructed emphasises the view of law as a process, rather than simple classification, and idea which is currently growing in importance.

Key features of PLAID are:

- Use of the sources with the minimum of adaptation. Too often the scalability of systems for conceptual retrieval is jeopardised by the sheer amount of analysis required (For an extreme example see Dick 1992). In PLAID available sources can be used as they stand, although the construction, or adaptation, of a KBS may be required.
- A highly modular design based on a multi agent architecture. This allows for the integration of separately developed components, and for the gradual refinement and replacement of components whilst maintaining the coherence of the overall system.
- Use of a knowledge based system to provide the raw materials for the actual building of an argument and to provide a structure for the later stages of the interaction, and dialogue game techniques for the construction of the argument and access to other information sources. This enables the separation of the identification of argument components from the moves that are made in forming the argument, and provides the user with a single interface to all the information sources that need to be accessed.
- Use of document preparation techniques to present the argument in the form of a coherent text with the required structure.

The specific task we have chosen for our work is that performed by Government officers, when briefing their superior on a legal point, whether for the purposes of answering correspondence, Parliamentary Questions, or an Adjournment debate. We have selected this specific task because it is one we have ourselves performed: there may or may not be differences in the way in which trained lawyers go about such matters. Focus on this task has the advantage that whilst it is clearly a legal task it is one that is performed by lay people, avoiding anything specifically legal in the reasoning, and so the findings can be readily transferred to brief construction in other domains.

We have prototyped PLAID using as our domain the fictional Poor Law described in (Bench-Capon 1991). This domain is at once rich enough to exhibit all the features that we wish to explore (or if not, could be extended to become so), and small enough to be completely represented in a tractable fashion, so that problems of representing the domain do not distract us from the main focus which is on the components of the PLAID architecture.

INFORMATION SOURCES

In general, a person creating a brief on a legal question will have a number of information sources available. Common examples are:

- legislation; the Acts and Statutory Instruments which form the legal framework. For many areas of law this material is currently available in the form of a free text database

- case law; again while printed reports are the traditional form, on-line access to cases is now widespread;

- commentaries; giving guidance on interpretation. Normally this is available only in books. In the case under consideration, however, this is not the case, since the Adjudication Officer's guide, which is the main commentary on Social Security legislation is available on-line;

- the facts of the case; normally a form, or a written account of an interview with the client;

In addition the briefer must bring to bear knowledge: this knowledge will relate to several aspects: the law itself, the form in which the brief is required, the notion of what constitutes a persuasive argument, the level of detail appropriate to various aspects and the like.

In our particular prototype the information sources are:

- A Knowledge Based System containing a formalisation of the domain as described in (Bench-Capon 1991). Additionally this has been annotated in the style of (Bench-Capon et al 1991);
- A database containing the pertinent legislation;
- A database containing the leading cases;
- A database containing personal information regarding births, marriages and deaths;
- A database containing the commentary information that would be found in the Adjudication Officers Guide (AOG);
- Documentation for the system which supplies an English language template for each predicate used by the KBS;
- A file containing the information that would be provided the claim form;
- A set of schemata for use by the report generator.

ARCHITECTURE OF PLAID

The overall architecture of PLAID is shown in Figure 1. The agents which access the various information sources will now be described in some more detail.

Argument Meta Interpreter

The argument meta-interpreter is that described in (Bench-Capon et al 1991). The form of argument that we use is based on the schema of Toulmin (1958), modified in accordance with the revised argument schema of (Bench-Capon et al 1991). This schema is shown in Figure 2.

The function of the meta-interpreter is to execute the top level goal - the question for which the brief is being prepared - and to solve this goal with respect to the knowledge base and the assembled case facts. It uses the annotations on the various clauses to create a set of relations describing the solution in terms of nodes in the graph corresponding to the representation of the argument as a set of schemata of the form of Figure 2. This set of relations then provides the raw material for playing the dialogue game in the next stage of the process.

