Modelling Grounding and Discourse Obligations Using Update Rules

Colin Matheson
University of Edinburgh
Edinburgh, Scotland
colin.matheson@ed.ac.uk

Massimo Poesio
University of Edinburgh
Edinburgh, Scotland
massimo.poesio@ed.ac.uk

David Traum
University of Maryland
Maryland, USA
traum@cs.umd.edu

Abstract

This paper describes an implementation of some key aspects of a theory of dialogue processing whose main concerns are to provide models of GROUNDING and of the role of DISCOURSE OBLIGATIONS in an agent's deliberation processes. Our system uses the TrindiKit dialogue move engine toolkit, which assumes a model of dialogue in which a participant's knowledge is characterised in terms of INFORMATION STATES which are subject to various kinds of updating mechanisms.

1 Introduction

In this paper we describe a preliminary implementation of a 'middle-level' dialogue management system. The key tasks of a dialogue manager are to *update* the representation of dialogue on the basis of processed input (generally, but not exclusively, language utterances), and to *decide* what (if anything) the system should do next. There is a wide range of opinions concerning how these tasks should be performed, and in particular, how the ongoing dialogue state should be represented: e.g., as something very specific to a particular domain, or according to some more general theory of (human or human inspired) dialogue processing. At one extreme, some systems represent only the (typically very rigid) transitions possible in a perceived dialogue for the given task, often using finite states in a transition network to represent the dialogue: examples of this are systems built using Nuance's DialogueBuilder or the CSLU's Rapid Application Prototyper. The other extreme is to build the dialogue processing theory on top of a full model of rational agency (e.g., (Bretier and Sadek, 1996)). The approach we take here lies in between these two extremes: we use rich representations of information states, but simpler, more dialogue-specific deliberation methods, rather than a deductive reasoner working on the basis of an axiomatic theory of rational agency. We show in this paper that the theory of information states we propose can, nevertheless, be used to give a characterisation of dialogue acts such as those proposed by the Discourse Resource Initiative precise enough to formalise the deliberation process of a dialogue manager in a completely declarative fashion.

Our implementation is based on the approach to dialogue developed in (Traum, 1994; Poesio and Traum, 1997; Poesio and Traum, 1998; Traum et al., 1999). This theory, like other action-based theories of dialogue, views dialogue participation in terms of agents performing dialogue acts, the effects of which are to update the information state of the participants in a dialogue. However, our view of dialogue act effects is closer in some respects to that of (Allwood, 1976; Allwood, 1994) and (Singh, 1998) than to the belief and intention model of (Sadek, 1991; Grosz and Sidner, 1990; Cohen and Levesque, 1990). Particular emphasis is placed on the *social* commitments of the dialogue participants (obligations to act and commitments to propositions) without making explicit claims about the actual beliefs and intentions of the participants. Also, heavy emphasis is placed on how dialogue participants socially GROUND (Clark and Wilkes-Gibbs, 1986) the information expressed in dialogue: the information state assumed in this theory specifies which information is assumed to be already part of the common ground at a given point, and which part has been introduced, but not yet been established.

The rest of this paper is structured as follows. The theory of dialogue underlying the implementation is described in more detail in Section 2. Section 3 describes the implementation itself. Section 4 shows how the system updates its information state while participating in a fairly simple dialogue.

2 Theoretical Background

One basic assumption underlying this work is that it is useful to analyse dialogues by describing the relevant 'information' that is available to each participant. The notion of INFORMATION STATE (IS) is therefore employed in deciding what the next action should be, and the effects of utterances are described in terms of the changes they bring about in ISs. A particular instantiation of a dialogue manager, from this point of view, consists of a definition of the contents of ISs plus a description of the update processes which map from IS to IS. Updates are typically triggered by 'full' dialogue acts such as assertions or directives,[1] of course, but the theory allows parts of utterances, including individual words and even subparts of words, to be the trigger. The update rules for dialogue acts that we assume here are a simplified version of the formalisations proposed in (Poesio and Traum, 1998; Traum et al., 1999) (henceforth, PTT).

The main aspects of PTT which have been implemented concern the way discourse obligations are handled and the manner in which dialogue participants interact to add information to the common ground. Obligations are essentially social in nature, and directly characterise spoken dialogue; a typical example of a discourse obligation concerns the relationship between questions and answers. Poesio and Traum follow (Traum and Allen, 1994) in suggesting that the utterance of a question imposes an obligation on the hearer to address the question (e.g., by providing an answer), irrespective of intentions.

