Natural Language Engineering **1** (1): 000–000. Printed in the United Kingdom

Information state and dialogue management in the TRINDI Dialogue Move Engine Toolkit

S. Larsson and D. Traum

(*Received 26 March 2000*)

Abstract

We introduce an architecture and toolkit for building dialogue managers currently being developed in the TRINDI project, based on the notions of *information state* and *dialogue move engine*. The aim is to provide a framework for experimenting with implementations of different theories of information state, information state update and dialogue control. A number of dialogue managers are currently being built using the toolkit, and we present a detailed look at one of them. We believe that this framework will make implementation of dialogue processing theories easier, also facilitating comparison of different types of dialogue systems, thus helping to achieve a prerequisite for arriving at a best practice for the development of the dialogue management component of a spoken dialogue system.

1 Introduction

We discuss a candidate model for best practice in the development of the dialogue management component of a spoken dialogue system. At this early stage in development of such systems, there is a wide range of current practice, depending in part on the different tasks that such systems are put to, but also on external factors, such as preference and background of the developers. In order to determine what the *best practice* is, if indeed there can be a single model that is useful for the whole range of such systems, it is necessary to be able to compare and evaluate these components. A big difficulty is that not only are the dialogue management components themselves radically different, but the systems of which they are a part are also quite heterogeneous, making principled comparison of the components difficult, if not impossible.

What we propose is a view of dialogue management functions in terms of *information state*. Key to this approach is identifying the relevant aspects of information in dialogue, how they are updated, and how updating processes are controlled. This simple view can be used to compare a range of approaches and specific theories of dialogue management within the same framework (as well as facilitating hybrid approaches). Use of the framework will thus allow comparison to determine empirically which is the best practice.

The term INFORMATION STATE of a dialogue represents the information necessary to distinguish it from other dialogues, representing the cumulative additions

from previous actions in the dialogue, and motivating future action. For example, statements generally add propositional information; questions generally provide motivation for others to provide specific statements. Information state is also referred to by similar names, such as "conversational score", or "discourse context" and "mental state". Generally, although not necessarily, we will also talk about the information state of participants of the dialogue, representing the information that those participants have at a particular point in the dialogue – what they brought with them to the dialogue, what they pick up, and how they are motivated to act in the (near) future.

We present a particular view toward the formalization of the notion of information state in such a way that allows specific theories of dialogue to be formalized, implemented, tested, compared, and iteratively reformulated. Key to this approach will be a notion of UPDATE of information state, with most updates related to the observation and performance of DIALOGUE MOVES. We will present the framework for modeling information state in the next section. As well as the framework itself, we have been developing a toolkit to allow system designers to build dialogue management components according to their particular theories of information states, this will be described in Section 3 . We have also begun to build several different sorts of systems using the toolkit, some of which will be briefly described in Section 4. We conclude, in Section 5, with a discussion comparing this approach to some other methods for building dialogue management components.

2 The Information State Approach to Dialogue Modeling

We view an information state theory of dialogue modeling as consisting of:

- a description of the **informational components** of the theory of dialogue modeling, including aspects of common context as well as internal motivating factors (e.g., participants, common ground, linguistic and intentional structure, obligations and commitments, beliefs, intentions, user models, etc.)
- **formal representations** of the above components (e.g., as lists, sets, typed feature structures, records, Discourse Representation Structures (DRSs), propositions or modal operators within a logic, etc.)
- a set of **dialogue moves** that will trigger the update of the information state. These will generally also be correlated with externally performed actions, such as particular natural language utterances. A complete theory of dialogue behavior will also require rules for recognizing and realizing the performance of these moves, e.g., with traditional speech and natural language understanding and generation systems.
- a set of **update rules**, that govern the updating of the information state, given various conditions of the current information state and performed dialogue moves, including (in the case of participating in a dialogue rather than just monitoring one) a set of selection rules, that license choosing a particular dialogue move to perform given conditions of the current information state
- an **update strategy** for deciding which rule(s) to select at a given point, from the set of applicable ones. This strategy can range from something as

simple as "pick the first rule that applies" to more sophisticated arbitration mechanisms, based on game theory, utility theory, or statistical methods.

It is important to distinguish information state approaches to dialogue modeling from other, structural, dialogue state approaches. These latter approaches conceive a "legal" dialogue as behaving according to some grammar, with the states representing the results of performing a dialogue move in some previous state, and each state licensing a set of allowable next dialogue moves. The "information" is thus implicit in the state itself and the relationship it plays to other states. It may be difficult to transform an information state view to a dialogue state view, since there's no necessary finiteness restriction on information states (depending on the type of information modeled), and the motivations for update and picking a next dialogue move (using update rules, and update strategy) may rely on only a part of the information available, rather than the whole state. On the other hand, it is very easy to model dialogue state as information state: the information is the dialogue state, itself. This is easily modeled as a register indicating the state number (for finite state models, or a stack for recursive transition networks). The dialogue moves will be the same moves that are used in the dialogue state theory, the update rules will be the transitions in the dialogue state theory, formulated as an update to a new state, given the previous state and performance of the action, and the update strategy will be much the same as in the transition network (i.e., deterministic or non-deterministic, etc.)

