Coding Instructional Dialogue for Information States

R. Cooper, S. Larsson, C. Matheson, M. Poesio, D. Traum

Distribution: PUBLIC

Task Oriented Instructional Dialogue
LE4-8314

Deliverable D1.1

February 1999

LE4-8314 TRINDI

Task Oriented Instructional Dialogue

Gothenburg University
Department of Linguistics

University of Edinburgh
Centre for Cognitive Science and Language Technology Group, Human Communication Research Centre

Universität des Saarlandes
Department of Computational Linguistics

SRI Cambridge

Xerox Research Centre Europe

For copies of reports, updates on project activities and other TRINDI-related information, contact:

The TRINDI Project Administrator
Department of Linguistics
Göteborg University
Box 200
S-405 30 Gothenburg, Sweden
trindi@ling.gu.se

Copies of reports and other material can also be accessed from the project's homepage, http://www.ling.gu.se/research/projects/trindi.

Contents

Chapter 1

Introduction

Probably the central issue in analysis of dialogues is the joint 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* Grosz and Sidner (1986) of how current topics relate to overall objectives, or in terms of the questions under discussion Ginzburg (1996), 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.

In this document, we begin to explore these issues by developing coding schemes for information states. We begin, in Chapter 2, with an exploration of dialogue moves, as is often done in the development of dialogue systems. Researchers often develop a taxonomy of dialogue moves and then use it to analyze the transcripts of conversations in the chosen domain (possibly by doing some large-scale annotation when the intention is to compute some statistics or train the dialogue manager). At the moment there are no serious alternatives to this

development methodology, which suffers however from two serious problems:

1. This type of annotation essentially amounts to guessing the mental state of the participants in the dialogue; while this is possible to some extent, there are always bound to be subjective components in this type of analysis.

2. The taxonomy of moves is also generally specified in fairly loose terms; this means that even when it's clear what an agent tried to do at a given point in a conversation, deciding on the move classification of that utterance is also in part a subjective matter.

   This problem already occurs when trying to analyze conversations within a single domain, but it becomes especially acute when trying to compare taxonomies developed for different domains. Each system adopts a slightly different notion of information state; therefore, different types of transitions are generally assumed. Yet, the same labels tend to be used, which means that comparisons between domains tend to be difficult.

There isn't much that we can do concerning the first problem. We think, however, that there is a way of addressing the second problem - namely, trying to specify the interpretation of moves in terms of simpler primitives relating to the information state that might be at least in part common across domains. Researchers could then either annotate their corpus directly in terms of information states, or at least specify the intended interpretation of the moves in their taxonomy in terms of these primitives.

In Chapter 3, we present an approach to defining information states, and a specific instantiation of two different theories of information state in dialogue. The first is based loosely on ideas from Jonathan Ginzburg (Ginzburg, 1995a,b, 1997, 1998), in which the main notions of information state are a structure of questions under discussion, which, along with agendas which include items of raising or resolving these questions, drives the production of utterances, eventually yielding relevant shared beliefs. The second theory attempts to more model more aspects of the motivations and effects of utterances, using the framework developed by Poesio and Traum (Traum, 1994; Poesio and Traum, 1997, 1998).

In Chapter 4, we continue with an analysis of the relation of these notions of information state to dialogue moves, as described in Chapter 2. This includes examples of two different approaches to defining the relations between these concepts. First, we show how with the first scheme, sets of simple operations on these information states can be used to characterize the dialogue moves from the HCRC coding scheme, described in Chapter 2 (and more fully in (Carletta *et al.*, 1997)). We also show how dialogue moves can be directly incorporated as part of the notion of information state, using the second information state scheme, and a move set also described in Chapter 2, incorporating aspects of those presented in (Traum and Hinkelman, 1992) and (Discourse Resource Initiative, 1997; Allen and Core, 1997).

In Chapter 5, we turn to the issue of coding information states, proper, including additional information needed to perform updates in each theory, and some annotated examples of the principles needed for coding. These examples are drawn from the dialogue in Appendix A.