Dialogue Manager

The dialogue game here is largely that described in (Bench-Capon et al 1992). The status of the participants is, however, a little different. In the general game both participants have equal status, but here the system is able to supply a complete set of arguments derived from the KBS, whereas the human participant is seeking to elicit these arguments. Thus in this situation the moves made by the human participant will be designed to elicit pieces of the argument graph, whereas the moves made by the computer participant will be to supply those pieces, on the basis of the KBS output. This facilitates matters in two ways: first the move by the computer participant will be forced by the human's moves, so that no game playing strategy is demanded of the computer player. Second we can be confident that the computer participant will always be able to respond, since the whole graph is available to it.

The purpose of this stage is to use the human player's knowledge of what makes a good argument to select from the whole graph, which will, like all expert systems explanations, contain more material than is required, since it has explanations for things which need no explanation, a subgraph that constitutes a convincing and pertinent argument. The human may examine any node, but only selected nodes, those considered to be germane will be passed to the argument constructor agent.

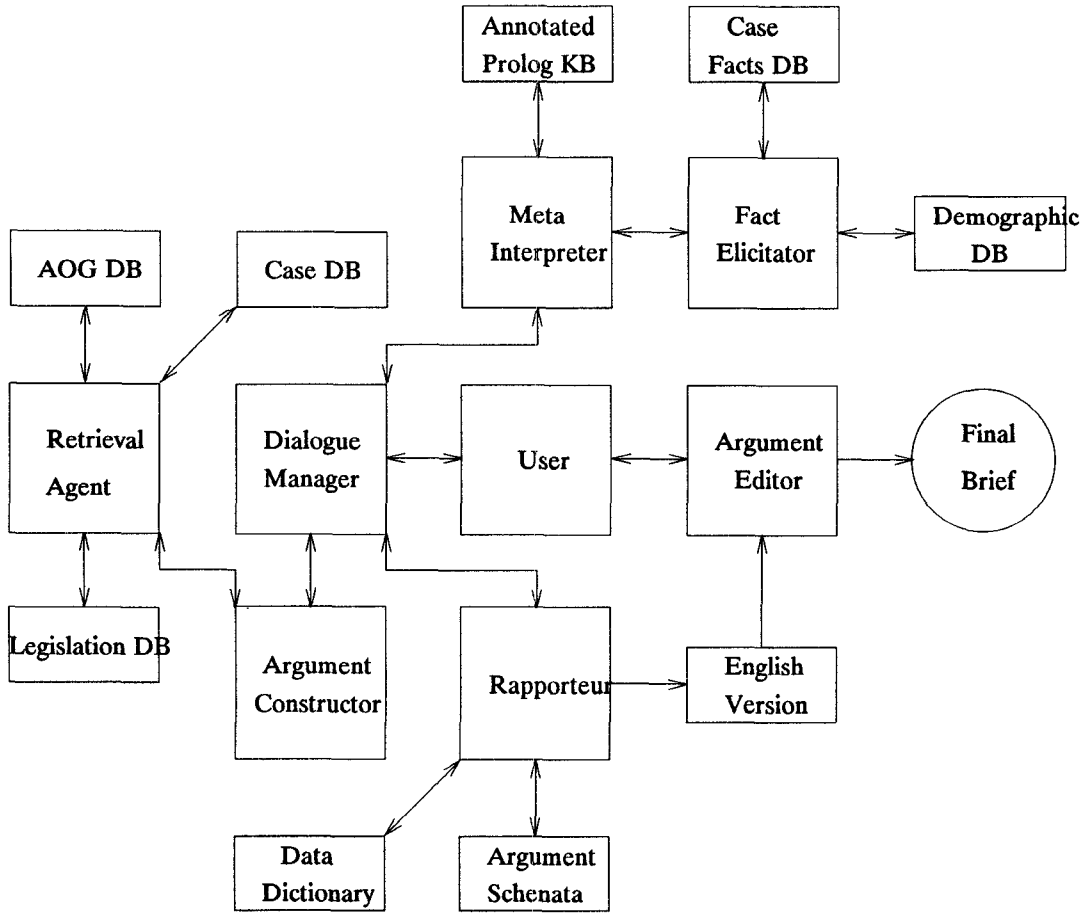

Figure 1: Overall Architecture of PLAID

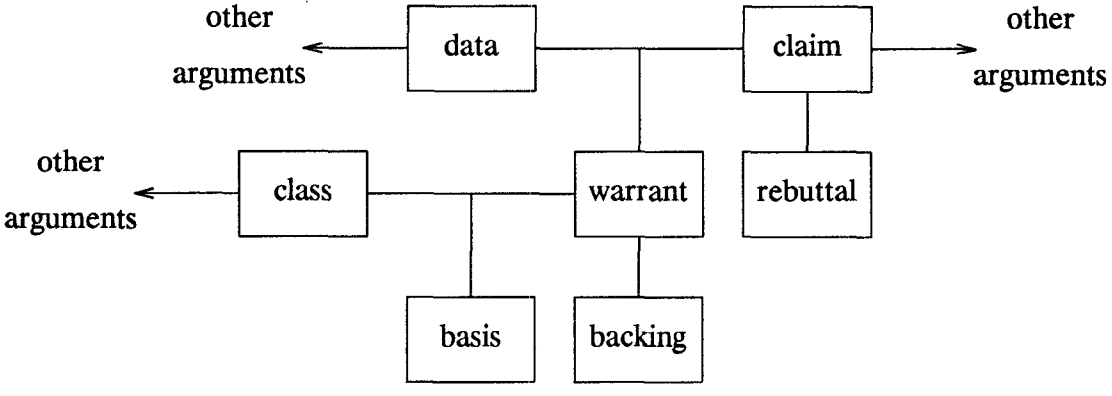

Figure 2: Modified Argument Schema

The new variation here is the range of options that are allowed when a backing node is examined. The KBS has been developed in accordance with the spirit of so-called "isomorphism" (Bench-Capon and Coenen 1992). The importance of this for our purposes is that the backing of the rules can be a reference to a single source fragment, whether a subsection of legislation, a case, or a paragraph from the AOG. At this point the user may or may not be familiar with the reference, and may or may not accept that the rule which has been derived from it for use in the KBS is a satisfactory interpretation. If the user wishes to see the source item, the dialogue is directed towards the retrieval agent, which when supplied with the reference from the backing node will be able to supply the text of the reference, by forming and transmitting an appropriate query to the relevant database. Having seen the