Structural dialogue state approaches have often been contrasted with plan-based approaches to dialogue modeling (e.g., by (Cohen, 1996; Sadek & De Mori, 1998)). Structure-based approaches are usually viewed as viable for simple, scripted dialogues, while plan-based approaches, though more complex and difficult to embed in practical dialogue systems, are seen as more amenable to flexible dialogue behavior. Plan-based approaches are also criticized as being more opaque, especially given the large amount of procedural processing and lack of a well-founded semantics for plan-related operations. An information-state approach allows one to fruitfully combine the two approaches, using the advantages of each. The information state may include aspects of dialogue state as well as more mentalistic notions such as beliefs, intentions, plans, etc. Moreover, casting the updates in terms of update rules and strategies that apply the rules under appropriate conditions provides for a more transparent, declarative representation of system behavior than most procedural programs, rendering the resulting dialogue manager easily amenable to experimentation with different dialogue strategies.

In the rest of this section, we will present the aspects of information state in a little more detail. To keep a degree of concreteness, we will make reference to an example theory of information state developed by Cooper and Larsson, described in more detail in (Cooper *et al.*, 1999; Traum *et al.*, 1999; Bohlin *et al.*, 1999a).

2.1 informational components

Information state is usually not conceived of as a monolithic node in a transition network (as with dialogue state), but rather as consisting of several interacting com-

ponents. There is a wide range of possibilities as to what kinds of components should be used to model dialogue. The first choice point comes as to whether to model the participants' internal state, or more external aspects of the dialogue. There are also many ways of modeling the internal state of a participant. One can choose to model the mental state of the agent (attitudes such as belief, desire, intention, along with social correlates such as mutual belief, joint intention, and obligation) (e.g., (Bretier & Sadek, 1996; Traum & Allen, 1994)), or one can take a more structural view of the dialogue, concentrating on the performance of actions and various sorts of accessibility relationships. (e.g., (Ahrenberg *et al.*, 1990)). It may also be useful to distinguish components of information state into *static* and *dynamic* aspects. The former are those aspects of information state that are not expected to change during the course of a dialogue, but are still very useful to modeling the progression of the dialogue. Examples of static information state components could include things like domain knowledge, or knowledge of dialogue conventions. It depends on the type of dialogue being modeled, and the scope of the conversation as to which aspects will be assumed to be static vs. dynamic (e.g., contrasting a knowledge acquisition system vs. a question answering system – the former may want to treat domain knowledge as dynamic, while the latter would see it as static). Marking some information as static may have some advantages in efficient implementation, since various compilation shortcuts and memory allocation could be performed. It is still good practice to have declarative knowledge sources rather than implicit knowledge in program (or finite state automaton) control structure, to be able to reuse the same knowledge for different dialogue situations.

Our example information state is a simplified version of the dialogue game board which has been proposed by Ginzburg (Ginzburg, 1996a; Ginzburg, 1996b; Ginzburg, 1998). There is some information assumed to be private (including *beliefs*, and an *agenda* of actions to perform in the dialogue) and some that is assumed to be shared (propositions assumed to be shared *beliefs*, questions under discussion (*QUD*), and the latest dialogue move performed (*lm*)). This small set of informational elements was used to track the behavior of participants in information seeking dialogues, including asking and answering (potentially elliptical) questions and accumulating information (Cooper *et al.*, 1999; Poesio *et al.*, 1999).

2.2 formal representations

Given a choice of what aspects of the dialogue structure and the participants' internal state to model, the question then arises as to *how* to model them. There are a wide number of choices, from simple abstract data types, to more complex informational systems, such as logics (with associated inference systems) and statistical systems of various flavors. These choices will be related to the particular theory of accessibility of these elements, and will also affect other processing issues, such as comprehensiveness and efficiency. As an example, consider an aspect of information state such as actions to be performed in the dialogue (e.g., an agenda, plan, or other such bundle of intentions). There's a choice as to whether to represent tokens separately (e.g., with some sort of list) or just types, not distinguishing between

multiple tokens of the same type (e.g., with a set). Given a choice of representing a list, there is still the question of accessibility – should it be a FIFO queue, a LIFO stack, or some more open structure, allowing access to the whole list? Likewise, if an agent's beliefs are represented, should this be a set, some sort of ordered list, or a complete logical inference system, in which implicit beliefs are also said to hold given some configuration of explicit beliefs?

(1)
$$\begin{bmatrix} \textsc{private} : \begin{bmatrix} \textsc{bel} : \textsc{Set}(\textsc{Prop}) \\ \textsc{agenda} : \textsc{Stack}(\textsc{Action}) \end{bmatrix} \\ \textsc{shared} : \begin{bmatrix} \textsc{bel} : \textsc{Set}(\textsc{Prop}) \\ \textsc{qud} : \textsc{Stack}(\textsc{Question}) \\ \textsc{lm} : \textsc{Move} \end{bmatrix} \end{bmatrix}$$

Our example information state is represented as a record (Cooper & Larsson, 1999), as shown in (1). Here private and shared information are represented as sub-records, each with several fields. Each field is either a value, a set or a stack, with the type of information (proposition, action, question or move) indicated.

2.3 dialogue moves

Dialogue moves are meant to serve as an abstraction between the large number of different possible messages that can be sent (especially in a natural language) and the types of update to be made on the basis of performed utterances. Dialogue moves can also provide an abstract level for content generation. There are also a number of dialogue move taxonomies to choose from; some principles regarding this issue are outlined in (Traum, 1999). There must be at least sufficient types of dialogue moves to provide the different kinds of updates desired. The set of dialogue moves to choose is also influenced by the task of language interpretation – how easy will it be to (reliably) determine that one move vs. another has been performed? Another complicating issue is how to capture the inherent multi-functionality of utterances – with complex moves and move taxonomies, where each move has multiple functions, or with a set of more simple moves that corresponding to each utterance. Dialogue moves do not need to be conceived of as *speech-acts* in the sense of (Searle, 1969), but can be any mediating input, e.g., logical forms or even word-lattices augmented with likelihoods.