In Chapter 6, we present some coding tools for facilitating the kind of coding described in Chapter 5. These tools include a prolog-based scripting language which takes in updates and produces reports including the updated state. Another tool uses the GATE system Gaizauskas (1998) as a front end. These tools were used to produce more complete annotations of the Dialogue in Appendix A. Appendix B includes a complete coding of the dialogue in Scheme 1, using the GATE tools. Appendix C shows the full information state and updates for Scheme 2 for the first seven utterances of that dialogue.

Finally, in Chapter 7, we conclude with some observations on coding information states and prospects for future work.

Chapter 2

Classifications of Moves

In this chapter we briefly review a few classification schemes for moves.

2.1 Comparison and integration of five classification schemes for moves

Among the classification schemes for moves which have been proposed are the following:

- The HCRC (Human Communication Research Centre) have developed a scheme with three complementary structural levels (move, game and transaction) for coding dialogue structure in the MapTask corpus. Reliability has been measured using the Kappa statistic (Carletta *et al.*, 1997) indicating various levels of agreement for different schemes, but generally good.
- The DRI scheme is the product of the Discourse Resource Initiative, consisting of researchers from several dialogue projects worldwide. The goal of DRI is to provide a standard for coding of dialogue acts, which if necessary can be augmented with further subdivisions of the given categories. Detailed Kappa statistics are given indicating various levels of success.
- Linköping University have coded a corpus of WOZ-dialogues using two different schemes (Ahrenberg *et al.*, 1995) - one very simple (henceforth referred to as LINLIN1) and one slightly more complex (LINLIN2). Reliability measures are given for the simple scheme, though not in terms of Kappa.
- In connection with the TRAINS project, Traum (Traum and Hinkelman, 1992) has developed a taxonomy of Conversation Act types, consisting of Turn-taking acts, Grounding acts, Core Speech Acts[1] and Argumentation acts. In this scheme, each conversation

[1] In recent publications, e.g. (Poesio and Traum, 1997), Poesio & Traum have replaced this level with the

act type corresponds to a Discourse Level (structural level) described in terms of *Discourse Units* (DUs) or Utterance Units (UUs). A DU are a bit like a MapTask game except that while a game usually ends when then goal of the initiating move has been fulfilled (e.g. a question has been answered), a DU ends when the initiating utterance has been mutually understood, or *Grounded*. Utterance Units correspond to more or less continuous speech by one speaker, punctuated by prosodic boundaries. Turn-taking acts, grounding acts, core speech acts and argumentation acts correspond to the Discourse Levels Sub-UU, UU, DU, and Multiple DUs, respectively. No reliability statistics are given.

- A parameterized account of communicative acts in communication management is given in (Allwood *et al.*, 1994). The parameters (or "dimensions") are expressive and evocative function. This scheme will henceforth be referred to as the GBG-IM scheme.)

In this section, we will attempt to establish some relevant parameters of variation in coding scheme design, and find corresponding parameter values for the LINLIN, MapTask, DRI, TRAINS and Göteborg schemes. When designing a coding scheme for dialogue moves, there are several choices that can be made:

- What range of phenomena is covered?
- How are these phenomena divided into different layers?
- How is each layer divided into categories, subcategories and so on
- What are the principles of (utterance) segmentation
- Can utterances have several functions?
- Is the scheme domain dependent?
- Is the scheme dependent on dialogue genre?
- Is the scheme theory-dependent?
- What kind of definitions are used? Intentional, surface-based or other?

Along all these dimensions we find variation between different schemes, and the choices along different dimensions are to various degrees dependent of each other. Regarding all these, we need to consider various tradeoffs between cognitive plausibility, ease of coding, reliability and computational tractability. Of course, it is not necessary to make all these choices in designing a scheme; some issues may simply be left open. This is especially true in the case of general coding schemes such as DRI.

These choices influence, in various ways, the reliability of a scheme and the potential for computational tractability of a model of dialog based on that scheme. A simple scheme

Forward Looking Function level of the DRI scheme.

will most likely make it easier both to achieve and assess its reliability. It might also lead to a more tractable computational model. However, a simple scheme may also produce an over-simplistic and unnatural model of dialogue. For example, the MapTask scheme does not permit utterances to be coded for more than one move. Also, it does not permit coding of relations, e.g. one cannot annotate which question an answer is an answer to. These limitations may make the scheme less expressive and less realistic, but it also seems to make it more reliable.

