INTERACTING WITH KNOWLEDGE BASED SYSTEMS THROUGH DIALOGUE GAMES

Trevor J.M. Bench-Capon Paul E.S. Dunne and Paul H. Leng

Department of Computer Science
University of Liverpool

ABSTRACT

Approaches to text generation have emphasised the need to plan dialogues between user and computer, but have paid insufficient attention to the highly conventional nature of much human-human interaction. It is possible to conceive of certain classes of dialogues as highly rule-governed activities, and to do so facilitates the planning of such dialogues, by decreasing the reliance on a model of the user. One such class of dialogue is interactive argument, in which a topic is discussed with the aim of reaching a conclusion as to the truth of some proposition. We contend that interactive argument provides a useful model of interaction with a KBS, could provide a novel means of knowledge elicitation, and would serve as a fruitful model for educational systems also. Details of a dialogue manager based on a well known dialogue game (Mackenzie's DC) are given. Since KBS will exist independently of this way of interacting with them we give details of a Dialogue Abstract Machine (DAM), which could serve as a target interface between our dialogue manager and a KBS of any formalism.

June 7, 101

INTERACTING WITH KNOWLEDGE BASED SYSTEMS THROUGH DIALOGUE GAMES

Trevor J.M. Bench-Capon Paul E.S. Dunne and Paul H. Leng

Department of Computer Science
University of Liverpool

Introduction

Knowledge Based Systems (KBS) represent a considerable repository of information and know-how about a variety of matters. There are, however, considerable problems in designing useful and effective interactions with such systems. To date the "consultative" model has dominated, whereby the user poses a question, is interrogated by the system so that it can obtain pertinent facts, and then receives an answer. Usually too, some highly stylised "explanation" facilities are provided, so that the user may be told why a question has been asked and how a conclusion was established. This style of interaction has its uses, but there is a disappointing lack of any notion of intelligence in such an interaction. The intelligence of the user is not harnessed at all, and the behaviour of the system is much the same as would be got from any conventional program performing the task. In order to introduce some flavour of intelligence into such dialogues, some researchers have worked on the generation of multi-sentence paragraphs of text. An application of this work to KBS would be to give better explanations, thus improving an interaction within the consultation paradigm by providing more satisfying answers to the users questions. There are currently two dominant approaches to text generation: the use of "schemata" which are intended to capture and exploit the structure of naturally occurring texts[1], or "Rhetorical Structure Theory" (RST)[2], which identifies the relations which can occur between sections of a text, together with the ends that these relations achieve in the text. These relations can be used as the basis of a planning operator[3]. The problems with schemata centre on the inability of the approach to explain why texts have the structure attributed to them, and the consequent failure to generalise to other types of text, requiring an exhaustive analysis for every different text type. RST tends to treat relations which are of different types as if they were homogenous, whereas since some will be, for example, due to relations between concepts deployed in the text and others due to personal writing style, a uniform treatment is not really appropriate. Both approaches tend to emphasise the coherence of the text at the expense of a genuinely explanatory account of the structure of texts. It is as if an observer of a football game noted that the ball crossing the goal line was always followed by a kick-off, a corner or a goal kick. Whilst this would provide a coherent basis for simulating a football game, understanding of the rules which give rise to these patterns is necessary to explain why each follows a particular occurrence (supposing the referee to be sighted). In this paper we will attempt to use a rather more explanatory notion of dialogue structure, as the basis of a novel model for interaction with a KBS.

The Rule-Governed Nature of Dialogues

We take as our starting point for explaining the structure of dialogues, the highly conventional nature of much of human discourse. Consider the following dialogue fragment representing a greeting:

A: Hello. How are you?

B: Not so bad. How are things with you?

A: OK. Now about ...

Here neither the questions nor the answers given to them can be understood by analysing the meaning of the words. The inquiry in A’s initial utterance is not seeking to elicit information about B’s health, nor should B’s response be taken as indicating that he is not ill in any way. A serious answer from B would be a breach of the convention which governs this style of greeting. The rules require that a general enquiry as to health or things in general, ("How are things?", "All right?") be followed by a non-committal reply

("OK", "Mustn't grumble"), with a similar general enquiry, followed by a similar non-committal answer before the meat of the discourse can be reached.