Our example information state theory uses only two moves, **ask** and **answer**.

2.4 update rules

Update rules formalize the way that information state is changed as the dialogue progresses. Each rule consists of a set of applicability conditions and a set of effects. The applicability conditions specify aspects of the information state that must be present for the rule to be appropriate. Effects are changes that will be made to the information state when the rule has been applied (assuming that all conditions hold). Update rules are meant to encapsulate coherent bundles of change to the information state, given a particular theory of dialogue. While conditions and effects are related to coherent operations on an abstract datatype, regardless of application

to a theory of dialogue, update rules specialize these operations further to be specific (potentially complex) building blocks of a dialogue theory.

Continuing our example information state theory, the rule for adding a question to QUD if an *ask* move has been performed is shown in (2). This rule has two conditions: that the latest move was of type *ask*, and that the top of the agenda was the action of raising a question, the effects are to pop this item from the agenda, and push the question that is the content of both the *raise* agenda item and the *ask* dialogue move. Other update rules in our sample information state theory include rules to remove the question from the QUD if an *answer* move has been performed, to add an action to answer the top question on the QUD (if asked by the other agent), and to select an answer move on the basis of the top of the agenda.

(2)
$$\begin{array}{l} \textsc{u-rule}\text{: } \textbf{integrateSysAsk} \\ \textsc{pre}\text{: } \left\{ \begin{array}{l} \mathrm{val}(\textsc{shared.lm}, \mathrm{ask}(\mathrm{usr},Q)) \\ \mathrm{fst}(\textsc{private.agenda}, \text{raise-question}(Q)) \end{array} \right. \\ \textsc{eff}\text{: } \left\{ \begin{array}{l} \mathrm{push}(\textsc{shared.qud}, Q) \\ \mathrm{pop}(\textsc{private.agenda}) \end{array} \right. \end{array}$$

2.5 update strategy

Along with the set of update rules, a strategy for how to apply the rules is needed. This is, in many cases, going to be crucial for the design of the rules themselves. Given a particular update strategy, one may need to make adaptations to the rules, and perhaps also aspects of the information state itself, in order to guarantee a particular sequence of rule applications. There's also a question of whether to have separate strategies for choosing different types of update rules (i.e., for observation and selection of dialogue moves), and whether these processes can be ordered or asynchronously applied. Some of the types of update strategies we have considered include:

1. Take the first rule that applies (iteratively until no rules apply)
2. Apply each rule (if applicable) in sequence
3. Apply rules according to class
4. Choose among applicable rules using probabilistic information
5. Present choices to user to decide (for development modes)

For our sample information state, we use algorithm 1 in the list.

2.6 Discussion

There is thus a synergy between choices for the components of information state: the conceptual notions, formal representations, dialogue moves, update rules, and update strategy. A complete theory of dialogue update will need to include a smoothly interacting combination of these aspects. However, it is still very possible to hold some of them constant while trying out other possibilities for the others. E.g., different ways of representing a concept, different rules for doing updates, or different update strategies for applying rules. This experimentation is a central part of our

current work. In the next section, we present an architecture for implementing and testing this approach to information state update.

Given the sample informational components described above, we can track information states in simple information-seeking dialogue. A short question-answer exchange is illustrated in 3. Before the exchange, the system has an agenda item to raise the question about the user's destination. This meets the conditions for the update rule that selects an **ask** move. After the system utterance, the update algorithm will first apply the rule shown in (2). After the user utterance, the rule for integrating user answers will check that the answer matches the question topmost on QUD, and will then pop the question off the QUD, and integrate the answer into the shared beliefs.

(3) `Sys: Where do you want to go?`
`Usr: Malvern`

3 TrindiKit: A Dialogue Move Engine Toolkit

The information-state approach to dialogue modeling described in the previous section yields a computational theory of dialogue that naturally lends itself to implementation. We call the implementation of such a theory of dialogue dynamics a *Dialogue Move Engine* (DME), since its main functions are updating information state based on the observance of moves and selecting moves to be performed. This DME, together with some connective material, forms the dialogue management and discourse tracking aspects of a dialogue system. A complete dialogue system would need modules (or sets of modules) to perform at least the following additional functions:

- user interface – to receive input from and present output to the user.
- interpretation – to calculate from the input which dialogue moves have been performed, adding these to a special *latest move* part of the information state.
- generation – to take the contents of the special *next move* part of the information state, and produce the output.
- control – to wire together the other modules, either serially or in parallel.

As part of the TRINDI project, and described more fully in (Larsson *et al.*, 1999), we are developing a DME toolkit called TrindiKit, which provides the basic architecture as well as facilities for implementing theories of information state. The general architecture of TrindiKit is shown schematically in Figure 1. Dialogue management is handled by the control module, the DME[1] and the information state.[2] The DME can consist of one or multiple Update Modules, each containing

[1] We use the term DME in two senses, a broader sense, meaning the implementation of the five components of the information-state based dialogue modeling approach described in the previous section, and more narrowly, within such an implementation, as the collection of modules implementing the update algorithm, applying update rules and "consuming" and "producing" dialogue moves, in the process.

