# Annotating Conversations for Information State Updates

Massimo Poesio†, Robin Cooper‡, Staffan Larsson‡,
Colin Matheson†, and David Traum♠
† University of Edinburgh, ‡ Göteborg University, ♠ University of Maryland

**Abstract**

We present an experiment in annotating a dialogue using two different notions of information state: a stripped down variant of Ginzburg's view of the dialogue game board, including questions under discussion (QUD), and the dialogue model of Poesio and Traum which builds upon previous work by Traum on the grounding process. In this first experiment we annotated a task oriented dialogue with the two notions of information state. We also present two tools useful for this kind of annotation: an annotation tool called *TranScript*, and the *Thistle Diagram Editor*. While annotating for information states is not suitable for large-scale annotation, we do feel that the methodology we are developing could be useful both for people who are interested in studying dialogue acts and for people who are building dialogue systems.

## 1 Introduction

Probably the central issues in analysis of dialogues are the questions of *why language participants say what they say*, and *what are the effects of those utterances.* These questions are obviously closely linked, because much of the reason for saying something is based on what has been said before. Modeling at least some aspect of the answers to these questions is also crucial for designing computational systems to engage in dialogue: these systems need to have procedures for determining what to say next and how to update their internal state on the basis of utterances. The general problem of coping with these issues is often termed *dialogue management*, and the components of the system most centrally concerned with these questions are termed *dialogue managers.* There are many different ways to model the process of finding these answers, ranging in degrees of complexity and closeness of approximation to human processing.

One very simple strategy is to either just produce particular utterances in sequence, or directly compute a response on the basis of the preceding utterance from the user. This is the strategy adopted by Eliza and other very simple programs. The problem with this approach is that often a context of more than just the previous utterance is needed to produce an appropriate next utterance. A more sophisticated approach involves using a *grammar* of acceptable dialogues, usually encoded as a finite state or recursive transition network, where the utterances represent transitions between states, and the states represent the context needed to decide what to say next. This approach also has its limitations, since it may lead to very large networks if all of the necessary context is encoded by differences in states. One common approach is to treat the utterances as encoding one of a limited set of abstract *moves*, and transitions are specified in terms of these moves, with other information being represented in other ways (e.g., variable or data structure values).

A more general approach is to view the dialogues in terms of the relevant *information* that the dialogue participants have (perhaps in addition to a notion of state in a network). From this vantage point, the main effect of an utterance is to change this information in some way, and the information is used by the participants to decide what to do next. The big question, then, is what kind of information is useful for this process. We can classify dialogue related information into two broad categories: *static*, which contains information critical to behaving appropriately in the dialogue, but does not actually change in the dialogue itself, and *dynamic*, which does change as the dialogue progresses, often after each utterance or sometimes in between utterances. The static information state will include both information about the domain such as how to do things, as well as meta-information about the dialogue participation process, such as how to update the dynamic state. The dynamic state will include the actual changes that motivate particular actions. *Moves* can now be seen as optional, since, while they might compactly serve to indicate the set of updates to the information state, one could also more directly represent the information change coming from the utterance without classifying the latter into one move or another.

The information state approach can also easily model the previous approaches, as well. For example, the connectivity of the network, would be the static information state, while the dynamic information state would be the particular current state, as well as any other information that might be useful for determining a next transition.

There are also different approaches to modeling the information state in terms closer to the dialogue itself or the mental and interactional states of the participants. In terms of mental states of dialogue participants, common mental attitudes include *Belief* (the participants' model of the world), *Desire* (what the participant wants the world to be like), and *Intentions* (plans the participant has developed for how to change the world). These are also often augmented with other attitudes, such as *Options* (ways that the agent can change things), and social states, with more than one agent involved. These latter include "mutual belief" or *common ground* which represents information that the participants believe to be shared, as well as various sorts of social commitment of one agent to others, including actions to be performed, or representations of how things are.

Other types of information refer more to the situation of the dialogue itself rather than the mental states of the participants. These include the *turn*, or which participant has the right to speak, and some notion of the *topic structure*, or what the participants are currently talking about. This notion is conceived of in many different ways, for example, in terms of the *intentional structure* [14] of how current topics relate to overall objectives, or in terms of the questions under discussion [12], which licenses what kinds of utterances may be made and understood.

A major goal of the TRINDI project is to be able to precisely characterize information states in dialogue, as well as their relationship to moves and providing answers to the important questions mentioned above. Doing this may provide a sound basis for empirical studies on which sorts of information states may be necessary or sufficient for engaging in particular kinds of dialogues.

# 2 Characterizing Information States

The characterization of the state of the conversation adopted in the spoken dialogue systems currently in real use, or close to actual use (e.g., [1]) can be represented in terms of feature structures as in (1): a list of fields which the system must fill before being able to ask a query.

(1) $$\begin{bmatrix} l_1 & = & a_1 \\ l_2 & = & a_2 \\ \ldots & & \\ l_n & = & a_n \end{bmatrix}$$

For example, in the Autoroute domain, the goal of the system is to identify the start and end points of the trip, and the departure time; this information can be represented as in (2).[1]

(2) $$\begin{bmatrix} \text{START} & = & \\ \text{END} & = & \\ \text{STIME} & = & \end{bmatrix}$$

This notation can be interpreted in various ways. One interpretation we have adopted is that in terms of typed records as discussed in [6, 5]. Using the notation $a : T$ to represent the judgment that $a$ is of type $T$, if $a_1 : T_1, a_2 : T_2, \ldots, a_n : T_n$ then the object in (1) is of the record type in (3).

(3) $$\begin{bmatrix} l_1 & : & T_1 \\ l_2 & : & T_2 \\ \ldots & & \\ l_n & : & T_n \end{bmatrix}$$

**Updates** to these information states can be formalized as operations on these features structures, which can be simply setting of values for the fields in the simple example in (2). Feature values are also allowed to be more complex types, including stacks, lists, or other records. In this more complex case, updating the information state amounts to performing the appropriate update operation for the specified field.