There are a number of philosophical theories of meaning. One model of meaning suggests that words have a meaning because they denote a concept, and an utterance is planned by combining these meanings to form a desired composite meaning. A contrasting view, found in the later writings of Wittgenstein[4], is that words do not have meaning in virtue of standing in such a relation to a concept, but as a result of their capacity to be used in a certain range of *language games.* Wittgenstein uses the term "language game" in a somewhat equivocal manner, sometimes to designate simplified forms of language, and at others the whole of language and the activities with which it is interwoven. Often it will designate a a fragment of language: sometimes a specific speech act, and sometimes general speech activities. We need not concern ourselves here with the analysis of the concept, nor accept the doctrine that "meaning is use" is its entirity: what is important is the light it casts on language when we view it from the standpoint of a rule governed activity, in which communication between the participants in a dialogue is enabled by a shared knowledge of the appropriate rules, by the fact of the participants playing the same game. Our contention is that the structure observable in texts and dialogues can (and should) be accounted for in terms of the rules of the particular "game" being played in the dialogue.

A significant advantage of exploiting the rule governed nature of dialogue is that it helps to decrease the reliance on a user model. This is an advantage because otherwise much work needs to be expended on inferring the user model from the input from the user[5], and in guessing the user's intentions in the dialogue[6]. This would be a difficult task for a human to perform, let alone a computer. Indeed the conventional, rule governed nature of communication has been developed precisely to perform the kind of harmonisation between the participants in a dialogue required, without the need to perform these kinds of inference.

These ideas could be applied to the greeting game represented in the dialogue given above, but this would not be interesting as only schoolboy BASIC programmers are concerned to have their computer greet them in a friendly manner. Of rather more interest, both because of its potential for use within KBS and because the rules of the game have been copiously studied, is the notion of a dialectic game. Dialectic games were studied by philosophers who wished to make the notion of logical fallacies, such as "begging the question", more precise. They therefore drew up a set of rules which governed a properly conducted argument, and accounted for the fallacies in terms of violations of these rules. The usefulness of these rules for our purposes is that they enable a dialectic dialogue to be seen as a path through a graph representing the space of well formed dialogues. This means that the system, in its turn, has a set of legitimate moves to choose from, which can be selected according to some game playing principles; that the system has, in the user's turn, knowledge of the range of possible moves for the user, so that the strategy of the user in making one of them can be more readily divined, and that, if an illegal move is made, the breakdown in the dialogue can be detected immediately, and appropriate repair behaviour initiated, by backtracking through the graph.

Dialectic as a Model of Interaction with a KBS

Sometimes a person with a problem will go to a more informed, or experienced person, and ask him to solve the problem for him and to tell him what to do. This kind of situation is well served by the consultative model of a KBS. But there are other, probably more common, situations, when the help sought is of a different nature altogether. Here help is obtained by discussing the problem with a colleague, and the problem solving is co-operative rather than passive. Both of the participants are expected to contribute something to the problem solving, and although there may be disparity between their skills and experience, and their areas of expertise may differ, the relationship is essentially one of equals, rather than expert and layman. Such problem solving discussions may take different courses: sometimes a person will use it simply to structure his ideas, or to test a previously developed solution to see that there are no flaws in it; at others both participants will make real and essential contributions to the development of the solution. We believe that this model of interaction provides a better and more flexible solution than the consultative model which can significantly extend the areas of application of KBS. In particular, the increasingly popular "know-how" systems, which are supposed to act as repositories of wisdom rather than specific task solving systems, are likely to be more effectively accessed in this way. Further a side effect of this kind of problem solving is that the knowledge of, and understanding of the domain, of both the participants may be increased by such

discussions, suggesting that it may be possible to extend the knowledge base itself through such interactions. Another potential application is in educational systems, particularly when these are modelled on the true "leading out" that is possible in a one-on-one tutorial situation as opposed to the more rigid "spoon-feeding" necessitated by larger classes[7].

Discussions of the sort indicated above all fall within the scope of the dialectic games alluded to earlier. It is our contention that such dialectic games can form the basis of an interaction with KBS which follows this discursive model. The next two sections will detail one such dialogue game[8], and show how it can be used to determine the dialogue space within a computer system.