[2] Likewise, Information State is used both to denote the components of the theoretical approach to dialogue modeling, and the specific blackboard-like system module that allows inspection and update (via update rules) of the current state.

a different set of rules and potentially a different algorithm. Typically the DME contains an update module and a selection module, encapsulating the functions of integrating observed dialogue moves, and selecting new ones for the system to say. Under an architecture of this sort, it is up to the control module as to how to interleave these two functions.

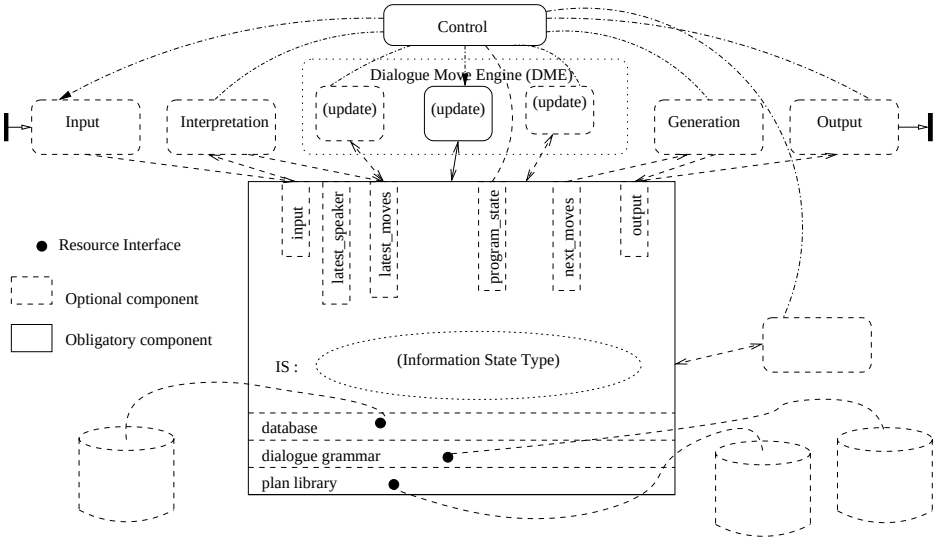


Fig. 1. The TRINDI DME Architecture

The components of the architecture are the the total information state (TIS), consisting of the information state proper (IS), as well as interface variables for communicating with language processing modules and non-linguistic resources; the Dialogue Move Engine, consisting of one or more DME modules; other dialogue system modules (DME-external); and a control module, wiring together the other modules, either in sequence or through some asynchronous mechanism. The IS is specified using abstract data types, each permitting a specific set of queries to inspect the type and operations to change it. These are the building blocks of update rules, which can be used by other modules to inspect and change the information state in coherent ways.

Some of the components are obligatory, and others are optional or user-defined. To build a system, one must minimally supply an information state type, at least one DME module, consisting of TIS update rules and an algorithm, and a control module, operating according to a control algorithm. Any useful system is also likely to need additional modules, e.g. for getting input from the user, interpreting this input, generating system utterances, and providing output for the user. The other modules will generally communicate with the rest of the system through designated interface variables. Other resources, such as databases, plan libraries, etc., can also be integrated into the system, using an interface that allows the same kinds of queries and operations as for the IS proper, allowing update rules to be oblivious to

whether the components are part of the information state or external resources. A possible setup of DME-external modules, is indicated by dashed lines in Figure 1. Note that the illustrated module setup is just an example. The TRINDIKIT provides methods for defining any number of both DME-modules and DME-external modules, with associated interface variables.

Apart from the general architecture defined above, the TRINDIKIT provides definitions of datatypes (for use in TIS variable definitions), a language and format for specifying TIS update rules, methods for accessing the TIS, an algorithm definition language for DME and control modules, default modules for input, interpretation, generation and output, methods for converting items from one type to another, methods for visually inspecting the TIS, and debugging facilities.

4 Implementations using TrindiKit

A number of systems are currently being developed using the TrindiKit. We will look at two of them in some detail: GoDiS, developed at Gothenburg University by Cooper, Larsson and Bohlin (Bohlin *et al.*, 1999a), which uses an extension of the information state theory used as an example in Section 2, and the EDIS system (Matheson *et al.*, 2000), developed at University of Edinburgh uses a notion of information state based on (Poesio & Traum, 1998). More details of these and other TrindiKit systems can be found in (Bos *et al.*, 1999).

4.1 GoDiS

GoDiS is an experimental dialogue system built using the TrindiKit. It uses fairly simple algorithms for control, update and selection modules, keyword-based interpretation and template-based generation. The notion of information state used in GoDiS is an extension of that illustrated in Section 2, and is currently able to handle simple grounding phenomena as well as *question accommodation*, allowing users to answer unasked but salient questions. The GoDiS system currently distinguishes 8 dialogue move types: *ask, answer, repeat, request_repeat, greet, goodbye, thank* and *quit*.

The main division in the information state is between information that is private to the system and that which is assumed to be shared between the dialogue participants. What we mean by shared information here is that which has been explicitly established during the conversation.[3] The SHARED field is divided into three subfields. One subfield is a set of propositions which the agent assumes for the sake of the conversation. The second subfield is for a stack of questions under discussion (QUD). These are questions that have been raised and are currently under discussion in the dialogue. The final SHARED field contains information about the latest move (speaker, move type and content).