As for the process by which common ground is established, or GROUNDING (Clark and Schaefer, 1989; Traum, 1994), the assumption in PTT is that classical speech act theory is inherently too simplistic in that it ignores the fact that co-operative interaction is essential in discourse; thus, for instance, simply asserting something does not make it become mutually 'known' (part of the common ground). It is actually necessary for the hearer to provide some kind of acknowledgement that the assertion has been received, understood or not understood, accepted or rejected, and so on. Poesio and Traum view the public information state as including both material that has already been grounded, indicated by GND here, and material that hasn't been grounded yet. These components of the information state are updated when GROUNDING ACTS such as acknowledgement are performed. Each new contribution results in a new DISCOURSE UNIT (DU) being added to the information state (Traum, 1994) and recorded in a list of 'ungrounded discourse units' (UDUS); these DUs can then be subsequently grounded as the result, e.g., of (implicit or explicit) acknowledgements.

3 Implementing PTT

In this section, we describe the details of the implementation. First, in Section 3.1, we describe the TrindiKit tool for building dialogue managers that we used to build our system. In Section 3.2, we describe the information states used in the implementation, an extension and simplification of the ideas from PTT discussed in the previous section. Then, in Section 3.3, we discuss how the information state is updated when dialogue acts are observed. Finally, in Section 3.4, we describe the rules used by the system to adopt intentions and perform its own actions. An extended example of how these mechanisms are used to track and participate in a dialogue is presented in Section 4.

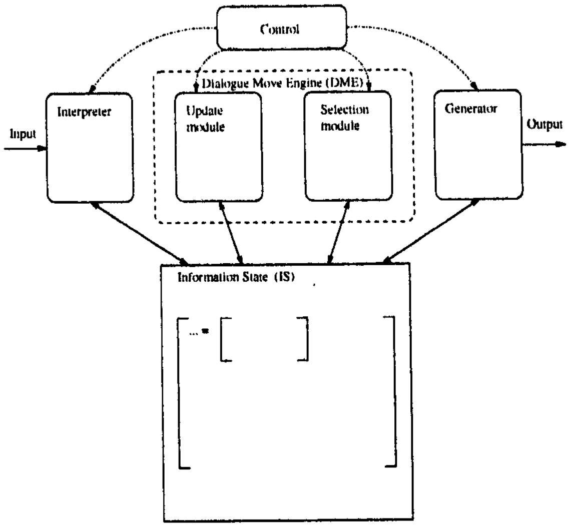

Figure 1: TrindiKit Architecture

3.1 TrindiKit

The basis for our implementation is the TrindiKit dialogue move engine toolkit implemented as part of the TRINDI project (Larsson et al., 1999). The toolkit provides support for developing dialogue systems, focusing on the central dialogue management components.

The system architecture assumed by the TrindiKit is shown in Figure 1. A prominent feature of this architecture is the information state, which serves as a central 'blackboard' that processing modules can examine (by means of defined CONDITIONS) or change (by means of defined OPERATIONS). The structure of the IS for a particular dialogue system is defined by the developer who uses the TrindiKit to build that system, on the basis of his/her own theory of dialogue processing; no predefined notion of information state is provided.[2]. The toolkit provides a number of abstract data-types such as lists, stacks, and records, along with associated conditions and operations, that can be used to implement the user's theory of information states; other abstract types can also be defined. In addition to this customisable notion of information state, TrindiKit provides a few system variables that can also used for inter-module communication. These include `input` for the raw observed (language) input, `latest_moves` which

[1] We assume here the DRI classification of dialogue acts (Discourse Resource Initiative, 1997).

[2] In TRINDI we are experimenting with multiple instantiations of three different theories of information state (Traum et al., 1999).

contains the dialogue moves observed in the most recent turn, latest_speaker, and next_moves, containing the dialogue moves to be performed by the system in the next turn.

A complete system is assumed to consist of several modules interacting via the IS. (See Figure 1 again.) The central component is called the DIALOGUE MOVE ENGINE (DME). The DME performs the processing needed to integrate the observed dialogue moves with the IS, and to select new moves for the system to perform. These two functions are encapsulated in the UPDATE and SELECTION submodules of the DME. The update and select modules are specified by means of typed rules, as well as sequencing procedures to determine when to apply the rules. We are here mainly concerned with UPDATE RULES (urules), which consist of four parts: a name, a type, a list of conditions to check in the information state, and a list of operations to perform on the information state. urules are described in more detail below, in Section 3.3. There are also two modules outside the DME proper, but still crucial to a complete system: INTERPRETATION, which consumes the input and produces a list of dialogue acts in the latest_moves variable (potentially making reference to the current information state), and GENERATION, which produces NL output from the dialogue acts in the next_moves variable. Finally, there is a CONTROL module, that governs the sequencing (or parallel invocation) of the other modules. In this paper we focus on the IS and the DME; our current implementation only uses very simple interpretation and generation components.