MacKenzie's Dialogue Game DC

Informally, MacKenzie's dialogue game, DC, involves the interaction of two protagonists - whom we shall call A(lice) and B(ob). The game proceeds in a sequence of *rounds*, each round consisting of a *locution* uttered by *A* or a locution uttered by *B*. Each protagonist has an associated *commitment store* containing a finite set of locutions. The locutions uttered by *A* and *B* may change the contents of these commitment stores in accordance with the *commitment rules* prescribed by the game.

In more formal terms, let *S* be a set of *statements* which we assume closed under negation, conjunction, and implication i.e. $\forall\, r, s \in S$ it holds that $(\neg s) \in S$, $(r\ s) \in S$ and $(r \Rightarrow s) \in S$. The set, *L*, of *locutions over S* is defined as follows:

L1) $\forall\, s \in S$ *s* is a locution (i.e. $s \in L$)

L2) $\forall\, s \in S$ the *question* "*Is it the case that s*?" is a locution.

L3) $\forall\, s \in S$ the *withdrawal* "*No commitment s*" is a locution.

L4) $\forall\, s \in S$ the *challenge* "*Why s*?" is a locution.

L5) $\forall\, s \in S$ the *resolution demand* "*Resolve whether s*" is a locution.

L6) All that are locutions arise by reason of (L1)-(L5) alone.

Note that each of (L2) through (L5) gives rise to a relation over $L \times S$: denoting these relations by *Q*, *W*, *Y*, *R* respectively it follows that *L* is completely defined by the set:

$$L = S \bigcup \bigcup_{s \in S} \{\, l : (l, s) \in Q \bigcup W \bigcup Y \bigcup R \,\}$$

With the formalism above, given a set *L* of locutions and protagonists $P = \{ A, B \}$ an *utterance* (or *locution act* in MacKenzie's phraseology) is any element of $P \times L$; an *n-round dialogue* is a sequence of *n* utterances i.e. an element of $(P \times L)^n$ and the set of all possible dialogues is just $D = \bigcup_{n=0}^{\infty} (P \times L)^n$. Note that any element of a dialogue (or *locution event*) is completely described by a triple $< i, p, l >$ where *i* is the *round number*, *p* is a protagonist and *l* is a locution.

Of course, many dialogues in *D* are uninteresting since there is no "logical" development in the progress of the dialogue. DC is concerned with a particular *dialectical system*. Such a system is a triple $<P, L, R>$ where *R* is a set of *dialogue rules* which render inadmissible certain locution events $<n, p, l>$. Let $C_{n,p} \subseteq L$ denote the set of commitments of protagonist *p* after the *n*'th round of some dialogue *d*. The significance of the rule set *R* is that it bars locutions in round $n+1$ solely through use of *syntactical relations* involving *L*, $C_{n,A}$, $C_{n,B}$ and the immediately preceding locution event $<n, p, l>$.

The remaining two features of DC that have to be described are the rules by which the commitment stores, $C_{n,A}$ and $C_{n,B}$, are affected by the locution event in round *n*; and the dialogue rules for prohibiting certain locution events. We will not describe these completely (the reader interested in a full definition is referred to MacKenzie's paper). The important property of dialogue rules has already been described and we may summarise the commitment rules as follows: $C_{0,A} = C_{0,B} = \varnothing$, that is before the game commences neither protagonist is committed to any locution. For $n > 0$ for each type of locution, *l*, i.e. whether *l* is a locution by reason of one of (L1) through (L5), there are commitment rules which describe the members of $C_{n+1,A}$ and $C_{n+1,B}$ in terms of $C_{n,A}$ (respectively $C_{n,B}$) and the locution *l*. For example, the commitment rule associated with the locution event $<n, A, \textit{Why } s?>$ is $C_{n,A} := C_{n-1,A} \bigcup \{\, \textit{Why } s \,\} - \{ s \}$; $C_{n,B} := C_{n-1,B} \bigcup \{ s \}$. The simplest example of a dialogue rule in DC is the rule R_{Form} which states that no

legal dialogue contains two consecutive locution utterances $< P, l >$, $< P, l' >$, where $P \in \{ A, B \}$; or an event $< n, p, l >$ where $l \notin L$. This forces the protagonists to take turns in making utterances and ensures that only locutions are admitted. A more complex rule is R_{Chall} concerning challenges. This states that no legal dialogue of length $n+1$ contains an event $< n-1, P,$ *Why s?* $>$ unless it also contains an event $< n, Q, l >$ such that

i) $< l, s > \in W$ or

ii) $(l \in S)$ & (*Why l?* $\notin C_{n,P}$) or

iii) $< l,$ (*the conjunction of* $t \in T \Rightarrow s$) $> \in R$ where $T \subseteq C_{n,P}$ and s is an immediate consequence of the statements in T.