The private information contains four subfields. The BEL field contains propositions that the system holds to be true. The AGENDA field contains the system's

[3] akin to the "conversational scoreboard" in (Lewis, 1979).

short term intentions for the next turn. The PLAN field is a list of actions that are longer-term dialogue goals. This plan can, however, be changed during the course of the conversation. We also have a field TMP that mirrors the shared fields. This field keeps track of shared information that has not yet been grounded, i.e. confirmed as having been understood by the other dialogue participant. In this way it is easy to delete information which the agent has optimistically assumed to have become shared if it should turn out that the other dialogue participant does not understand or accept it. A variant of the system (with a few different update rules) allows a system to pursue a *cautious* rather than *optimistic* strategy with respect to grounding. In this version, information will at first only be placed on TMP until it has been acknowledged by the other dialogue participant whereupon it can be moved from TMP to the appropriate shared field, in a manner similar to the way CDU works in the EDIS system, described in the next section.

The update rules in GoDiS include an ability to perform several kinds of *accommodation* when the appropriate expected structures are not on the QUD or Plan. In real human-human dialogues, dialogue participants can address questions that have not been explicitly raised in the dialogue. However, the question is still needed, to allow interpretation of elliptical utterances. An example from a travel agency dialogue[4] Is shown in (4).

(4) J: what month do you want to go
P: well around 3rd 4th april / some time there
P: as cheap as possible

The strategy we adopt for interpreting elliptical utterances is to think of them as short answers to questions on QUD (in the sense of (Ginzburg, 1996a; Ginzburg, 1996b; Ginzburg, 1998)). A suitable question here is *What kind of price does C want for the ticket?*. This question is not under discussion at the point when the customer says "as cheap as possible", but it can be inferred from the context. In fact J will have as part of his plan, the action of raising this question . On our analysis it is this fact which enables J to interpret the ellipsis. He finds the matching question on his plan, accommodates by placing it on QUD and then continues with the integration of the information expressed by *as cheap as possible* as normal.

A similar situation arises when there is no current plan to use as context for interpretation. For example, if a travel agent discovers that his customer wishes to get information about a flight he will adopt a plan to ask her where she wants to go, when she wants to go, what price class she wants and so on. In cases where the customer does not state her errand explicitly, but rather answers some question(s) (e.g. about destination and means of transport), the agent must infer what the task is. We call this process *task accommodation*, and it is closely related to question accommodation.

A sample information state, resulting from the dialogue in (5), is shown in (6).

[4] This dialogue has been translated from Swedish. It was collected by the University of Lund.

(5) `Sys: Welcome to the travel agency!`
`Usr: flights to paris`
`Sys: What city do you want to go from?`

(6)
$$\left[\begin{array}{lcl} \textsc{private} & = & \left[\begin{array}{lcl} \textsc{bel} & = & \{\} \\ \textsc{agenda} & = & \langle\rangle \\ \textsc{plan} & = & \left\langle\begin{array}{l} \textsc{raise}(\mathrm{R}\hat{}(\textsc{return}{=}\mathrm{R})), \\ \textsc{raise}(\mathrm{M}\hat{}(\textsc{month}{=}\mathrm{M})), \\ \textsc{raise}(\mathrm{C}\hat{}(\textsc{class}{=}\mathrm{C})), \\ \textsc{respond}(\mathrm{P}\hat{}(\textsc{price}{=}\mathrm{P})) \end{array}\right\rangle \\ \textsc{tmp} & = & (\text{same as } \textsc{shared}) \end{array}\right] \\ \textsc{shared} & = & \left[\begin{array}{lcl} \textsc{bel} & = & \{(\textsc{to}{=}\textsc{paris}),(\textsc{how}{=}\textsc{plane})\} \\ \textsc{qud} & = & \langle \mathrm{X}\hat{}(\textsc{from}{=}\mathrm{X})\rangle \\ \textsc{lm} & = & \textsc{ask}(\textsc{sys},\mathrm{Y}\hat{}(\textsc{from}{=}\mathrm{Y})) \end{array}\right] \end{array}\right]$$

The user utterance is seen as answering two questions; however, no task or plan has yet been established and no questions have been raised. To be able to integrate the utterance, the system must find a task associated with a plan which includes the raising of questions which match the answers given. Once the task (in this case getting price information about a trip) has been accommodated and the plan entered into the PLAN field, the questions can be accommodated and the answers integrated. As a consequence of this process, the information state now contains a plan which guides the system behavior.

Given the kind of information state illustrated in (6), we can provide update rules which accommodate questions. A formalization of the **accommodate_question** move is given in (7). When interpreting the latest utterance by the other participant, the system makes the assumption that it was a **reply** move with content A. This assumption requires accommodating some question Q such that A is a relevant answer to Q. The condition "answer-to(A, Q)" is true if A is a relevant answer to Q given the current information state, according to some (possibly domain-dependent) definition of question-answer relevance.

(7)
$$\begin{array}{l} \textsc{u-rule}: \textbf{accommodateQuestion}(Q, A) \\ \textsc{pre}: \left\{\begin{array}{l} \text{valRec}(\textsc{shared.lm}, \text{ answer}(\text{usr},A)), \\ \text{inRec}(\textsc{private.plan}, \text{ raise}(Q)) \\ \text{answer-to}(\ A, Q\ ) \end{array}\right. \\ \textsc{eff}: \left\{\begin{array}{l} \text{delRec}(\textsc{private.plan}, \text{ raise}(Q)) \\ \text{pushRec}(\textsc{shared.qud}, \ Q) \end{array}\right. \end{array}$$

GoDiS uses an update algorithm where different types of rules are applied at different stages of the update process. There are currently 6 rule types and 28 rules:

- *refill*: puts new actions on the AGENDA (8 rules)
- *grounding*: handles grounding (2 rules)
- *integrate*: integrates the effects of the latest move (12 rules)
- *accommodate*: handles question (and task) accommodation (3 rules)
- *database*: performs database search (2 rules)
- *store*: stores current SHARED in PRIVATE.TMP (1 rule)

and the update algorithm is shown in (8).

```
(8) if ( latest_moves = failed )
      then repeat( refill )
      else ( grounding;
             repeat( integrate;
                     accommodate;
                     database );
             if ( latest_speaker = usr )
             then repeat( refill )
             else store
             )
```

If the interpreter failed on the latest utterance, the agenda is refilled (usually with a request_repeat move); otherwise, a *grounding* rule is applied which enters the latest move into SHARED.LM. After grounding, the effects of the move are integrated into the information state, which may require question and/or task accommodation. If necessary, database searches are performed. Finally, if the latest move was performed by the user, the agenda is refilled; otherwise, the current SHARED field is stored in PRIVATE.TMP in case the next optimistic grounding assumption should prove to be wrong.

The current control algorithm in GoDiS simply calls each module in turn in a serial fashion. The selection algorithm simply picks the first applicable rule.

4.2 EDIS

The EDIS system (Matheson *et al.*, 2000), uses a notion of information state based on (Poesio & Traum, 1997; Poesio & Traum, 1998), using the record representation used for coding information states in (Cooper *et al.*, 1999; Poesio *et al.*, 1999). The informational components consist of a common ground part, a semi-public part, and a private part, as with GoDiS. The common part includes four types of information: obligations of dialogue participants to perform actions (OBL), social commitments that participants have that propositions hold (SCP), a dialogue history of acts that have been performed (DH), and conditional statements that will establish obligations or commitments, given the performance of appropriately typed dialogue acts (COND). The semi-public part, analogous to TEMP in GoDiS, are a collection of *discourse units* (DUs) (Traum & Hinkelman, 1992), which represent coherent bundles of information that are *grounded* (added to the common ground (Clark & Schaefer, 1987)) together. Private information includes the intentions of the agent being modeled. The formal representations of EDIS are shown in (9), where PT-R is a record containing the type of information contained in common ground, shown to the right. Only two DUs are represented, CDU for current, and PDU, the previous one. UDUs is a list of the DUs (which may include the ones identified in PDU and/or CDU) which are not grounded.

$$(9)\quad \begin{bmatrix} \text{G} & : & \textit{PT-R} \\ \text{CDU} & : & \begin{bmatrix} \text{C} & : & \textit{PT-R} \\ \text{ID} & : & \text{DU-ID} \end{bmatrix} \\ \text{PDU} & : & \begin{bmatrix} \text{C} & : & \textit{PT-R} \\ \text{ID} & : & \text{DU-ID} \end{bmatrix} \\ \text{UDUs} & : & \text{List(DU-ID)} \\ \text{INT} & : & \text{List(Action)} \end{bmatrix} \qquad \textit{PT-R} : \begin{bmatrix} \text{DH} & : & \text{List(Action)} \\ \text{OBL} & : & \text{List(Action)} \\ \text{SCP} & : & \text{List(Prop)} \\ \text{COND} & : & \text{List(Action)} \end{bmatrix}$$

EDIS uses a modified version of the dialog acts from the DRI coding scheme (Dis-

course Resource Initiative, 1997), giving them precise effect conditions according to aspects of the information state in (9). EDIS uses the same general pipelining of modules as GoDiS, however the update algorithm is a bit different. Whenever a set of dialogue acts are placed in `latest_moves`, the following algorithm is applied, applying a set of update rules in each step.

(10) a. Create a new DU and push it on top of UDUs (and set CDU to this one, moving the old CDU to PDU).
b. Perform updates on the basis of backwards grounding acts, such as merging PDU.C with G for an `acknowledgement`.
c. 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.
d. Apply update rules to all parts of the IS which contain newly added acts.

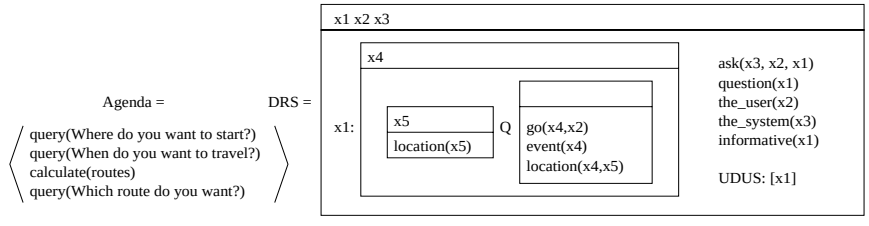
There is also a deliberation step, applied for each system turn, which leads to the system developing new intentions on the basis of obligations, potential obligations that would result from conditions (in the COND field of G or CDU) if an intended act were performed, as well as insufficiently understood dialogue acts and intentions to perform complex acts. Following deliberation, dialogue acts are selected to fulfill any intentions, and placed in the `next_moves` interface variable, for the generation module to act on.

4.3 Other TrindiKit Systems

We briefly mention two other TrindiKit systems. More details on the theories of information state underlying these systems can be found in (Traum *et al.*, 1999), while details of the systems themselves can be found in (Bos *et al.*, 1999).

The MIDAS system uses the DRS structures of DRT (Kamp & Reyle, 1993) as a major component of its information state. As part of the root DRS will be subordinate DRSes representing events mentioned in the dialogue, as well as tracking of grounding, using a simplified version of the theory proposed in (Poesio & Traum, 1998). Multiple theorem provers are used both for pragmatic aspects of dialogue act interpretation and to implement some of the conditional tests on the update rules. An example of a MIDAS information state can be seen in (11), in which the DRS represents a question the system has asked one of a set of questions, with other future questions remaining on an agenda.