3.2 Information States in PTT

In this section we discuss the information state used in the current implementation. The main difference between the implemented IS and the theoretical proposal in (Poesio and Traum, 1998) is that in the implementation the information state is partitioned in fields, each containing information of different types, whereas in the theoretical version the information state is a single repository of facts (a DISCOURSE REPRESENTATION STRUCTURE). Other differences are discussed below. An example IS with some fields filled is shown in Figure 2; this is the IS which results from the second utterance in the example dialogue discussed in Section 4, *A route please.*[3]

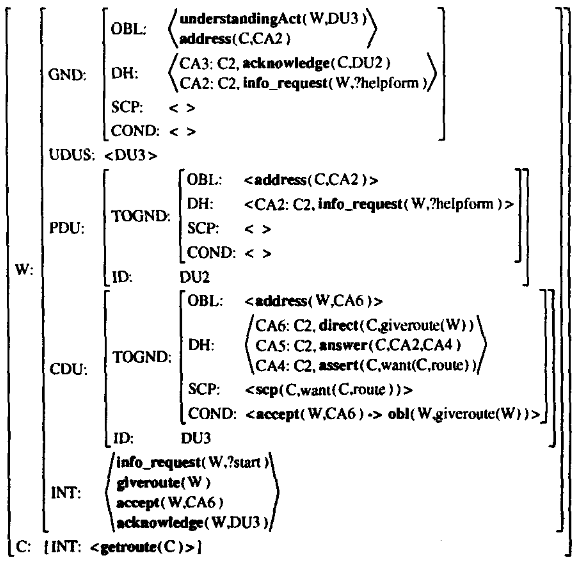
W:
GND:
OBL: ⟨understandingAct(W,DU3) address(C,CA2)⟩
DH: ⟨CA3: C2, acknowledge(C,DU2) CA2: C2, info_request(W,?helpform)⟩
SCP: < >
COND: < >
UDUS: <DU3>
PDU:
TOGND:
OBL: <address(C,CA2)>
DH: <CA2: C2, info_request(W,?helpform)>
SCP: < >
COND: < >
ID: DU2
CDU:
TOGND:
OBL: <address(W,CA6)>
DH: ⟨CA6: C2, direct(C,giveroute(W)) CA5: C2, answer(C,CA2,CA4) CA4: C2, assert(C,want(C,route))⟩
SCP: <scp(C,want(C,route))>
COND: <accept(W,CA6) -> obl(W,giveroute(W))>
ID: DU3
INT: ⟨info_request(W,?start) giveroute(W) accept(W,CA6) acknowledge(W,DU3)⟩
C: [INT: <getroute(C)>]

Figure 2: Structure of Information States

The IS in Figure 2 is a record with two main parts, W and C. The first of these represents the system's (Wizard) view of his own mental state and of the (semi-)public information discussed in the dialogue; the second, his view of the user's (Caller) information state. This second part is needed to model misunderstandings arising from the dialogue participants having differing views on what has been grounded; as we are not concerned with this problem here, we will ignore C in what follows.

W contains information on the grounded material (GND), on the ungrounded information (UDUS, PDU and CDU), and on W's intentions (INT). GND contains the information that has already been grounded; the other fields contain information about the contributions still to be grounded. As noticed above, in PTT it is assumed that for each new utterance, a new DU is created and added to the IS. The current implementation differs from the full theory in that only two DUs are retained at each point; the current DU (CDU) and the previous DU (PDU). The CDU contains the information in the latest contribution, while the PDU contains information from the penultimate contribution. Information is moved from PDU to GND as a result of an **ack** (acknowledgement) dialogue act (see below.)

The DUs and the GND field contain four fields, representing obligations (OBL), the dialogue history (DH), propositions to which agents are socially committed (SCP) , and conditional updates (COND). The value of OBL is a list of action types: actions that agents are obliged to perform. An action type is specified by a PREDICATE, a DIALOGUE PARTICIPANT, and a list of ARGUMENTS. The value of SCP is a list of a particular type of mental states, social commitments of agents to propositions.[4] These are specified by a DIALOGUE PARTICIPANT, and a PROPOSITION. Finally, the elements in DH are dia-

[3] All diagrams in this paper are automatically generated from TrindiKit system internal representations and displayed using the Thistle dialogue editor (Calder, 1998). Some have been subsequently edited for brevity and clarity.

[4] SCPs play much the same role in PTT as do beliefs in many BDI accounts of speech acts.

logue actions, which are instances of dialogue action types. A dialogue action is specified by an action type, a dialogue act id, and a confidence level CONF (the confidence that an agent has that that dialogue act has been observed).

The situation in Figure 2 is the result of updates to the IS caused by utterance [2] in the dialogue in (6), which is assumed to generate a **direct** act as well as an **assert** act and an **answer** act.[5] That utterance is also assumed to contain an implicit acknowledgement of the original question; this understanding act has resulted in the contents of DU2 being grounded (and subsequently merged with GND), as discussed below.