text the user may wish to explore the sources further: where the text contains a reference to a case, item of legislation or AOG paragraph, that too may be retrieved. The dialogue with the retrieval agent will continue until the user has assembled enough source fragments to provide allow a justification for the rule to be produced. The selected source fragments can then be passed to the argument constructor agent.

The importance of this step is that the appeal to authority represented by the backing node, may not be sufficient to convince. Often to be persuaded by an interpretation it is necessary to go through that process of interpretation oneself. By making the source materials available in this way, this is made possible.

Note that in our prototype, we have restricted queries to the databases to be explicit references. More sophisticated retrieval agents is one line of possible future development.

The source materials passed to the argument constructor may be just references, or the full text. or for cases, since the full text may be lengthy, the head-note. The user will mark the fragments according to whether they support the interpretation or represent the foundation for an alternative (although rejected) interpretation, to signal the manner in which they should be included in the final brief.

Thus at the end of this stage we have constructed an argument graph which represents a sub-graph of the argument graph produced by the KBS, augmented as appropriate by attaching source fragments to the backings, so that the validity of the interpretation embodied in the KBS can be assessed. This refined argument graph provides the input to the Rapporteur Agent.

RAPPORTEUR

Rapporteur has been described in (Staniford 1994). Whilst the graphical form is a useful tool for working on building the argument, it is not the form in which we wish the final brief to appear. The role of Rapporteur, therefore is to transform the refined argument graph into something with the structure appropriate to the brief, in some semblance of natural language, so that the final production of the brief by the user is a matter of polishing rather than drafting.

Rapporteur makes use of two types of templates. The high level templates organise the different argument graph structures into a coherent paragraph of argument. The lower level templates supply English renditions of the predicates of the KBS, so that the nodes containing predicates can be expanded.