In addition to task-specific aspects of the information state, such as that expressed in (2), it is very important to represent the state of the participants themselves, which is needed to interpret and participate coherently in a dialogue. There are several different dimensions to this state, which can be conveniently represented as hierarchical records and fields.

A main concern is whose information state is being represented. For dialogues with two participants, **A** and **B** there are three options: **A**'s state, **B**'s state, and an external "objective" state. When things are running smoothly, these will all tend to converge, however they may diverge in cases of un-repaired misunderstanding. Even when things are going well, there will be short-term differences in the information state, e.g., when **A** has decided what she will say but before she has said it. We take a middle ground between these three perspectives, representing an "objective hypothesis" of the information state of each participant, though not representing the participants views of the information state of the other participant. Thus, for the two-party dialogues we will be annotating, the top-level information state of the dialogue is a record with two fields, one for the information state of each participant.

Within each agent, there is also the question of how that agent views the commonality of the information: whether it is information private to the speaker, or shared between the participants. There may also be "quasi-shared" information which is accessible to all in some way, but not demonstrated or perhaps even assumed to be shared (yet).

Within each modality, there are also the individual types of information, themselves, represented variously as sets, lists, etc. Thus the kinds of information states we are looking at are generally records of the following structure:

$$(4) \quad \begin{bmatrix} \textsc{participant A} & : & \begin{bmatrix} \text{modality}_1 & : & \begin{bmatrix} \text{infotype}_1 & : & T_1 \\ \ldots & & \\ \text{infotype}_\text{k} & : & T_k \end{bmatrix} \\ \ldots & & \\ \text{modality}_\text{n} & : & \begin{bmatrix} \ldots \end{bmatrix} \end{bmatrix} \\ \textsc{participant B} & : & \begin{bmatrix} \ldots \end{bmatrix} \end{bmatrix}$$

Updates of individual aspects of the information state can be represented using the appropriate update operation and record location. E.g., for an information state of the type in (4), assuming $T_k$ is the type *stack*, then (5) would be an example update operation:

$$(5) \quad \text{popRec}(\textsc{participant A}.\text{modality}_1.\text{infotype}_\text{k})$$

More complex updates can be handled with sequences of such operations.

# 3 Two examples of annotation schemes

In this section we discuss in more detail how we have used this methodology to develop annotation schemes to study two taxonomies for classifying dialogue acts, one developed by Cooper and Larsson on the basis of work by Ginzburg [7, 13], and one developed by Poesio and Traum [18]. The discussion uses example annotations of a dialogue from the Autoroute corpus, listed in Appendix A.

### 3.1 Scheme 1: The Cooper-Larsson model of Information States

In this section we present a model of information states, using a stripped down variant of Ginzburg's [10, 11, 13] view of the dialogue game board, including questions under discussion (QUD). The development strategy has been to start as simply as possible and to add additional complexities only as they are required for representing the features of the dialogues in question. In particular, the instantiation of (4) for this information state type is shown in (6).

(6)
$$\begin{bmatrix} \text{PRIVATE} & : & \begin{bmatrix} \text{BEL} & : & \textit{Set(Prop)} \\ \text{AGENDA} & : & \textit{Stack(Action)} \end{bmatrix} \\ \text{SHARED} & : & \begin{bmatrix} \text{BEL} & : & \textit{Set(Prop)} \\ \text{QUD} & : & \textit{Stack(Question)} \end{bmatrix} \end{bmatrix}$$

That is, we made a division between PRIVATE and SHARED information. The private information consisted of a set of private beliefs (a set of propositions). Propositions are represented as English sentences with deictics referring to the dialogue participants replaced by the labels $A$ and $B$. At the level of detail we were aiming at in this analysis it did not seem relevant to commit to one particular formal semantic theory. We are more interested in the dynamic modifications to the various fields in the information state rather than the exact formal representation of the objects.

The second private field is an AGENDA which is a stack of actions which the agent is to perform. The idea here is that the agenda represents very local actions. More general goals that the agent wishes to achieve with the conversation (or her life) would, on the simple view presented here, be included in the private beliefs[2]. In contrast to goals, agenda items are actions that should in general be performed in the next move. Agenda items are introduced as a result of the previous move.

The first SHARED field in the information state is again for a set of beliefs (i.e. a set of propositions). It is something of a misnomer to call this beliefs since it is meant to represent what has been established for the sake of the conversation and we do not really mean that this necessarily represents a commitment on the part of the dialogue participants to the shared propositions. The shared beliefs represent rather what has been established as part of the conversational record, assumptions according to which the rest of the dialogue should proceed. This can, of course, be distinct from what the dialogue participants "really think".

The second SHARED field is QUD, a stack of questions under discussion. Like the agenda, this is meant to be a local affair, representing question(s) that should be addressed more or less in the next turn and not general issues that have been raised by the conversation so far or issues that the agent feels to be generally relevant.

We tried to make minimal assumptions about what actions could be put on the agenda (i.e. what actions could be performed by the dialogue participants). We characterize possible actions informally by the following inference rules, assuming that we have a type *Question* and a type *Prop(osition)*.

(7) $$\frac{q{:}\text{Question}}{\textbf{respond}(q){:}\text{Action}} \qquad \frac{q{:}\text{Question}}{\textbf{raise}(q){:}\text{Action}} \qquad \frac{p{:}\text{Prop}}{\textbf{instruct}(p){:}\text{Action}}$$

That is, dialogue participants may either raise questions (put them on QUD), respond to questions (which are maximal in QUD) or give an instruction to the other dialogue participant. We are trying here the experiment of doing as much as possible in terms of raising or responding to questions.