GND.OBL in Figure 2 includes two obligations. The first is an obligation on W to perform an understanding act (the predicate is **understandingAct**, the participant is W, and there is just one argument, DU3, which identifies the DU in CDU by referring to its ID). The second obligation is an obligation on C to **address** conversational act CA2; this ID points to the appropriate **info_request** in the DH list by means of the ID number. Obligations are specified in CDU and PDU, as well. Those in PDU are simply a subset of those in GND, since at point in the update process shown in Figure 2 this field contains information that has already been grounded (note that DU2 is not in UDUS anymore); but CDU contains obligations that have not been grounded yet – in particular, the obligation on W to **address** CA6.

GND.DH in this IS contains a list of dialogue actions whose occurrence has already been grounded: the **info_request** performed by utterance 1, with argument a question,[6] and the implicit **acknowledge** performed by utterance 2.[7] The DH field in CDU contains dialogue acts performed by utterance 2 that *do* need to be grounded: a directive by C to W to perform an action of type **giveroute**, and an **assert** by C of the proposition **want**(C, route), by which C provides an **answer** to the previous **info_request** CA2.

The COND field in CDU contains a conditional update resulting from the directive performed by that utterance. The idea is that directives do not immediately lead to obligations to perform the mentioned action: instead (in addition to an obligation to address the action with some sort of response), their effect is to add to the common ground the information that *if* the directive is **accepted** by the addressee, *then* he or she has the obligation to perform the action type requested. (In this case, to give a route to C.)

3.3 Update Rules in PTT

We are now in a position to examine the update mechanisms which are performed when new dialogue acts are recognised. When a dialogue participant takes a turn and produces an utterance, the interpretation module sets the system variable `latest_moves` to contain a representation of the dialogue acts performed with the utterance. The updating procedure then uses update rules to modify the IS on the basis of the contents of `latest_moves` and of the previous IS. The basic procedure is described in (1) below.[8]

(1)

1. Create a new DU and push it on top of UDUs.
2. Perform updates on the basis of backwards grounding acts.
3. If any other type of act is observed, record it in the dialogue history in CDU and apply the update rules for this kind of act
4. Apply update rules to all parts of the IS which contain newly added acts.

The first step involves moving the contents of CDU to PDU (losing direct access to the former PDU contents) and putting in CDU a new empty DU with a new identifier. The second and third steps deal explicitly with the contents of `latest_moves`, applying one urule (of possibly a larger set) for each act in `latest_moves`. The relevant effects for each act are summarised in (2), where the variables have the following types:

IDx	Dialogue Act Identification Number
DUx	DU Identification Number
DP	Dialogue Participant (i.e., the speaker)
Q	A Question
PROP	A Proposition
Act	An Action
o(DP)	The other dialogue participant
P(ID)	The content of the ID, a proposition
Q(ID)	The content of the ID, a question

[5] The fact that the utterance of *a route please* constitutes an answer is explicitly assumed; however, it should be possible to derive this information automatically (perhaps along the lines suggested by Kreutel (Kreutel, 1998)).

[6] We use the notation ?p to indicate a question of the form ?([x],p(x)) .

[7] We assume here, as in (Traum, 1994) and (Poesio and Traum, 1998), that understanding acts do not have to be grounded themselves, which would result in a infinite regress.

[8] See (Poesio et al., 1999; Traum et al., 1999) for different versions of this update procedure used for slightly different versions of the theory.

(2)

act	ID:2, **accept**(DP,ID2)
effect	*accomplished via rule resolution*
act	ID:2, **ack**(DP,DU1)
effect	peRec(w.Gnd,w.pdu.tognd)
effect	remove(DU1,UDUS)
act	ID:2, **agree**(DP,ID2)
effect	push(SCP,**scp**(DP,**P**(ID2)))
act	ID:2, **answer**(DP,ID2,ID3)
effect	push(SCP,**ans**(DP,**Q**(ID2),**P**(ID2)))
act	ID:2, **assert**(DP,PROP)
effect	push(SCP,**scp**(DP,PROP))
effect	push(COND,**accept**(**o**(DP),ID)→ **scp**(**o**(DP),PROP))
act	ID:1, **assert**(DP,PROP)
effect	push(COND,**accept**(**o**(DP),ID)→ **scp**(**o**(DP),PROP))
act	ID:2, **check**(DP,PROP)
effect	push(OBL,**address**(**o**(DP),ID))
effect	push(COND,**agree**(**o**(DP),ID) → **scp**(DP,PROP))
act	ID:2, **direct**(DP,Act)
effect	push(OBL,**address**(**o**(DP),ID))
effect	push(COND,**accept**(**o**(DP),ID) → **obl**(**o**(DP),Act))
act	ID:2, **info_request**(DP,Q)
effect	push(OBL,**address**(**o**(DP),ID))

The **ack** act is the only backward grounding act implemented at the moment. The main effect of an **ack** is to merge the information in the acknowledged DU (assumed to be PDU) into GND, also removing this DU from UDUS. Unlike the other acts described below, **ack** acts are recorded directly into GND.DH, rather than into CDU.TOGND.DH.