When Rapporteur has finished its transformations, the raw argument is loaded into either an ordinary text editor so that the user make make any alterations that are required or into the purpose built editor supplied with PLAID that provides sophisticated enhancements for the creation and manipulation of hypertext documents and the cooperative editing of texts.

AGENT ARCHITECTURE

The PLAID system contains logic based agents that have been specified according to the agent architecture fully detailed in (Staniford, 1993) and based upon work in (Genesereth & Nilsson, 1987). This is a layered model of agent societies and the behavioural aspects of such a model are based upon normative (or deontic) logic (Staniford, 1994). Agents are partitioned within the system into four layers according to their inherent abilities; with the layers corresponding to approximately equal divisions of a continuum ranging from deliberative agents through to purely reactive agents. All such agents we view as atomic agents.

The key idea in defining such agents is the use of an automated inference method such as resolution and the use of the Horn clause subset of predicate calculus to form the basis of a mechanised reasoning ability with a form of deontic logic. In modelling the control of a complex virtual structures (e.g. argument graphs, report graphs, etc.) in a mathematical space we use two primary mechanisms to impose social structures upon autonomous agents: networks of cooperating equals and hierarchies supporting the division of responsibilities. The vocabulary of an agent may be used to prescribe which other agents are allowed to meaningfully communicate with it and we can use the computational ability of an agent to implement cooperation strategies that enforce a hierarchy of control.

Atomic agents are agents of any type that cannot be sub-divided into simpler agents. Complex agents, on the other hand, are agents that are composed using at least two atomic agents from more than one level in

the four layer model and we can say that every complex agent is architecturally a layered agent. In a complex agent atomic agents will use preemptive cooperation in a downward direction—we use the term preemptive cooperation to allow for the situation in which lower level agents may refuse to accept a commission if they are already engaged upon an action commanded by some other higher level agent; this approach allows high level agents to share resources—which leads to the principle that a complex agent embodies within it the notion of a hierarchical control structure. A number of agents on the same level cooperating upon a common task may not be described as a complex agent, such a grouping is a set of autonomous communicating agents, that we might describe with collective nouns such as team, society, etc. From the foregoing it follows that whichever level the highest level agent in a complex agent is drawn, there will be only one agent from that level in the complex agent and we cannot combine groups of lower level agents into complex agents without the formation of a higher level agent, therefore agents from the highest level may only be grouped together to form teams etc.

Consider Figure 1., the agents shown in the figure are all high level agents (low level agents have been omitted for the sake of clarity) that fall into the two classifications of atomic and complex. The dialogue manager and argument constructor agents are both atomic agents, they interact with other agents and do not need to be aware of any low level environmental detail. On the other hand all other agents in the figure are complex agents in which the lower level agents (not shown) are responsible for the direct interaction with the detail of the environment upon and within which the agent society is operating.

EXAMPLE EXECUTION OF THE PROTOTYPE

The knowledge base used is a version of that used in Bench-Capon (1991), annotated in accordance with (Bench-Capon et al 1991). The code is given at Appendix A for anyone who may wish to work through the following example. The only requirements on the KBS are that it be "isomorphic", in the sense that there is a single identifiable source to serve as backing for each clause, and that it have the requisite annotations. Thus we would expect that any existing knowledge base which meets this requirement could be readily adapted to use in PLAID, and that if none exists, one can be straightforwardly produced.

Suppose that the following are facts:

```
isABeadle(bumble).
appliesTo(bumble,X):- not died(X,D).
thisYear(1594).
married(harry,mary,1558).
died(harry,1582).
```