In less formal terms the challenge rule states that a challenge event from protagonist P must be followed by a locution event from protagonist Q that consists of: withdrawing any commitment to the statement under challenge; or a statement not under challenge by P; or a request to resolve whether the statement under challenge follows as an immediate consequence of statements to which P is presently committed.

Using DC in a Computer System

The application of dialogue formalisations to computational activities has earlier been considered in the field of structural complexity theory[9]. The model examined in that domain is of some interest to KBS work: two agents interacting in order to construct a proof of a given assertion. A recent result[10] establishes that the class of assertions that can be proved within a "small" number of interactions is identical to the class of decision problems solvable by programs with "small" memory size requirements. The latter class includes problems such a determining whether a given quantified formula in logic is a tautology.

The schema with which this paper is concerned is depicted in Figure 1, below.

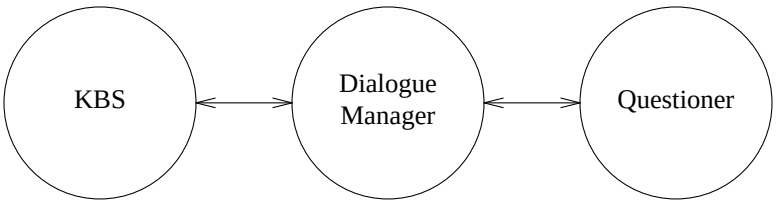

Figure 1

In terms of the formal model introduced in the previous section, the protagonists are the KBS questioner (Q) and the reasoning engine of the KBS (R). The underlying statement set, S, consists of the data (facts, inference rules, relations etc.) encoded in the knowledge base. As in DC the commitment stores of Q and R are initially empty.

The dialogue manager (DM) is responsible for a three main tasks, summarised below.

i) It acts as an interface between the KBS and the Questioner, translating utterances from the latter into requests to the former; and translating responses from the former into locutions to the latter.

ii) It must maintain the commitment stores for Q and R.

iii) It must enforce the rules of the dialectical system underlying the dialogue game. Thus if the questioner attempts to make an illegal move this must be rejected and if necessary a return to an previous state in the dialogue space must be effected. A similar process must be carried out if the response from R is illegal.

Note that there is no asymmetry involved in the DM's treatment of Q and R. This is appropriate for the scenarios described earlier in the paper since, in these, both agents cooperate in a problem solving task rather being in a "master-slave" relationship. The DM supplies both Q and R with a *template* of valid locution forms cf. the description of the dialogue abstract machine below.

Task (i) is, in essence, a compilation process that involves mapping between two different formalisms. The second is a straightforward set maintenance activity. To realise task (iii) we employ a *graph*

modification system specific to dialogues. Such systems have been considered earlier with respect to document modelling methods by several authors[11,12,13].

Graph Modification Systems for Dialogue Management

A *dialogue graph* is a labelled directed acyclic graph $G(V, E)$ satisfying the following constraints: each $v \in V$ has an associated label $\lambda(v) \in \mathbf{N} \times P \times L \times \{0, 1\}$; if $< v, w > \in E$ and $\lambda(v) = < n, p, l, 1 >$ then $\lambda(w) \in < n+1, q, l' > \times \{0,1\}$ where $<n,p,l>$; $< n+1, q, l' >$ must be a legal fragment of a dialogue in *D*. We call nodes whose final label entry is 1 *expanded* nodes and those with label entry 0 *unexpanded* nodes. Every unexpanded node has no outgoing edges. Finally, there is at most one path, *d*, in the graph such that *d* commences in the source node and ends in an expanded node all of whose successors are unexpanded nodes. We call this path the *dialogue path* of *G*.