All of the other updates are performed in the third step of the procedure in (1). The only effect of **accept** acts is to enable the conditional rules which are part of the effect of **assert** and **direct**, leading to social commitments and obligations, respectively. **agree** acts also trigger conditional rules introduced by **check**; in addition, they result in the agent being socially committed to the proposition introduced by the act with which the agent agrees. Performing an **answer** to question ID2 by asserting proposition P(ID3) commits the dialogue participant to the proposition that P(ID3) is indeed an answer to Q(ID2).

The two rules for **assert** are where the confidence levels are actually used, to implement a simple verification strategy. The idea is that the system only assumes that the user is committed to the asserted proposition when a confidence level of 2 is observed, while some asserts are assumed not to have been sufficiently well understood, and are only assigned a confidence level 1. This leads the system to perform a check, as we will see shortly.

The next three update rules, for **check**, **direct**, and **info_req**, all impose an obligation on the other dialogue participant to address the dialogue act. In addition, the **direct** rule introduces a conditional act: acceptance of the directive will impose an obligation on the hearer to act on its contents.

In addition, all FORWARD ACTS[9] in the DRI scheme (Discourse Resource Initiative, 1997) impose an obligation to perform an understanding act (e.g., an acknowledgement):

(3)

act	ID:c, **forward-looking-act**(DP)
effect	push(OBL,**u-act**(**o**(DP),CDU.id))

The internal urules implementing the updates in (2) have the format shown in (4), which is the urule for **info_request**.

(4)
```
urule( doInfoReq, ruletype3,
   [ hearer(DP),
     latest_moves: in(Move),
     Move:valRec(pred,inforeq) ],
   [ incr_set(update_cycles,_),
     incr_set(next_dh_id,HID),
     next_du_name(ID),
     pushRec(w^cdu^tognd^dh,
           record([atype=Move,clevel=2,id=HID ])),
     pushRec(w^cdu^tognd^obl,
           record([pred=address,dp=DP,
                   args=stackset(
                     [record([item=HID])])])),
     pushRec(w^gnd^obl,
           record([pred=uact,dp=P,
                   args=stackset(
                     [record([item=ID])])])) ]).
```

As noted above, these rules have four parts; a name, a type, a list of conditions, and a list of effects. The conditions in (4) state that there must be a move in latest_moves whose predicate is inforeq. The effects[10] state that the move should be recorded in the dialogue history in CDU, that an obligation to address the request should be pushed into OBL in CDU, and that the requirement for an understanding act by W should be pushed directly into the list in W.GND.

The fourth and final step of the algorithm cycles through the updating process in case recently added facts have further implications. For instance, when an action has been performed that matches the antecedent of a rule in COND, the consequent is established. Likewise, when an action is performed it releases any obligations to perform that action. Thus, **accept**, **answer**, and **agree** are all ways of releasing an obligation to **address**, since these are all appropriate backward looking actions. Similarly, an agent will drop intentions to perform actions it has already (successfully) performed.

3.4 Deliberation

We assume, in common with BDI-approaches to agency (e.g., (Bratman et al., 1988)) that intentions

[9] Forward acts include **assert**, **check**, **direct**, and **info_request**.

[10] The ID and HID values simply contain numbers identifying the discourse units and conversational acts.

are the primary mental attitude leading to an agent's actions. The main issues to explain then become how such intentions are adopted given the rest of the information state, and how an agent gets from intentions to actual performance.

For the latter question, we take a fairly simplistic approach here: all the intentions to perform dialogue acts are simply transferred to the next_moves system variable, with the assumption that the generation module can realise all of them as a single utterance. A more sophisticated approach would be to weight the importance of (immediate) realisation of sets of intentions and compare this to the likelihood that particular utterances will achieve these effects at minimal cost, and choose accordingly. We leave this for future work (see (Traum and Dillenbourg, 1998) for some preliminary ideas along these lines), concentrating here on the first issue – how the system adopts intentions to perform dialogue acts from other aspects of the mental state.