(11)



Another system focuses on conversational game theory, recasting the existing Autoroute system (Lewin, 1998) in the current framework. Conversational games

are formalized as recursive transition networks. The top-level information state type is a record, as shown in (12). The top level distinguishes a dialogue participants role as rational agent and conversational game-player, with the intuition that the details of the latter should not be major factors in the former. Actions and agenda-items are themselves records with multiple fields.

(12)
$$\begin{bmatrix} \textsc{rational\_agent} & : & \begin{bmatrix} \textsc{plan} & : & \textsc{Stack(Action)} \\ \textsc{scoreboard} & : & \textsc{Set(Prop)} \end{bmatrix} \\ \textsc{game\_player} & : & \begin{bmatrix} \textsc{agendaitem} & : & \textsc{Stack(Agendaitem)} \\ \textsc{currtoken} & : & \textsc{Token} \\ \textsc{alltokens} & : & \textsc{Stack(Set(Prop))} \end{bmatrix} \end{bmatrix}$$

5 Discussion

We feel the work presented here makes several modest contributions toward an objective of "best practice" in dialogue systems design. The first is a toolkit for building dialogue managers that can be closer to the level of a theory of dialogue processing than is available in most systems. There are a number of dialogue system toolkits available (e.g., those reviewed in (Bohlin *et al.*, 1999b; Luz, 1999)), however these are mostly at the level of wiring together global dialogue states and actions, rather than at the level of considering and acting on specific information in the dialogue context. We hope that a tool such as TrindiKit, and especially the computational view of dialogue modeling as using precisely defined information states and updates can help bridge the currently fairly considerable gap between practical dialogue systems and more theoretical approaches to dialogue.

Second, we hope that the specification of components of information state can help clarify both what is meant by *dialogue management* and the role of different aspects of dialogue management. There are a number of dialogue systems functions closely related to dialogue management, though there is a wide difference in particular systems as to how these functions are divided into (conceptual or implementational) modules, and whether, even if there is a module called *dialogue manager*, it will have these functions. For example, in many systems, the key dialogue management task is control of the dialogue, e.g., by tracking dialogue state and performing state-specific functions. In other systems, (e.g., (Allen *et al.*, 1996)), there is a central hub controller, and the dialogue manager is one of many "spokes". Control can also be distinguished into program control and data-flow aspects. If data flow is too implicit or inaccessible, this may block certain kinds of dialogue management functions, such as timely explicit feedback on the source of problems, when the system has trouble deciding what was said or what to do. Finally, there is a distinction between features of the current state of processing and operations to perform on the basis of that state. The framework presented in this paper, with clear separations between information state, update mechanisms, and control flow can help clarify some of these tricky issues, allowing for more flexible experimentation on a wide range of dimensions.

Indeed, perhaps the main advantage of implementing dialogue systems in a framework like this one is the ability to compare many aspects of such systems, e.g., the

different sorts of examples presented in Section 4. While any well-designed system (e.g., (Smith & Hipp, 1994; Walker *et al.*, 1998)) should allow a range of experimentation of different parameters, it is much more difficult to compare different systems or approaches to the same phenomena. By specifying different notions of information state (and related updates and algorithms), we can compare, e.g., whether an approach to question-answering is best made with reference to dialogue game grammars, *Questions Under Discussion*, or elements of rational and social agency, such as intentions and obligations (see also (Larsson, 1998)).

5.1 Next Steps

We are currently extending the work presented here in several directions. First, gaining more experience with using the TrindiKit to build dialogue managers, and improving the tools provided to the system designer. Along with this will come evaluation, both of the resulting systems and the utility of the toolkit for building such systems.

Another avenue is visualization tools to better examine the contents of the information state as the dialogue progresses, and allow for easier writing of rules. We are currently exploring integration with the THISTLE visualization tool (Calder, 1998). THISTLE is a parameterizable display engine and editor for diagrams, allowing the inclusion of interactive diagrams within Web pages.

Acknowledgments

This work was supported by the TRINDI (Task Oriented Instructional Dialogue) project, EU TELEMATICS APPLICATIONS Programme, Language Engineering Project LE4-8314. Other members of the consortium were instrumental in developing the ideas and systems described here. Peter Bohlin and Johan Bos helped develop the Trindikit distribution. GoDiS was developed by the first author and Bohlin, Robin Cooper and Elisabet Engdahl. EDIS was developed by Colin Matheson, Massimo Poesio, and the second author. The MIDAS system was developed by Bos. Ian Lewin adeveloped the SRI conversational game theory system. In addition, all of the above and other TRINDI consortium participants have contributed to the development of the framework presented here. As usual, all mistakes and misrepresentations are due to the authors of this paper.

References

Ahrenberg, Lars, Dahlbäck, Nils, & Jönsson, Arne. 1990. Discourse representation and discourse management for a natural language dialogue system. *In: Proceedings of the second nordic conference on text comprehension in man and machine.*

Allen, James F., Miller, Bradford W., Ringger, Eric K., & Sikorski, Teresa. 1996. A robust system for natural spoken dialogue. *Pages 62–70 of: Proceedings of the 1996 annual meeting of the association for computational linguistics (acl-96).*

Bohlin, Peter, Cooper, Robin, Engdahl, Elisabet, & Larsson, Staffan. 1999a. Information states and dialogue move engines. *Pages 25–31 of: Proceedings of the ijcai99 workshop: Knowledge and reasoning in practical dialogue systems.*

Bohlin, Peter, Bos, Johan, , Larsson, Staffan, Lewin, Ian, Matheson, Colin, & Milward, David. 1999b. *Survey of existing interactive systems.* Deliverable D1.3. Trindi Project.