A dialogue graph completely describes a point in the dialogue space: the sequence of labels on the dialogue path describe a legal dialogue; unexpanded nodes represent possible continuations of a dialogue which have not been pursued; expanded nodes with no successors represent former dialogue paths which have ended with no legal continuation; finally a graph which does not contain a dialogue path encodes the situation where the dialogue being pursued cannot be continued.

Let $DG(L)$ denote the set of all dialogue graphs with underlying locution set *L*. Then for any (computable*) dialectical system *DS* (of which *DC* is one example) one may define an associated graph grammar $\Gamma(DS)$ such that

i) $DG(L)$ is closed under the application of production rules in $\Gamma(DS)$.

ii) $\Gamma(DS)$ contains production rules to turn a dialogue graph with no dialogue path into a dialogue graph representing some earlier stage of the dialogue in progress.

iii) $\Gamma(DS)$ contains production rules to advance the state of a dialogue by one round.

Any such graph grammar constitues a graph modification system for a particular dialogue game.

A Dialogue Abstract Machine (DAM).

In an earlier section we introduced the notion of a dialogue manager responsible for supervising the progress of a dialogue between a (presumed) human participant and a KBS. Since the task of the dialogue manager is not to take part in the dialogue, but rather to mediate between the two active participants and to enforce the rules of the dialogue game, it is appropriate to regard this task as a sequence of steps in the execution of an abstract machine, which we will define as a Dialogue Abstract Machine (DAM). Like the analogous Hypertext Abstract Machine (HAM) of Campbell and Goodman[14], DAM offers a general storage and execution model which may be implemented in many ways and may interact with KB systems of many formalisms.

The principal components of DAM are the following:

1. A *machine state*, which defines: the dialogue sequence number *n*; the currently active participant (Alice or Bob); and the set of currently allowable locutions as defined by the dialogue rules of DC. Dialogue steps produce a sequence of state transitions in DAM.
2. A dialogue graph, representing the history of the dialogue in progress. A node in the dialogue graph takes the form of a locution made by one of the participants, accompanied by a *Reply Template* defining the set of possible responses which are allowable within the rules of DC. The reply template may be seen as a derivative of the state of DAM at the time the locution was made; it provides a set of unexpanded nodes for each locution, defining possible continuations of the dialogue.
3. The *Commitment Store*, elements of which are locutions (statements and challenges) to which one or both participants are committed.

---

*) 'computable' in the sense that the relation $VALID \in (\mathbf{N} \times P \times L)^2$ - where $< x, y > \in VALID$ if and only if *y* is a legal continuation of *x* - is computable.

4. A set of *Dialogue Step Operations*, corresponding to the five locution types L1-L5 defined in the preceding section.
5. A set of *Communication Primitive Functions*, which provide interfaces between DAM and the dialogue paticipants (human and KBS).

Each locution offered by a participant invokes a dialogue step operation in DAM, which proceeds as follows:

i) The validity of the locution is verified, according to the dialogue rules of DC;

ii) The machine state is advanced, including incrementing the locution sequence counter, switching the participant status, and defining the dialogue step status;

iii) The relevant commitment rule is applied, leading to amendments to the state of the commitment store;

iv) A reply template is constructed by applying the dialogue rules and syntactical relations to the current locution and commitment store;

v) The dialogue graph is extended by appending the current locution and reply template.

The purpose of the rule-set of DC (and hence of the operational semantics of DAM) is to enable dialogues from which both participants can draw conclusions from chains of implications linking statements to which both are committed (and, incidentally, to prohibit circularity in these chains, with the corresponding inference of 'begging the question'). Consider, for example, the response to the challenge: *Why s?* The general reply template for this would be the triplet of continuations: (*withdraw s*; *t*; *resolve X* ☞ *s*). However, in not every case is each reply allowable. In particular, the resolution demand is an allowable response only if *X* defines a set of commitments accepted by the challenger, of which *s* is an immediate consequence. In constructing the specific reply template at this step, DAM must inspect the commitment store to establish whether any such set of commitments exists, and if not, no such unexpanded node will be added to the dialogue graph.

The reply *t* to the challenge *Why s*? carries the implication: $t => s$. If this implication exists in the commitment store, then DAM will produce it as a specific response in the reply template. If not, then the reply template will contain an unexpanded node, indicating that the responding participant may extend the dialogue along this arc by introducing a new statement, i.e. one not previously included in his/her commitments. If the participant at this step is the KBS, the dialogue manager may at this stage interrogate the KBS to explore whether any such implication can be identified in the knowledge base. It is for this purpose that the communication primitive functions are provided. A communication primitive function presents to a participant (in this case, the KBS) a locution and accompanying response template, with the invitation to fill in an appropriate response.