The current system takes the following factors into account:

- obligations (to perform understanding acts, to address previous dialogue acts, to perform other actions)
- potential obligations (that would result if another act were performed, as represented in the COND field)
- insufficiently understood dialogue acts (with a 1 confidence level in CDU.DH)
- intentions to perform complex acts

The current deliberation process assumes maximal cooperativity, in that the system always chooses to meet its obligations whenever possible, and also chooses to provide a maximally helpful response when possible. Thus, when obliged to **address** a previous dialogue act such as a question or directive, it will choose to actually return the answer or perform the action, if possible, rather than reject or negotiate such a performance, which would also be acting in accordance with the obligations (see (Kreutel, 1998) on how acts might be rejected).

In the current implementation, the following rules are used to adopt new intentions (i.e., to update the INT field):

(5)
1. add an intention to **acknowledge**(W,CDU), given an obligation to perform a **u-act**, if everything in CDU is sufficiently understood (i.e., to level 2);
2. add an intention to accept a directive or answer a question as the result of an obligation to address a dialogue act;
3. add an intention to perform an action if COND contains a conditional that will establish an obligation to perform the action, and the antecedent of this conditional is another action that is already intended. (This *anticipatory* planning allows the obligation to be discharged at the same time it is invoked, e.g., without giving an intermediate acceptance of an directive.)
4. add an intention to perform a (dialogue) action motivated by the intention to perform the current task. In the case of the Autoroute domain, we have two cases: the system may decide
   (a) to **check** any dialogue acts in CDU at confidence level 1, which contain information needed to discharge the intention to give a route; or
   (b) to perform a question asking about a new piece of information that has not been established (this is decided by inspecting GND.SCP and CDU.SCP). For example, it may decide to ask about the starting point, the time of departure, etc.

4 Extended Example

In this section, we discuss more examples of how the information state changes as a result of processing and performing dialogue acts. It is useful to do this by looking briefly at a typical Autoroute dialogue, shown in (6).[11] Our implementation can process this sort of dialogue using very simple interpretation and generation routines that provide the dialogue acts in latest_moves from the text strings, and produce W's output text from the dialogue acts which the system places in next_moves.

(6)
W [1]: How can I help?
C [2]: A route please
W [3]: Where would you like to start?
C [4]: Malvern
W [5]: Great Malvern?
C [6]: Yes
W [7]: Where do you want to go?
C [8]: Edwinstowe
W [9]: Edwinstowe in Nottingham?
C [10]: Yes
W [11]: When do you want to leave?
C [12]: Six pm
W [13]: Leaving at 6 p.m.?
C [14]: Yes
W [15]: Do you want the quickest or the shortest route?
C [16]: Quickest
W [17]: Please wait while your route is calculated.

We assume that before the dialogue starts, W has the intention to ask C what kind of help is required,

[11] The interchanges have been cleaned up to some extent here, mainly by removing pauses and hesitations.

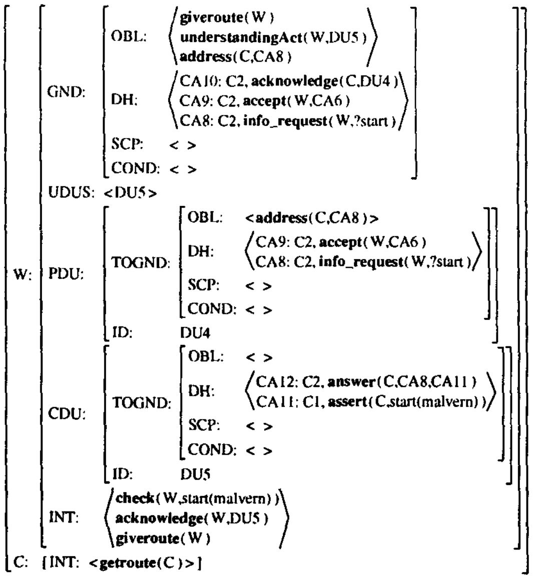
W:
GND:
OBL: giveroute(W) understandingAct(W,DU5) address(C,CA8)
DH: CA10: C2, acknowledge(C,DU4) CA9: C2, accept(W,CA6) CA8: C2, info_request(W,?start)
SCP: < >
COND: < >
UDUS: <DU5>
PDU:
TOGND:
OBL: <address(C,CA8)>
DH: CA9: C2, accept(W,CA6) CA8: C2, info_request(W,?start)
SCP: < >
COND: < >
ID: DU4
CDU:
TOGND:
OBL: < >
DH: CA12: C2, answer(C,CA8,CA11) CA11: C1, assert(C,start(malvern))
SCP: < >
COND: < >
ID: DU5
INT: check(W,start(malvern)) acknowledge(W,DU5) giveroute(W)
C: [INT: <getroute(C)>]

Figure 3: Information State Prompting Check in [5]

and that C has the intention to find a route. We also assume that W has the turn, and that the presence of the *how can I help* intention triggers an utterance directly. Figure 2, presented above, shows the information state after utterance [2]. The intentions in that figure lead directly to the system producing utterance [3].