Bos, Johan, Bohlin, Peter, Larsson, Staffan, Lewin, Ian, & Matheson, Colin. 1999. *Evaluation of the model with respect to restricted dialogue systems.* Tech. rept. Deliverable D3.2. Trindi.

Bretier, P., & Sadek, M. D. 1996. A rational agent as the kernel of a cooperative spoken dialogue system: Implementing a logical theory of interaction. *In:* Müller, J. P., Wooldridge, M. J., & Jennings, N. R. (eds), *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.

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

Clark, Herbert H., & Schaefer, Edward F. 1987. Collaborating on contributions to conversation. *Language and cognitive processes*, **2**, 1–23.

Cohen, Phil. 1996. Dialogue modeling. *Chap. 6.3 of:* Cole, Ron, Mariani, Joseph, Uszkoreit, Hans, Zaenen, Annie, & Zue, Victor (eds), *Survey of the state of the art of human language technology.* Cambridge, MA: Cambridge University Press.

Cooper, R., Larsson, S., Matheson, C., Poesio, M., & Traum, D. 1999. *Coding instructional dialogue for information states.* Deliverable D1.1. Trindi Project.

Cooper, Robin, & Larsson, Staffan. 1999. Dialogue moves and information states. *In:* Bunt, H.C., & Thijsse, E. C. G. (eds), *Proceedings of the third international workshop on computational semantics.*

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

Ginzburg, J. 1996a. Dynamics and the semantics of dialogue. *In:* Seligman, Jerry, & Westerståhl, Dag (eds), *Logic, language and computation, vol. 1.* CSLI Publications.

Ginzburg, J. 1996b. Interrogatives: Questions, facts and dialogue. *In:* Lappin, Shalom (ed), *The handbook of contemporary semantic theory.* Blackwell, Oxford.

Ginzburg, J. 1998. Clarifying utterances. *Pages 11–30 of:* Hulstijn, J., & Nijholt, A. (eds), *Proc. of the twente workshop on the formal semantics and pragmatics of dialogues.* Universiteit Twente, Faculteit Informatica, Enschede.

Kamp, H., & Reyle, U. 1993. *From discourse to logic.* Dordrecht: D. Reidel.

Larsson, Staffan. 1998. Questions under discussion and dialogue moves. *In: Proceedings of twlt13/twendial '98: Formal semantics and pragmatics of dialogue.*

Larsson, Staffan, Bohlin, Peter, Bos, Johan, & Traum, David. 1999. *Trindikit manual.* Tech. rept. Deliverable D2.2 - Manual. Trindi.

Lewin, Ian. 1998. *The autoroute dialogue demonstrator.* Tech. rept. CRC-073. SRI Cambridge Computer Science Research Centre.

Lewis, David K. 1979. Scorekeeping in a language game. *Journal of philosophical logic*, **8**(3), 339–359.

Luz, Saturnio. 1999. *State-of-the-art survey of dialogue management tools.* Deliverable D2.7a. Disc Project.

Matheson, Colin, Poesio, Massimo, & Traum, David. 2000. Modelling grounding and discourse obligations using update rules. *In: Proceedings of the first conference of the north american chapter of the association for computational linguistics.*

Poesio, M., Cooper, R., Larsson, S., Matheson, C., & Traum., D. 1999. Annotating conversations for information state update. *In: Proceedings of amstelogue'99 workshop on the semantics and pragmatics of dialogue.*

Poesio, Massimo, & Traum, David R. 1997. Conversational actions and discourse situations. *Computational intelligence*, **13**(3).

Poesio, Massimo, & Traum, David R. 1998. Towards an axiomatization of dialogue acts. *In: Proceedings of twendial'98, 13th twente workshop on language technology: Formal semantics and pragmatics of dialogue.*

Sadek, David, & De Mori, Renato. 1998. Dialogue systems. *In:* Mori, R. De (ed), *Spoken dialogues with computers.* Academic Press.

Searle, John R. 1969. *Speech acts.* New York: Cambridge University Press.

Smith, Ronnie W., & Hipp, D. Richard. 1994. *Spoken natural language dialog systems: A practical approach.* Oxford University Press.

Traum, David, Bos, Johan, Cooper, Robin, Larsson, Staffan, Lewin, Ian, Matheson, Colin, & Poesio, Massimo. 1999. *A model of dialogue moves and information state revision.* Tech. rept. Deliverable D2.1. Trindi.

Traum, David R. 1999. 20 questions on dialogue act taxonomies. *In: Proceedings of amstelogue'99 workshop on the semantics and pragmatics of dialogue.*

Traum, David R., & Allen, James F. 1994. Discourse obligations in dialogue processing. *Pages 1–8 of: Proceedings of the 32nd annual meeting of the association for computational linguistics.*

Traum, David R., & Hinkelman, Elizabeth A. 1992. Conversation acts in task-oriented spoken dialogue. *Computational intelligence*, **8**(3), 575–599. Special Issue on Non-literal language.

Walker, Marilyn A., Fromer, Jeanne C., & Narayanan, Shrikanth. 1998. Learning optimal dialogue strategies: A case study of a spoken dialogue agent for email. *In: Proceedings coling-acl-98.*