Transitions between information states which are occasioned by a dialogue contribution are defined in terms of a restricted set of operations. Again, this is probably more restricted than is ultimately needed, but we want to start small and then see what motivation there is for making additions. The operations we have used in this coding are given in (8).

(8) Stack: pushRec, popRec (push and pop stack in record)
Set: addRec (add to set in record)

The following example shows how updates change the information state during a dialogue. (9) shows the information state before utterance **U4** in the Autoroute dialogue presented in full in Appendix A. (10) shows the utterance itself and the accompanying updates. (11) shows the information state after the updates.

(9) $$\left[\begin{array}{lll}
\textsc{a} & = & \left[\begin{array}{lll}
\textsc{private} & = & \left[\begin{array}{lll}\textsc{agenda} & = & \left\langle\begin{array}{l}
\textbf{raise}(\text{'}\textit{Where does B want to start?}\text{'})\\
\textbf{raise}(\text{'}\textit{Where does B want to go?}\text{'})\\
\textbf{raise}(\text{'}\textit{What time does B want to go?}\text{'})\\
\textbf{raise}(\text{'}\textit{Does B want the quickest or}\\
\textit{shortest route?}\text{'})
\end{array}\right\rangle\end{array}\right]\\
\textsc{shared} & = & \left[\begin{array}{lll}\textsc{bel} & = & \left\{\begin{array}{l}
\text{'}\textit{B wants a route from A}\text{'}\\
\text{'}\textit{B has A's attention}\text{'}\\
\text{'}\textit{A has B's attention}\text{'}
\end{array}\right\}\end{array}\right]
\end{array}\right]\\
\textsc{b} & = & \left[\begin{array}{lll}
\textsc{private} & = & \left[\begin{array}{lll}\textsc{bel} & = & \left\{\text{'}\textit{B wants assistance}\text{'}\right\}\end{array}\right]\\
\textsc{shared} & = & \left[\begin{array}{lll}\textsc{bel} & = & \left\{\begin{array}{l}
\text{'}\textit{B wants a route from A}\text{'}\\
\text{'}\textit{B has A's attention}\text{'}\\
\text{'}\textit{A has B's attention}\text{'}
\end{array}\right\}\end{array}\right]
\end{array}\right]
\end{array}\right]$$

(10) **U4 [A]: Where would you like to start your journey.**

popRec(A.PRIVATE.AGENDA)
pushRec(A.SHARED.QUD, '*Where does B want to start?*')
pushRec(B.SHARED.QUD, '*Where does B want to start?*')
pushRec(B.PRIVATE.AGENDA, **respond**('*Where does B want to start?*'))

$$
(11)\quad \begin{bmatrix} \text{A} & = & \begin{bmatrix} \text{PRIVATE} & = & \begin{bmatrix} \text{AGENDA} & = & \left\langle \begin{array}{l} \textbf{raise}(\textit{'Where does B want to go?'}) \\ \textbf{raise}(\textit{'What time does B want to go?'}) \\ \textbf{raise}(\textit{'Does B want the quickest or} \\ \textit{shortest route?'}) \end{array} \right\rangle \end{bmatrix} \\ \text{SHARED} & = & \begin{bmatrix} \text{BEL} & = & \left\{ \begin{array}{l} \textit{'B wants a route from A'} \\ \textit{'B has A's attention'} \\ \textit{'A has B's attention'} \end{array} \right\} \\ \text{QUD} & = & \left\langle \textit{'Where does B want to start?'} \right\rangle \end{bmatrix} \end{bmatrix} \\ \text{B} & = & \begin{bmatrix} \text{PRIVATE} & = & \begin{bmatrix} \text{BEL} & = & \left\{ \textit{'B wants assistance'} \right\} \\ \text{AGENDA} & = & \left\langle \textbf{respond}(\textit{'Where does B want to start?'}) \right\rangle \end{bmatrix} \\ \text{SHARED} & = & \begin{bmatrix} \text{BEL} & = & \left\{ \begin{array}{l} \textit{'B wants a route from A'} \\ \textit{'B has A's attention'} \\ \textit{'A has B's attention'} \end{array} \right\} \\ \text{QUD} & = & \left\langle \textit{'Where does B want to start?'} \right\rangle \end{bmatrix} \end{bmatrix} \end{bmatrix}
$$

Note that the SHARED fields are not the same for the two dialogue participants. They may have different views about what has been established in the dialogue and what is currently under discussion. Such differences may arise because of genuine misunderstanding. But they may also arise because of the general dialogue strategy pursued by the participants which lead to mismatches which would not be intuitively construed as misunderstandings.

For a more elaborate discussion of the Cooper-Larsson information state and its relation to dialogue moves and an "optimistic" grounding strategy, see [7]. The theory has also been extended and implemented in a dialogue system, described in [3].

## 3.2 Scheme 2: The Poesio-Traum model of Information States

The second model of information states is based on the dialogue model of Poesio and Traum [17, 18]. One of the central concerns of this work, which builds upon previous work by Traum [20], is the GROUNDING process, by which common ground is established [4, 21]. Poesio and Traum view the public information state as including both the material that has already been grounded, indicated as G here, and of the material that hasn't yet been grounded; the ungrounded part consists of a specification of the current 'contributions,' or DISCOURSE UNITS (DUs), as they are called in [21].

As in the case of the notion of information state developed by Cooper and Larsson, the information state of each agent is explicitly represented in the feature-based representation. A difference, though, is the representation of individual DUs representing information introduced into the dialogue but not yet considered shared. G and each DU will be represented as a separate record within each participant's record. Also, the private information about an agent's mental state is not given a separate record, like `private` scheme 1, but represented as individual fields in the record for the participant.[3] In terms of private information, we generally represent two types. First, a list of ungrounded DUs (UDUs) which represents which of the DUs are on the way to being grounded. Secondly, the participants intentions to act related to the dialogue. This is currently represented as an ordered list of prioritised actions, as in (12)

(12) A: $<$ Ask for start place (GET-SP),
Ask for destination (GET-DEST),
Ask for start time (GET-ST),
Ask if quickest or shortest route desired (GET-ROUTE-TYPE)$>$

The record for each participant is thus of the type shown in (13).