Looking a little further ahead in the dialogue, Figure 3 shows the information state after utterance [4].[12] Here we can see in CDU.TOGND.DH (along with the **ack** act CA10, in GND.DH) the dialogue moves that this utterance has generated. Note that the assert, CA11, is only at confidence level 1, indicating lack of sufficient certainty in this interpretation as the town 'Great Malvern'. This lack of certainty and the resulting lack of a relevant SCP in CDU.TOGND lead the deliberation routines to produce an intention to check this proposition rather than to move directly on to another information request. This intention leads to utterance [5], which, after interpretation and updating on the new dialogue acts, leads to the information state in Figure 4. The interesting thing here is the condition which appears in CDU.TOGND.COND as a result of the check; the interpretation of this is that, if C agrees with the check, then W will be committed to the proposition that the starting place is Malvern (C would also be committed to this by way of the direct effects of an agree act).

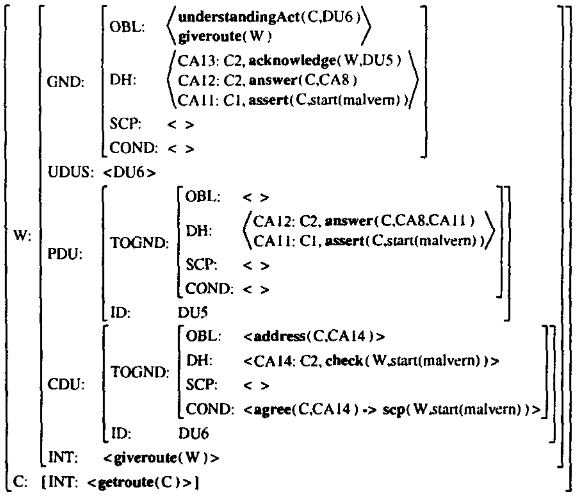
W:
GND:
OBL: understandingAct(C,DU6) giveroute(W)
DH: CA13: C2, acknowledge(W,DU5) CA12: C2, answer(C,CA8) CA11: C1, assert(C,start(malvern))
SCP: < >
COND: < >
UDUS: <DU6>
PDU:
TOGND:
OBL: < >
DH: CA12: C2, answer(C,CA8,CA11) CA11: C1, assert(C,start(malvern))
SCP: < >
COND: < >
ID: DU5
CDU:
TOGND:
OBL: <address(C,CA14)>
DH: <CA14: C2, check(W,start(malvern))>
SCP: < >
COND: <agree(C,CA14) -> scp(W,start(malvern))>
ID: DU6
INT: <giveroute(W)>
C: [INT: <getroute(C)>]

Figure 4: Information State Following Check in [5]

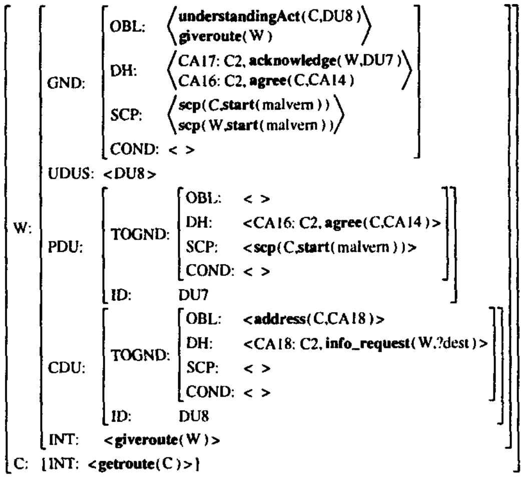
W:
GND:
OBL: understandingAct(C,DU8) giveroute(W)
DH: CA17: C2, acknowledge(W,DU7) CA16: C2, agree(C,CA14)
SCP: scp(C,start(malvern)) scp(W,start(malvern))
COND: < >
UDUS: <DU8>
PDU:
TOGND:
OBL: < >
DH: <CA16: C2, agree(C,CA14)>
SCP: <scp(C,start(malvern))>
COND: < >
ID: DU7
CDU:
TOGND:
OBL: <address(C,CA18)>
DH: <CA18: C2, info_request(W,?dest)>
SCP: < >
COND: < >
ID: DU8
INT: <giveroute(W)>
C: [INT: <getroute(C)>]

Figure 5: Information state following [7]

After C's agreement in [6], the deliberation routine is able to move past discussion of the starting point, and add an intention to ask about the next piece of information, the destination. This leads to producing utterance [7], which also implicitly acknowledges [6], after which C's agreement is grounded, leading to the IS shown in Figure 5. Note that the list in W.GND.SCP in Figure 5 indicates that both C *and* W are committed to the proposition that the starting place is Malvern.