Now suppose we wish to brief someone on Mary's claim. The following set of relations will be generated by the meta-interpreter:

```
claim(a(1), providedWithParishRelief(mary)).
claim(a(2), thinksDeserving(bumble, mary)).
claim(a(3), knownDeserving(mary)).
claim(a(4), widow(mary)).

data(a(4), married(harry, mary, 1558), fact).
data(a(4), died(harry, 1582), fact).
data(a(3), widow(mary), a(4)).
data(a(2), knownDeserving(mary), a(3)).
data(a(1), thinksDeserving(bumble, mary), a(2)).

warrant(a(4),
   [widow( 'X'),if,married( 'Y',  'X',  'D'),
   died( 'Y',  'D2')]).
warrant(a(3), [knownDeserving( 'Y'),if,widow( 'Y')]).
warrant(a(2), [thinksDeserving( 'X',  'Y'),
   if,knownDeserving( 'Y')]).
warrant(a(1),
   [providedWithParishRelief( 'X'),
   if,thinksDeserving( 'Y',  'X')]).

rebuttal(a(4), not remarried(mary, X, A), fact).

basis(a(1),
   [providedWithParishRelief( 'X'),if,isABeadle( 'Y'),
   appliesTo( 'Y',  'X'),thinksDeserving( 'Y',  'X')]).

presupposition(a(1), isABeadle(bumble), fact).
presupposition(a(1), appliesTo(bumble, mary), fact).

backing(a(4), [definition,widow,1]).
backing(a(3), [case,quickly,1]).
backing(a(2), [guidance,general,1]).
backing(a(1), [legislation,poorLawAct1561,1]).
```

DIALOGUE

The dialogue now begins with the top level claim, the claim of argument a(1):

```
providedWithParishRelief(mary)).
```

The user will want to know the grounds for this and so will seek the data for a(1). This will be supplied:

```
thinksDeserving(bumble, mary)
```

If the user is familiar with the legislation, this may suffice, but a full explanation will require that the presupposition is sought:

```
isABeadle(bumble)
appliesTo(bumble, mary)
```

Next the user demands the backing:

```
[legislation,poorLawAct1561,1]
```

In order to see the content of the legislation the reference is passed to the retrieval agent, which retrieves the required text:

```
Poor Law Act 1561

(1). Any person shall be provided with parish relief if
   he apply to a beadle of the parish and in the opinion
   of that beadle he is deserving of such relief.
```

The user now turns attention to a(2), and demands the grounds for thinksDeserving(bumble, mary). The data for this claim is supplied:

```
knownDeserving(mary)
```

Recognising that this serves only to indicate that Bumble is acting on policy rather than on his own discretion, the user decides not to pass these nodes through to the final argument, and instead examines a(3), asking for the grounds for the claim knownDeserving(mary). These are

```
widow(mary)
```

The user seeks authority for this argument by asking for the backing:

```
case,quickly,1
```

This reference can now be passed to the retrieval agent, which will recall the headnote.

```
Quickly r1-72

It was held that
Mistress Quickly is a widow robbed of the husband
to whom she will have expected to look for support
Of all people widows and orphans are most deserving
of our compassion and so most deserving of relief
```

Finally justification for Mary being found to be a widow is sought. First the data for a(4) is requested:

```
married(harry, mary, 1558)
died(harry, 1582)
```

This argument has a potential rebuttal:

```
remarried(mary, X, A)
```

This concludes the exploration of the argument, and the selected nodes are passed to the Rapporteur agent.

ENGLISH RENDITION

Rapporteur now generates the following output:

> mary should be provided with Parish Relief because bumble thinks that mary is deserving This presupposes that bumble is a properly appointed beadle and mary has made an application for Parish Relief to bumble
>
> The justification for this is that it follows from the Poor Law Act 1561 which states that:
>
> (1). Any person shall be provided with parish relief if he apply to a beadle of the parish and in the opinion of that beadle he is deserving of such relief
>
> mary is known to be deserving because mary is a widow The justification for this is that it was established by the quickly case in the case of quickly r1-72 it was held that:
>
> Mistress Quickly is a widow robbed of the husband to whom she will have expected to look for support Of all people widows and orphans are most deserving of our compassion and so most deserving of relief
>
> mary is a widow because harry and mary were married in 1558 and harry died in 1582 Provided that it is not the case that mary has remarried

This can now be passed to an editor so that the user can make any desired stylistic changes. Currently, because of some rough edges in the Rapporteur module, the argument will at least need to be repunctuated and recapitalised, but the above suffices how the information from several sources has been brought together and structured.