There are signs that the DRI scheme is being accepted as a standard of dialogue move coding. As noted, DRI uses intention-based rather than surface-based definitions, which is probably a better approach for most purposes. The disadvantages of DRI is the fact that it is still under development and has not yet been extensively used or tested in coding actual dialogues, while the LINLIN and MapTask schemes have the advantage of having actually been used, and they also have higher reliability rates than DRI. The latter fact can probably in part be explained by the smaller number of categories in the LINLIN and MapTask schemes.

2.1.1 Scope and layers

There are clear differences in the scope of phenomena and division of these phenomena into layers between the schemes we have considered above. A rough impression of these differences are given in figure 2.1.

LINLIN2	**MapTask**	**DRI**	**TRAINS**	**GBG-IM**
—	Game		Argumentation acts	
Type	Move	Forward-Looking, Backward-Looking	Core Speech Acts	
		Signal-Understanding	Grounding Acts	Feedback function
—	—	—	Turn-taking	Turn characteristics
Discourse management	—	Conventional	—	—
Topic	—	Information-level	—	—
—	—	Communicative Status	—	—

Table 2.1: Rough impression of relations between scheme layers

The transaction and game levels of the MapTask scheme seem to have no corresponding layers in the LINLIN and DRI schemes[2]. Likewise, the Communicative-status layer of DRI has no obvious counterpart in the other schemes. However, the Topic layer of LINLIN is very similar to the Information Level layer of DRI, in that they both try to capture some general semantics of utterances in terms of what they are about.

[2]The Argumentation Acts of the TRAINS scheme seems to have largely the same coverage as these levels

2.1.2 Move taxonomies

A somewhat speculative characterization of relation between scheme categories can be found in Tables 2.2 and 2.3. Italics indicate a category-set.

LINLIN2	**MapTask**	**DRI**	**TRAINS**	**GBG-IM**
Initiative	*Initiating moves*	*Forward Looking Function*	*Core speech acts*	—
Update	Explain	*Statement* Assert Reassert Other	Inform	
Question	Query-yn Query-w Check Align	Info-request	YNQ WHQ	
—	Instruct	*Influencing-addressee-future-action* Action-directive Open-Option	 Request Suggest	
	—	*Committing-speaker future-action* Offer Commit	 Offer	
		Explicit-performative	Promise	
		Exclamation	—	
	Response moves	*Backward Looking Function*	*Core speech acts*	—
Response (Answer)	Reply-y, Reply-n, Reply-w, Clarify	Answer	Eval	
	—	*Agreement* Accept Accept-part Maybe Reject Reject-part Hold	 Accept Reject	 +Accept-content −Accept-content
—	Ready ???	—	—	—

Table 2.2: Rough impression of relations between move taxonomies, pt. 1

There are clearly similarities between the schemes, e.g. the top level division into initiatives and responses. The DRI scheme is generally the most complex, but in some cases the MapTask scheme gives a more fine-grained analysis. The TRAINS and GBG-IM schemes include some aspects of feedback and turntaking not covered by the other schemes. To make this kind of comparisons more exact, however, there is a need for finding a way of giving exact semantics for coding schemes. This is clearly a subject worth further studies.

LINLIN2	**MapTask**	**DRI**	**TRAINS**	**GBG-IM**
—	*(Response moves)*	*Understanding*	*Grounding*	*Feedback function*
	—	—	ReqAck	Elicit FB
			ReqRepair	
	(Acknowledge)			+Accept-com-act
	—	—	—	−Accept-com-act
	(Acknowledge)	*Signal-understanding* Acknowledge Repeat-rephrase Completion	Ack	+Understanding
	—	Signal-Non-Understanding	—	−Understanding
	Acknowledge	—	—	+Perception
	—	(Signal-non-und.)	—	−Perception
	—	—	—	+Contact
	—	(Signal-non-und.)	—	−Contact
	—	Correct-Misspeaking	Repair	
	—	—	Initiate	—
			Continue	
			Cancel	
—	—	—	*Turn-taking*	*Turn Management*
			take-turn	Turn acceptance
			keep-turn	Turn holding
			release-turn	Turn closing
			assign-turn	—
Discourse management	—	*Conventional*	—	—
Opening		Opening		
Continuation				
Closing		Closing		

Table 2.3: Rough impression of relations between move taxonomies, pt. 2

2.1.3 Dependencies on dialogue genre, domain and theory

The five coding schemes described above have all been designed for different genres of dialogue, different domains and based on different theories. A rough overview of these differences are given in Table 2.4.