(13)
$$\begin{bmatrix} \mathrm{G} & : & \textit{PT-record} \\ \mathrm{DU}_1 & : & \textit{PT-record} \\ \ldots & & \\ \mathrm{DU}_\mathrm{n} & : & \textit{PT-record} \\ \mathrm{UDUs} & : & \textit{List} \\ \mathrm{INT} & : & \textit{List} \end{bmatrix}$$

A second difference between the Poesio-Traum information states and that of Cooper-Larsson, described in the previous section, is the information types within the modalities. In the Poesio-Traum model there are several kinds of information kept in the shared (G) and semi-public (DUs) part of a participants information state. First, an explicit history of the dialogue acts[4] that have been performed. For simplicity, we represent that here as a list, abbreviated DH. Next we represent the social commitments, or obligations of the agents. These kinds of commitments come in two forms, depending on whether the agent is committed to a fact being the case, or to act in a particular way. We term the former SOCIAL COMMITMENTS TO A PROPOSITION, abreviated as SCP in the information state. The latter we call "Obligations", abbreviated as OBL. Also, we have a set of OPTIONS, abbreviated as OPT, representing actions which no agents have been obliged to perform, but which have been explicitly discussed as possibilities. Thus, each DU, as well as G will be a record of the type shown in (14) (abbreviated PT-record in (13)).

(14)
$$\begin{bmatrix} \mathrm{DH} & : & \textit{List} \\ \mathrm{OBL} & : & \textit{List} \\ \mathrm{SCP} & : & \textit{List} \\ \mathrm{OPT} & : & \textit{List} \end{bmatrix}$$

The obligations that are part of OBL are generally to perform a particular type of dialogue action, (e.g., 'address' or 'answer') with pointers to the relevant moves in the DH. An example is given in (15), which indicates that participant A has an obligation to answer Move 2, while participant B has obligations to answer Move 3 and to address Move 1. Obligations and commitments can also be conditional on particular actions being performed in the future.

(15) $<$ A ANSWER 2, B ANSWER 3, B ADDRESS 1 $>$

To summarize, each information state will be of the type in (16).

(16)
$$\left[\begin{array}{ll}
\text{A:} & \left[\begin{array}{ll}
\text{G:} & \left[\begin{array}{l}
\text{OBL: } < \ldots > \\
\text{DH: } < \ldots > \\
\text{SCP: } < \ldots > \\
\text{OPT: } < \ldots >
\end{array}\right] \\
\multicolumn{2}{l}{\text{INT: } < \ldots >} \\
\multicolumn{2}{l}{\text{DUi: } \ldots} \\
\multicolumn{2}{l}{\text{UDUS: } < \text{DUi}, \ldots >}
\end{array}\right] \\
\text{B:} & \left[\begin{array}{ll}
\text{G:} & \left[\begin{array}{l}
\text{OBL: } < > \\
\text{DH: } < > \\
\text{SCP: } < \ldots > \\
\text{OPT: } < \ldots >
\end{array}\right] \\
\multicolumn{2}{l}{\text{INT: } < \ldots >} \\
\multicolumn{2}{l}{\text{DUi: } \ldots} \\
\multicolumn{2}{l}{\text{UDUS: } < \text{DUi}, \ldots >}
\end{array}\right]
\end{array}\right]$$

To see how this notion of information state applies to representing the effects of utterances, consider the same utterance used to exemplify the Cooper and Larsson approach. (17) shows the information state after utterance U4. The effect of a new utterance is to create a new DU (DU4), which becomes part of the information state of both agents. The main difference between the information states of the agents in this case is that B has the intention to get a route from Malvern to Edwinstowe, whereas A has the intentions to get the information he needs to address that request.

(17) **U4 [A]: Where would you like to start your journey.**

$$\left[\begin{array}{ll}
\text{A:} & \left[\begin{array}{ll}
\text{G:} & \left[\begin{array}{l}
\text{OBL: } < \text{B } \textsc{understanding-act} \text{ 4B, A } \textsc{address} \text{ 3C} > \\
\text{SCP: } < \text{B } \textsc{wants a route} >
\end{array}\right] \\
\text{INT:} & < \begin{array}{l} \textsc{get}\text{(SP), } \textsc{get}\text{(DEST), } \textsc{get}\text{(ST), } \textsc{get}\text{(ROUTE-TYPE),} \\ \textsc{give} \text{ B } \textsc{route}\text{(SP,DEST,ST,ROUTE-TYPE)} \end{array} > \\
\text{DU4A:} & \left[\begin{array}{l}
\text{OBL: } < \text{B } \textsc{answer} \text{ 4} > \\
\text{DH: } < \text{4: } \textsc{info-request} >
\end{array}\right]
\end{array}\right] \\
\text{B:} & \left[\begin{array}{ll}
\text{G:} & \left[\begin{array}{l}
\text{OBL: } < \text{B } \textsc{understanding-act} \text{ 4B, A } \textsc{address} \text{ 3C} > \\
\text{SCP: } < \text{B } \textsc{wants a route} >
\end{array}\right] \\
\multicolumn{2}{l}{\text{INT: } < \textsc{get a route from Malvern to Edwinstowe} >} \\
\text{DU4B:} & \left[\begin{array}{l}
\text{OBL: } < \text{B } \textsc{answer} \text{ 4} > \\
\text{DH: } < \text{4: } \textsc{info-request} >
\end{array}\right]
\end{array}\right]
\end{array}\right]$$

(18) shows the information state resulting from B's answer in U5. This results in DU4 being grounded, i.e., added to G. B's obligation to answer 4 is moved to G and stays there until his action is grounded. B commits himself to the belief that the starting point is Malvern. (We only show A's info state for brevity.)