CONCLUSION

Clearly what has been described in this paper is only a prototype, and a vehicle for further exploration. Most of the agents require further refinement. Some of the chief desired enhancements are as follows:

- Meta Interpreter: Currently the meta-interpreter works very well for positive demonstrations of a proposition, and has performed effectively in several projects. For full generality, however, it needs also to be able produce arguments *against* a proposition. One approach (used in Denton 1994) is to modify the knowledge base, effectively explicitly supplying the completion of the knowledge base. Whilst this works reasonably well, it does require additional work in constructing the KB. An alternative, sketched in (Bench-Capon and Leng 1994) is to extend the meta-interpreter to produce

arguments from the failure to show a proposition from the KB. We feel that the latter approach is the more attractive.

- Dialogue: The Dialogue agent could be extended in two ways. First, if the system is to argue effectively against propositions, heuristics, along the lines of those suggested in (Bench-Capon and Leng 1994) will be required. More ambitious is the idea that if heuristics could capture the game playing strategy of the user in the system described above, support for the traversal of and selection from the full argument graph might be automated. If this proved possible, PLAID would move from being a support tool to a system capable of a first cut execution of the briefing task.

- Retrieval: Currently we retrieve from the various external databases only on explicit references. Work is required to provide additional flexibility: obviously this will depend on the nature of the database to be accessed, and each new database will need to be treated on its merits, and a customised retrieval agent developed for it.

- Rapporteur: Work is needed here on extending the range and flexibility of the templates, both with respect to the arguments graphs, and with respect to the incorporation of material retrieved from external sources.

The PLAID system described here is in its very early stages. We believe, however, that it provides a framework for tackling an important and pervasive problem, and that this framework therefore has considerable potential.

REFERENCES

T.J.M. Bench-Capon, *Using a Common Knowledge Base in Different Applications* in T.J.M. Bench-Capon (ed), *Knowledge Based Systems and Legal Applications,* Academic Press, 1991, pp129-136.

T.J.M. Bench-Capon, D.Lowes and A.M. McEnery, *Using Toulmin's Argument Schema to Explain Logic Programs.* Knowledge Based Systems, Vol 4 No 3, September 1991, pp177-83.

T.J.M. Bench-Capon, P.E.S. Dunne and G.Staniford, *RAPPORTEUR: From Dialogue to Document* Proceedings of the Fourth Annual Conference on Computers and the Writing Process, Computers and Writing Association, Edinburgh, 1991, pp175-183.

Bench-Capon, T.J.M., Dunne, P.E. and Leng,P.H., *A Dialogue Game for Dialectical Interaction with Expert Systems* 12th Annual Conference on Expert Systems and Their Applications, Avignon, 1992.

Bench-Capon, T.J.M., and Coenen, F.P., *Isomorphism and Legal Knowledge Based Systems,* AI and Law, Vol 1, No 1, pp65-86, 1992

Bench-Capon, T.J.M., Coenen, F.P., and Orton, P., *Argument Based Explanation of the British Nationality Act as a Logic Program,* Computers, Law and AI, vol 2 No 1, 1993, pp 53-66.

Bench-Capon, T.J.M., and Leng, P.H., *Developing Heuristics for the Argument Based Explanation of Negation in Logic Programs*, Proceedings of the AAAI Workshop on Computational Dialectics, Seattle, 1994.

Denton, P.D., *An Expert System in Law* B.Sc Dissertation, Department of Computer Science, University of Liverpool, 1994.

Dick, J.P., *A Conceptual, Case Relation Representation of Text for Intelligent Retrieval,* Technical Report CSRI-265, Computer Systems Research Institute, University of Toronto, 1992.

Rissland, E.L., Skalak, B.B., and Timur Friedman, M., *BankXX: A Program to Generate Argument Through Case Based Search* in Proceedings of the Fourth International Conference on AI and Law, Amsterdam, ACM Press, 1993.