5 Conclusions

It has only been possible here to introduce the basic concerns of the PTT account of dialogue modelling and to pick out one or two illustrative examples to highlight the implementational approach which has

[12] The actual information state contains all the previously established dialogue acts, SCPs and Obligations in GND, from Figure 2 and intermediate utterances. Here we have deleted these aspects from the figures for brevity and clarity.

been assumed. Current and future work is directed towards measuring the theory against more challenging data to test its validity; cases where grounding is less automatic are an obvious source of such tests, and we have identified a few relevant problem cases in the Autoroute dialogues. We do claim, however, that the implementation as it stands validates a number of key aspects of the theory and provides a good basis for future work in dialogue modelling.

Acknowledgments

The TRINDI (Task Oriented Instructional Dialogue) project is supported by the Telematics Applications Programme, Language Engineering Project LE4-8314. Massimo Poesio is supported by an EPSRC Advanced Research Fellowship.

References

J. Allwood. 1976. *Linguistic Communication as Action and Cooperation.* Ph.D. thesis, Göteborg University, Department of Linguistics.

J. Allwood. 1994. Obligations and options in dialogue. *Think Quarterly*, 3:9–18.

M. E. Bratman, D. J. Israel and M. E. Pollack. 1988. Plans and Resource-Bounded Practical Reasoning. *Computational Intelligence*, 4(4).

P. Bretier and M. D. Sadek. 1996. A rational agent as the kernel of a cooperative spoken dialogue system: Implementing a logical theory of interaction. In J. P. Müller, M. J. Wooldridge, and N. R. Jennings, editors, *Intelligent Agents III — Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages (ATAL-96)*, Lecture Notes in Artificial Intelligence. Springer-Verlag, Heidelberg.

J. Calder. 1998. Thistle: diagram display engines and editors. Technical Report HCRC/TR-97, HCRC, University of Edinburgh, Edinburgh.

H. H. Clark and E. F. Schaefer. 1989. Contributing to discourse. *Cognitive Science*, 13:259–294.

H. H. Clark and D. Wilkes-Gibbs. 1986. Referring as a collaborative process. *Cognition*, 22:1–39. Also appears as Chapter 4 in (Clark, 1992).

H. H. Clark. 1992. *Arenas of Language Use.* University of Chicago Press.

P. R. Cohen and H. J. Levesque. 1990. Rational interaction as the basis for communication. In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, *Intentions in Communication.* MIT Press.

Discourse Resource Initiative. 1997. Standards for dialogue coding in natural language processing. Report no. 167, Dagstuhl-Seminar.

B. J. Grosz and C. L. Sidner. 1990. Plans for discourse. In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, *Intentions in Communication.* MIT Press.

J. Kreutel. 1998. An obligation-driven computational model for questions and assertions in dialogue. Master's thesis, Department of Linguistics, University of Edinburgh, Edinburgh.

S. Larsson, P. Bohlin, J. Bos, and D. Traum. 1999. Trindikit manual. Technical Report Deliverable D2.2 - Manual, Trindi.

M. Poesio, R. Cooper, S. Larsson, D. Traum, and C. Matheson. 1999. Annotating conversations for information state update. In *Proceedings of Amstelogue 99, 3rd Workshop on the Semantics and Pragmatics of Dialogues.*

M. Poesio and D. R. Traum. 1997. Conversational actions and discourse situations. *Computational Intelligence*, 13(3).

M. Poesio and D. R. Traum. 1998. Towards an axiomatization of dialogue acts. In *Proceedings of Twendial'98, 13th Twente Workshop on Language Technology*, pages 207–222.

M. D. Sadek. 1991. Dialogue acts are rational plans. In *Proceedings of the ESCA/ETR workshop on multi-modal dialogue.*

M. P. Singh. 1998. Agent communication languages: Rethinking the principles. *IEEE Computer*, 31(12):40–47.

D. R. Traum and J. F. Allen. 1992. A speech acts approach to grounding in conversation. In *Proceedings 2nd International Conference on Spoken Language Processing (ICSLP-92)*, pages 137–40, October.

D. R. Traum and J. F. Allen. 1994. Discourse obligations in dialogue processing. In *Proceedings of the 32nd Annual meeting of the Association for Computational Linguistics*, pages 1–8, June.

D. R. Traum, J. Bos, R. Cooper, S. Larsson, I. Lewin, C. Matheson, and M. Poesio. 1999. A model of dialogue moves and information state revision. Technical Report Deliverable D2.1, Trindi.

D. R. Traum and P. Dillenbourg. 1998. Towards a Normative Model of Grounding in Collaboration. In *Proceedings of the ESSLLI98 workshop on Mutual Knowledge, Common Ground and Public Information.*

D. R. Traum. 1994. *A computational theory of grounding in natural language conversation.* Ph.D. thesis, Computer Science, University of Rochester, New York, December.