(18) **U5 [B]: Malvern.**

$$
\left[\begin{array}{ll}
\text{A:} & \left[\begin{array}{ll}
\text{G:} & \left[\begin{array}{l}
\text{OBL: } < \text{A } \textsc{understanding-act} \text{ DU5B, B } \textsc{answer} \text{ 4, A } \textsc{address} \text{ 3C} > \\
\text{SCP: } < \text{B } \textsc{wants a route} >
\end{array}\right] \\
\text{INT: } < & \begin{array}{l}
\textsc{get}(\text{SP}), \textsc{get}(\text{DEST}), \textsc{get}(\text{ST}), \textsc{get}(\text{ROUTE-TYPE}), \\
\textsc{give} \text{ B } \textsc{route}(\text{SP,DEST,ST,ROUTE-TYPE})
\end{array} > \\
\text{DU5B:} & \left[\begin{array}{l}
\text{OBL: } < \text{A } \textsc{address} \text{ 5B} > \\
\text{SCP: } < \text{B } \textsc{believes} \text{ SP} = \textsc{Malvern} > \\
\text{DH: } < \text{5A: } \textsc{answer}\text{, 5B: } \textsc{assert} >
\end{array}\right]
\end{array}\right]
\end{array}\right]
$$

As well as the updates on the individual aspects of the information state, we will also want a more complex *merger* of DU modalities, to represent the grounding process. The basic idea is that when a DU is acknowledged, all of the information from that DU is merged into G. The immediate effect will be to merge the various fields in the updated record. Other effects will involve removing some items from fields, e.g., when noticing that an obligation has been fulfilled. We will represent this merging of G with information from another DU as in (19).

(19) G+= DU

# 4 Annotation

Annotating for information states is not a particularly quick business, even given the simple information states we have in our current examples (and it requires LaTeXnical stamina). We have investigated two different ways of coping with this problem, one using an annotation tool called *TranScript* and the other using the *Thistle Diagram Editor*.

## 4.1 Annotation as scripting: TranScript

TranScript is a coding tool we have developed intended for the kind of relatively complex annotation necessary when annotating transcriptions (usually of spoken dialogue) with information state updates. In this kind of annotation, simple tags as those used in e.g. part-of-speech tagging (noun, verb etc.) are not sufficient. An information state update annotation (henceforth "update annotation") may have several arguments, i.e. the participant affected, what part of the information state is being updated (private beliefs, agenda etc.), type of update (add, push etc.), and additional arguments such as propositional content and action type. A parser and and a LaTeX generator for TranScript annotation has been implemented in SICSTUS Prolog.

The basic idea behind TranScript is that the annotation can be seen as a kind of script, which is a variation on the idea of tagging with logic programs as in TagLog [15]. The major difference is that the ordering of the annotation clauses are important. In TagLog, each clause contains a reference to a stretch of transcribed text in the transcription file. For example, in the clause `part_of_speech(34-35,noun).`, the range 34-35 refers to the word between positions 34 and 35 in the transcription. In TranScript, this reference is indicated by the ordering of clauses. Each update implicitly refers to the latest range of transcription indicated above it. For example, in the following example the updates refer to the range 157-209.

(20)
```
# range(157-209).

label(q8, "Where does B want to start?").

# update([ popRec(a*private*agenda),
           pushRec(a*shared*qud, $q8),
           pushRec(b*shared*qud, $q8),
           pushRec(b*private*agenda, respond($q8))
        ]).

# print_state.
```

TranScript contains elements of logic programming. The `$` sign indicates a label, and labels are declared with the `label` predicate as in the example above. The use of labels provides a simple way to refer to propositional contents, actions etc. in annotation clauses. A typical use of TranScript annotation files is to parse them and translate them into a sequence of information states, which then can be used to give a LaTeXversion of the transcription with information states and updates indicated as in (9)-(11).

#### 4.1.1 TranScript Commands

There are two kinds of commands in TranScript: order-independent (purely declarative) commands and order-dependent commands, or *script commands*. The declarative commands include `initial` and *label*, and the script commands are any defined operations or moves, `update`, `range`, `print_state` and `comment`. Script commands must always be preceded by the `#` symbol.

Update operations are used to actually update the previous information state to produce a new one. The `update` command takes as argument a list of updates, which consist of an update operator (or a move) and its arguments. The type of update operators available depend on the notion of information state; if the information state is a set of propositions, a typical update would be `add($p12)`, where `add` is an update operator and `$p12` is a label for a proposition (whatever that might be). The order of the updates list is important, since the information state will be updated with each operation one at a time in the order they are given. For example, if two things are to be pushed onto a stack, the order of the operations will determine the resulting stack.

(21)

| **Datatype** | **Operators** |
|---|---|
| Set | `add`, `del`(ete) |
| Stack | `push`, `pop` |
| Record | `addField`, `get_valueRec`, `set_valueRec`, `pushRec`, `popRec`, `addRec`, `delRec`, `peRec` |
| DRS | `get_valueDRS`, `set_valueDRS`, `mergeDRS` |

### 4.1.2 Parsing TranScript files

The TranScript implementation consists of two main modules: a parser and an output generator. These modules communicate via a set of instructions which can be regarded as an "inflated" version of the TranScript instructions, where information states have been filled in and labels have been replaced with their corresponding values. The parser reads the TranScript instructions and updates and keeps track of the current information state, successively updating it. The "translations" done by the default TranScript parser can be summed up in the following table:

(22)

| TranScript command | Inflated TranScript command(s) |
|---|---|
| any operator or move $U$ | `print_update(`$U_i$`)`<br>where $U_i$ is the inflated version of $U$ |
| `update(`$Us$`)` | `print_updates(`$Us_i$`); print_state(`$IS$`)`<br>where $Us_i$ is the inflated version of $Us$<br>and $IS$ is the current infostate |
| `print_state` | `print_state(`$IS$`)`<br>where $IS$ is the current infostate |
| `range(`$R$`)` | `print(`$S$`)`<br>where S us the string in range R of the transcription file |
| `print(`$S$`)` | `print(`$S$`)` |
| `comment(`$C$`)` | `print_comment(`$C$`)` |

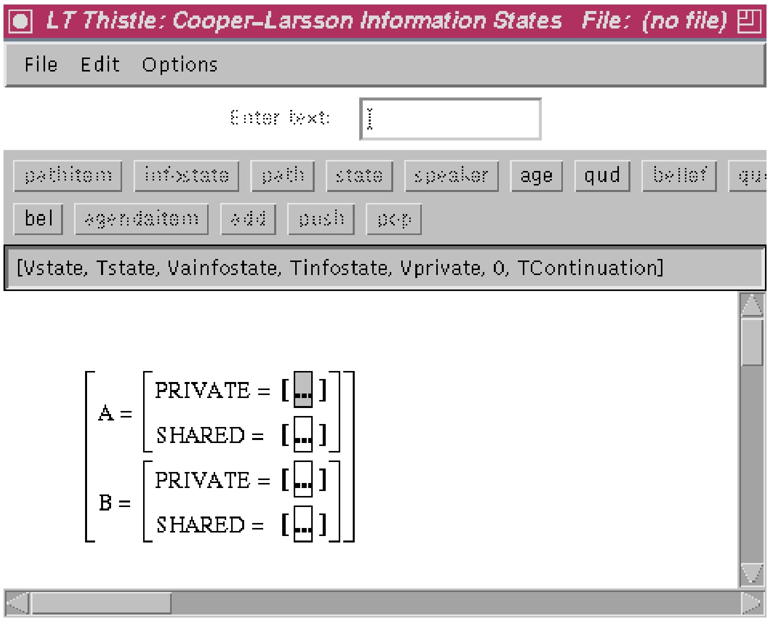
### 4.1.3 Generating output

The LaTeX˜generator takes an inflated TranScript file and produces a LaTeX˜file, following the specified list of instructions. Various conventions are used to increase readability; for example, sentences are printed in *italics*, record labels are printed in SMALL CAPS, and actions are printed in **bold** style. It is also possible to implement output generators for other kinds of output (ASCII text, HTML etc), as long as they understand the inflated TranScript instruction set.

## 4.2 Annotation Using the Thistle Diagram Editor

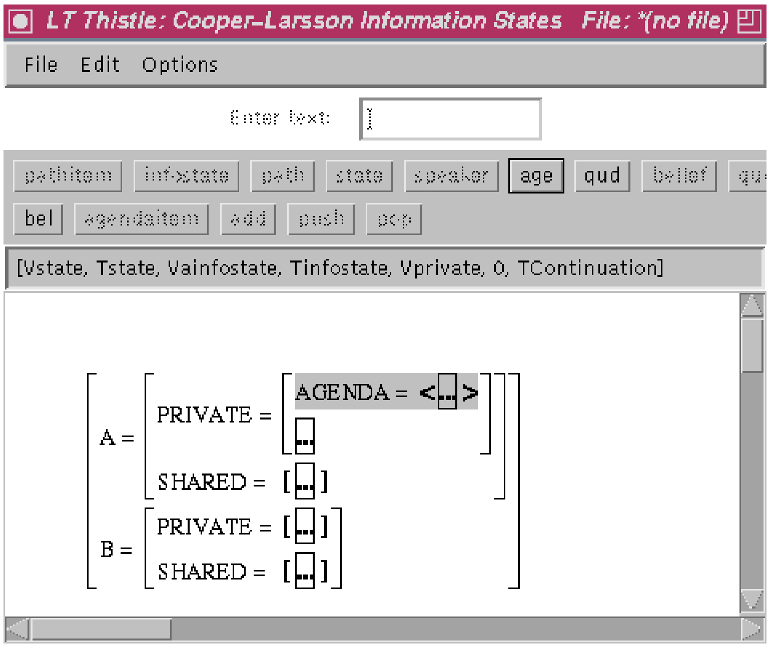
Thistle is a parameterizable display engine and editor for diagrams which allows the inclusion of interactive diagrams within Web pages. See `http://www.ltg.ed.ac.uk/software/thistle` for full information on the program and for a number of demos. We have adapted Thistle to the task of annotating ISs in dialogue.

Diagram specifications for ISs in both the Cooper-Larsson and Poesio-Traum models have been written, allowing the structures to be displayed and edited. In order to illustrate this, the diagram below shows an 'empty' Cooper-Larsson IS in Thistle:



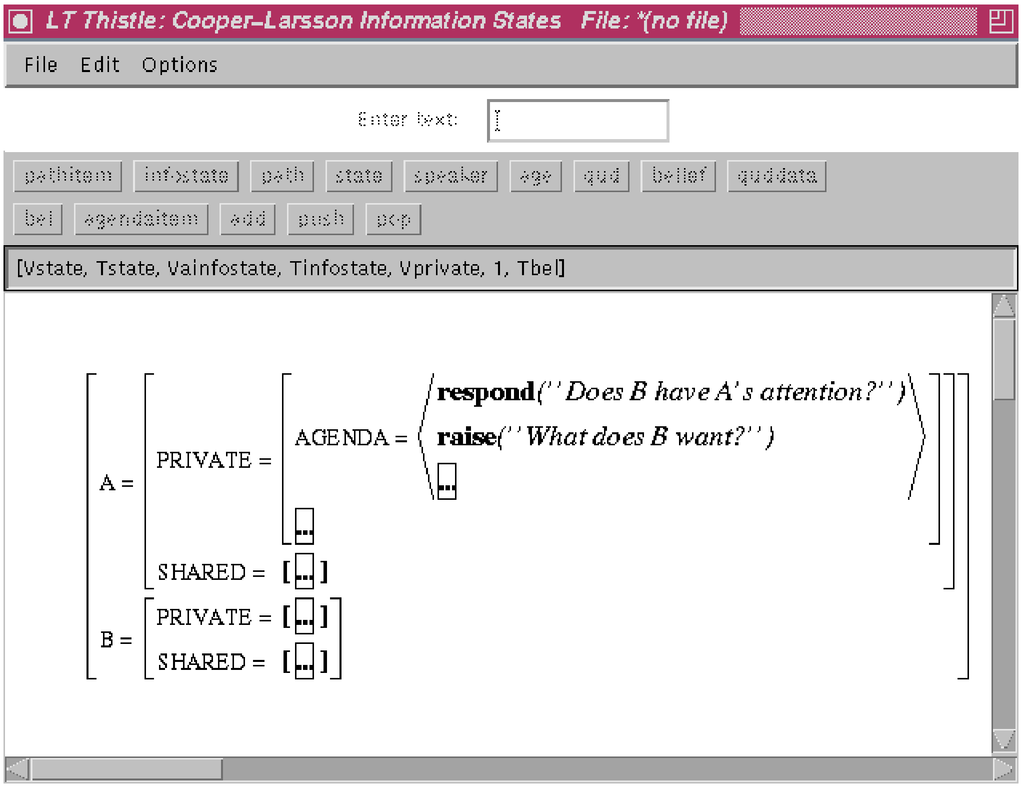

(23)

This diagram class is designed as an Attribute-Value Matrix, a standard way of representing linguistic (and other) information. Here three dots inside a box indicate areas where a diagram can be expanded, and in this case the user has clicked on the value of the path A=PRIVATE, with the result that the box is shaded and the possible types which can expand the diagram are enabled in the menu bar at the top half of the window. The options (defined by the theory, of course), are *age* (to expand the AGENDA field), *qud* (expand QUESTION UNDER DISCUSSION), and *bel* (expand BELIEFS). If the user selects *age*, the result is:




(24)

The diagram is expanded appropriately, and if the user now clicks in the expansion box in the value of AGENDA, selects *agendaitem* (the only choice) and adds the appropriate text, the result is:

(25) 


The appearance of the text (the bold and italic fonts and the small capitals for the attribute names) are all handled automatically by the diagram class specification. Similarly, the precise layout of the AVM is done by the program – all the user has to do is supply the appropriate paths and values.

It is thus possible to use Thistle to produce annotations in the form of ISs, and including a representation of the updates in the diagram class is a simple matter. However, doing large-scale annotation in this way is clearly time-consuming, and it is likely that this method would only be useful to produce small illustrations.

# 5 Discussion

Our pilot work (reported in [8]) involved using the annotation schemes discussed above to study the dialogue in Appendix A; this involved several annotations and subsequent revisions. In this section we are going to discuss our preliminary conclusions about the methodology and some empirical issues raised by this work.

We do feel that the methodology we are developing could be useful both (i) for people who are interested in studying dialogue acts either from an empirical perspective or by looking in more detail at the formal differences between systems; and (ii) for people who are building systems, who could just come up with a characterization of their information states without worrying about formal details, a characterization of the updates each dialogue act performs, and then use the tool to check that their definitions of dialogue acts behave as intended.

There are however some potential problems to be considered. First of all, since the notation does not wear its semantics on its sleeves, more detailed comparisons between theories will involve either more detailed annotations, or spelling out the interpretation of primitives such as intentions and obligations, or both. For example, we have been investigating the differences between a model based on obligations and a model based on questions under discussions; but such differences cannot be revealed as long as the only constraint we impose on the fields is that their values are stacks.

Secondly, it has become even clearer to us that annotating for information states is not suitable for large-scale annotation: both because it is time consuming, and because it is even more difficult to agree on the composition of the information state of an agent than it is to agree on which dialogue act has been performed. It definitely seems to be the case that this type of annotation should be used in the preliminary phases of an annotation work, to come up with a taxonomy of dialogue acts that appears to have adequate coverage and matches the operations that the system has to perform; subsequent, large scale annotation can then be done in terms of atomic labels.

For future work, we plan to integrate the annotation tools into the TRINDI Dialogue Move Engine toolkit (see [19], [16]). This will make it possible to define dialogue moves and information state update rules in terms of update operations, and to annotate using these moves and rules. The toolkit will also enable visualization (and possibly editing) of information states using Thistle.

## Acknowledgments

This work was supported by the TRINDI (Task Oriented Instructional Dialogue) project, EU TELEMATICS APPLICATIONS Programme, Language Engineering Project LE4-8314. We are grateful to the Speech Research Unit of the Defence Evaluation and Research Agency, Malvern, UK, for making the Autoroute dialogues available to the Trindi project.

MASSIMO POESIO, COLIN MATHESON
*Human Communication Research Centre*
*University of Edinburgh*
*2 Buccleuch Place Edinburgh EH8 9LW, Scotland, UK*

ROBIN COOPER, STAFFAN LARSSON
*Department of Linguistics*
*Göteborg University*
*Box 200, SE-405 30 Göteborg, Sweden*

DAVID R TRAUM
*UMIACS*
*A. V. Williams Building*
*University of Maryland*
*College Park, MD 20742 USA*

# Notes

[1]Additional constraints can also be imposed by the user - e.g., minimizing time, or toll cost, etc. We will ignore these constraints here.

[2]This is most likely an oversimplification; it will probably be necessary to have a separate field for goals.