Dialogues are concluded, when successful, when both participants are committed to a chain of implications linking agreed premises to required conclusions. To facilitate this, the implementation of the commitment store uses as atomic elements the set of 'basic' statements, from which the forms of negation, conjunction and implication are represented as linked structures. This form of implementation enables efficient identification of implications and immediate consequences, as is required for an effective dialogue.

Unsuccessful dialogues may terminate when no valid continuation is possible, or no information is available to continue the dialogue along the path taken. In this case, DAM enables a backtracking procedure to be effected, returning the dialogue to a preceding node in the dialogue graph at which some unexpanded node successor is available. The complete history of the dialogue, including abortive paths, is retained in the graph structure.

Conclusion

In this paper we have proposed a novel style of interaction with a KBS, which we believe would be more effective than the traditional consultative style, for a variety of applications, where the user has a degree of expertise in the area covered by the system. We have been at pains to make the model as general as possible, so that the style could be applied to a range of KBS, constructed using different formalisms. At the heart of our approach is the conviction that we should take the rule governed nature of much of human discourse seriously and incorporate an appropriate set of rules in the interface. In this way genuine

conversation is enabled, rather than the mimicry that results from the incorporation of only surface features of dialogue. For purposes of illustration, we have used the set of rules presented in MacKenzie's Dialogue Game DC, since these are widely known, well worked out and can be presented succinctly. It may well be, however, that these rules, deriving as they do from a very formal system, would result in a somewhat stilted dialogue. A rather more promising approach to produce natural discussions would be to produce a set of rules based on a less formal approach to dialectic, such as than of Stephen Toulmin[15]. The substitution of these rules for the rules of DC would not significantly change the approach, and could readily be mapped into the DAM outlined above.

References

[1] McKeown, K.R., *Text Generation: Using discourse strategies and focus constraints to generate natural language text.* Cambridge University Press, 1985.

[2] Mann, W.C., and Thompson, S.A., *Rhetorical Structure Theory: A Framework for the Analysis of Texts*, Technical Report RS-87-190, USC/Information Sciences Institute, 1987.

[3] e.g. Moore, Johanna D., *A Reactive Approach to Explanation in Expert and Advice Giving Systems*, Ph. D. Thesis, Information Sciences Institute, University of Southern California, 1989.

[4] Wittgenstein, L., *Philosophical Investigations*, Blackwell, Oxford, 1953.

[5] Shadbolt, N.R., *Constituting Reference in Natural language: the Problem of Referential Opacity* Ph. D. Thesis, University of Edinburgh, 1984.

[6] Carberry, S., and Cebulka, K., *Capturing Rational Behaviour in Natural Language Information Systems*, Cohn, A., (ed) Proceedings of AISB 7, Pitman, London, 1989, pp153-164.

[7] Moore, D., *Dialogue Game Theory for Explanation Giving Systems*, Proceedings of the 5th Explanation Workshop, University of Manchester, 1990.

[8] Mackenzie, J.D., *Question-Begging in Non-Cumulative Systems*, Journal of Philosophical Logic, vol 8, 1979, pp 159-177.

[9] Hopcroft, J., and Ullman, J., *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley, 1979

[10] Shamir, A., *IP=PSPACE*, (to appear)

[11] Bench-Capon, T.J.M., and Dunne, P.E.S., *An approach to the integration of legal support systems*, Database and Expert Systems Applications (Ed: A. Min Tjoa and R. Wagner), Springer, 1990, pp 105-112

[12] Koo, R., *A model for electronic documents*, ACM SIGOIS Bulletin, Vol 10 (1), 1989, pp 23-33

[13] Bench-Capon, T.J.M., and Dunne, P.E.S., *Some computational properties of a model for electronic documents*, Electronic Publishing, Vol. 2 (4), 1989, pp 231-256

[14] Campbell, B., and Goodman, J.M., *HAM: a general purpose Hypertext Abstract Machine*, Comm. ACM., Vol. 31 (7), 1988, pp. 856-861

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