Skalak, D.B., and Rissland,E.L., *Arguments and Cases: An Inevitable Intertwining* Artificial Intelligence and Law, 1, pp3-48, 1992.

Staniford, G., *Multi Agent Systems in Support of Co-Operative Authorship,* PhD Thesis, University of Liverpool, 1994.

Toulmin, S., *The Uses of Argument,* Cambridge University Press, 1958.

APPENDIX A

Knowledge Base Used in the Prototype

```
[legislation, poorLawAct1561, 1]:
    providedWithParishRelief(X):-
       isABeadle(Y) : class,
       appliesTo(Y,X) :class,
       thinksDeserving(Y,X) :data.
[guidance, general, 1]:
    thinksDeserving(X,Y):-
        knownDeserving(Y):data.
[guidance, general, 2]:
    thinksDeserving(X,Y):-
       not (knownUndeserving(Y)) :qual,
       looksDeservingTo(X,Y) :data.
[case, quickly, 1]:
    knownDeserving(Y) :- widow(Y) : data.
[case, quickly, 2]:
    knownDeserving(Y) :- orphan(Y) : data.
[case, goodbody, 1]:
    knownUndeserving(Y):-
        canMakeOwnShift(Y):data.
[case, goodbody, 2]:
    canMakeOwnShift(Y):-sturdy(Y): class,
        vagabond(Y) : data.
[guidance, goodbody, 1]:
    canMakeOwnShift(Y):-
        male(Y): class,
        health(Y,good): data,
        age(Y,A): data,
        A < 65 : cond.
[guidance, goodbody, 2]:
    canMakeOwnShift(Y):-
        health(Y,good): data,
        female(Y): class,
        age(Y,A): data,
        A < 60: cond.
[guidance, quickly, 1]:
    orphan(Y):- age(Y,A): data,
        A < 14 : cond,
        not ( parent(Y,X) , living(X)): class.
allowableDiscretion2:
    looksDeservingTo(X,Y):-
        deemedDeserving(Y): data.
[guidance, general, 3]:
    looksDeservingTo(X,Y):-
        not (deemedUndeserving(Y)): qual,
        judgedDeservingBy(X,Y): data.
[guidance, sickness, 1]:
    deemedDeserving(Y):-
        hasCertificateOfUnsturdiness(Y,C):data,
        issuedBy(C,P): data,
        registeredPhysician(P): cond.
[guidance, wives, 1]:
    deemedUndeserving(Y):-female(Y): class,
                    married(Y): data.
[guidance, children, 1]:
    deemedUndeserving(Y):-age(Y,A): data,
        A < 14: cond,
        parent(Y,X): class,
        living(X): class.
[register, birth, 1]:
    male(X):-born(X,D,boy,F,M): data.
[register, birth, 2]:
    female(X):-born(X,D,girl,F,M): data.
[register, birth, 3]:
    age(X,A):-born(X,D,S,F,M):data, thisYear(Y):
        data, A is Y - D: cond.
[definition, widow, 1]:
    widow(X):-married(Y,X,D) :data,
        died(Y,D2): data,
        not (remarried(X,Y2,Z)): qual.
[register, marriage, 1]:
    married(X):-married(X,Y,D): data,
        not (died(Y,D2)): qual.
[register, marriage, 2]:
    married(X):-married(Y,X,D): data,
        not (died(Y,D2)): qual.
[register, marriage, 3]:
    remarried(X,Y,Z):- married(Y,X,D) : class,
        married(Z,X,D2) :data,
        not Y == Z: class,
        D2 > D: cond.
[register, death, 1]:
    living(X):-born(X,D,S,F,M): data,
        not (died(X,D2)): qual.
[register, marriage, 4]:
    parent(X,Y):-born(X,D,S,Y,M) : data.
[register, marriage, 5]:
    parent(X,Y):-born(X,D,S,F,X) : data.
[assumption, of, health]:
    health(X,good):-
        not (hasCertificateOfUnsturdiness(X,C)):
    qual.
[bumblesOpinion]:
    judgedDeservingBy(bumble,X):-age(X,A):data,
        A > 80 :cond.
```