[3]We do this for two reasons. First, just to avoid the need for an extra record indirection when coding, and secondly, to be closer to the DRT-based theory in [17, 18], which relied on DRT accessibility relations. For the purposes of this record-based model of information state, there is nothing wrong with viewing these other aspects of the mental state as belonging to a subrecord for the modality `private`, so as to conform to the specification in (4).

[4]In the Poesio-Traum model, the DRI dialogue acts ([2], [9]) are used.

# References


[1] D. Albesano, P. Baggia, M. Danieli, R. Gemello, E. Gerbino, and C. Rullent. A robust system for human-machine dialogue in a telephony-based application. *Journal of Speech Technology*, 2(2):99–110, 1997.

[2] J. Allen and M. Core. DAMSL: Dialogue act markup in several layers. Draft contribution for the Discourse Resource Initiative, October 1997.

[3] P. Bohlin, R. Cooper, E. Engdahl, and S. Larsson. Information states and dialogue move engines. In J. Alexandersson, editor, *IJCAI-99 Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, 1999.

[4] H. H. Clark and E. F. Schaefer. Contributing to discourse. *Cognitive Science*, 13:259 – 94, 1989.

[5] R. Cooper. Information states, attitudes and dependent record types. In *Proceedings of ITALLC-98*, 1998.

[6] R. Cooper. Mixing situation theory and type theory to formalize information states in dialogue exchanges. In *Proceedings of TWLT13/Twendial '98: Formal Semantics and Pragmatics of Dialogue*, 1998.

[7] R. Cooper and S. Larsson. Dialogue moves and information states. In *Proc. of the Third IWCS*, Tilburg, 1999.

[8] R. Cooper, S. Larsson, C. Matheson, M. Poesio, and D. Traum. Coding instructional dialogue for information states. deliverable D1.1, TRINDI, 1999.

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

[10] J. Ginzburg. Resolving questions, i. *Linguistics and Philosophy*, 18(5):567–609, 1995.

[11] J. Ginzburg. Resolving questions, ii. *Linguistics and Philosophy*, 18(6):567–609, 1995.

[12] J. Ginzburg. Dynamics and the semantics of dialogue. In Jerry Seligman and Dag Westerståhl, editors, *Logic, Language and Computation, Vol. 1*, volume 1. CSLI Publications, 1996.

[13] J. Ginzburg. Clarifying utterances. In J. Hulstijn and A. Niholt, editors, *Proc. of the Twente Workshop on the Formal Semantics and Pragmatics of Dialogues*, pages 11–30, Enschede, 1998. Universiteit Twente, Faculteit Informatica.

[14] B. J. Grosz and C. L. Sidner. Attention, intention, and the structure of discourse. *Computational Linguistics*, 12(3):175–204, 1986.

[15] T. Lager. *A Logical Approach to Computational Corpus Linguistics.* PhD thesis, Detp. of Linguistics, Göteborg Universtity, 1995.

[16] S. Larsson, P. Bohlin, J. Bos, and D. Traum. Coding instructional dialogue for information states. deliverable D2.2, TRINDI, forthcoming.

[17] M. Poesio and D. Traum. Conversational actions and discourse situations. *Computational Intelligence*, 13(3):309–347, 1997.

[18] M. Poesio and D. Traum. Towards an axiomatisation of dialogue acts. In J. Hulstijn and A. Nijholt, editors, *Proc. of the Twente Workshop on the Formal Semantics and Pragmatics of Dialogues*, pages 207–222, Enschede, 1998. Universiteit Twente, Faculteit Informatica.

[19] D. Traum, J. Bos, R. Cooper, S. Larsson, I. Lewin, C. Matheson, and M. Poesio. Coding instructional dialogue for information states. deliverable D2.1, TRINDI, forthcoming.

[20] D. R. Traum. *A Computational Theory of Grounding in Natural Language Conversation.* PhD thesis, University of Rochester, Department of Computer Science, Rochester, NY, July 1994.

[21] D. R. Traum and E. A. Hinkelman. Conversation acts in task-oriented spoken dialogue. *Computational Intelligence*, 8(3), 1992. Special Issue on Non-literal Language.

# A Autoroute dialogue 127

**U1 [A]: Welcome to the Route Planning Service.**
**U2 [A]: How can I help you.**
**U3 [B]: A route please.**
**U4 [A]: Where would you like to start your journey.**
**U5 [B]: Malvern.**
**U6 [A]: Starting in Great Malvern.**
**U7 [B]: Yes.**
**U8 [A]: Where would you like to go.**
**U9 [B]: Edwinstowe.**
**U10 [A]: Edwinstowe.**
**U11 [B]: Yes.**
**U12 [A]: Please wait.**
**U13 [A]: Is that Edwinstowe in Nottingham.**
**U14 [B]: Yes.**
**U15 [A]: What time would you like to make your journey.**
**U16 [B]: Six p.m.**
**U17 [A]: Leaving at six p.m.**
**U18 [B]: Yes.**
**U19 [A]: Would you like the quickest or the shortest route.**
**U20 [B]: Quickest.**
**U21 [A]: Please wait while your route from Malvern to Edwinstowe is calculated.**
**U22 [A]: The quickest route is one hundred and thirteen miles and will take two hours eight minutes.**
**U23 [A]: Would you like me to send the instructions to you.**
**U24 [B]: No.**

**U25 [A]: Do you require any further information now.**
**U26 [B]: No.**
**U27 [A]: Can I have your name please.**
**U28 [B]: Mr Smith**
**U29 [A]: Mr Smith**
**U30 [B]: Yes**
**U31 [A]: And your location please.**
**U32 [B]: T 43**
**U33 [A]: T 43**
**U34 [B]: Yes.**
**U35 [A]: OK.**
**U36 [A]: Thank you for calling.**
**U37 [A]: Goodbye.**