	Dialogue genre	**Theory**	**Domain**
LINLIN	information retrieval	dialogue grammar	various
MapTask	instructional	dialogue games	route following
DRI	general	speech acts	general
TRAINS	interactive planning	conversation acts	route planning
GBG-IM	general	activity-based pragmatics	general

Table 2.4: Dialogue genre, intended domain and foundational theory for the three schemes described above.

2.1.4 Integration of schemes

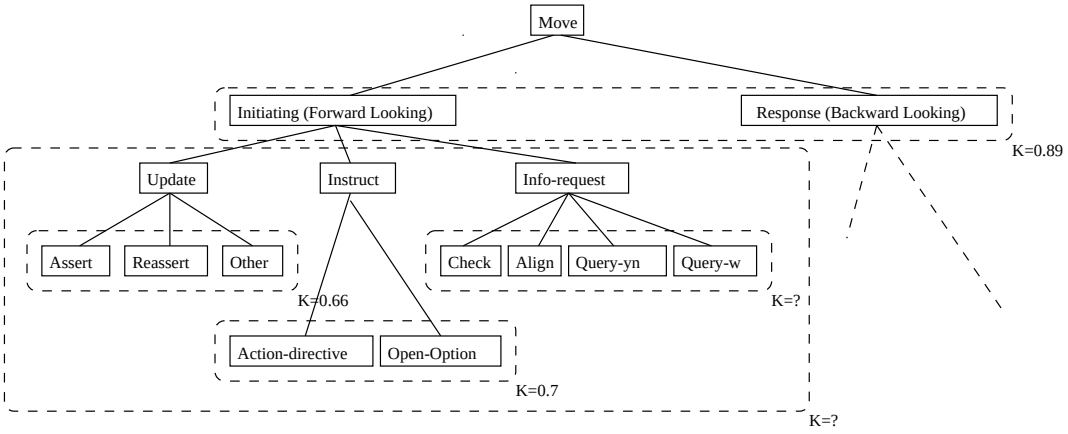


Figure 2.1: Part of hypothetical scheme formed by uniting the DRI and MapTask schemes, with phony Kappa values for groups at different levels

Tables 2.2 and 2.3 suggests that, if we view coding schemes as type hierarchies for dialogue moves, we can embed complex (parts of) schemes in simpler schemes. For example, we can produce a maximally complex scheme by extending the DRI scheme with parts of the MapTask scheme (see Figure 2.1), and code dialogues using this maximally complex scheme. This, of course, requires among other things that the category definitions of the different schemes are made compatible, and preferably also more precise. We can then compute reliability at any desired level (or combination of levels) in the hierarchy. Comparing Kappa statistics obtained for the three coding schemes indicate that we can expect higher reliability for less complex schemes. When analyzing coded dialogues, we may choose to collapse some distinctions (e.g.

seeing checks, aligns and queries as info-requests) if we want to. We may also choose to allow for coding of "non-leaf" categories (e.g. Initiative and Response). A hypertext version of a possible scheme encompassing all five of the above-mentioned schemes is available at `http://www.ling.gu.se/ sl/sdime_type.html`.

2.2 A Modified Version of the DRI Scheme

In this section we discuss the move classification scheme used in (Poesio and Traum, 1998).

Most classic theories of speech acts concentrate on the actions performed by the conversational participants as a way of 'getting the job done'—e.g., instructions to the other conversant, requests for information necessary to accomplish the task, etc. But these actions are only a part of what happens in conversations; the participants in a conversation spend a lot of their time making sure they do not talk over each other and ensuring that 'informational' coordination is achieved. Recent theories of speech acts (e.g., Novick (1988); Kowtko *et al.* (1992); Traum (1994); Bunt (1995)) are built on the assumption that a good theory of the actions involved in these aspects of a conversation is as important to a dialogue system as a good theory of task-oriented acts.

The multi-level CONVERSATION ACTS theory, presented in Traum and Hinkelman (1992), maintains the classical illocutionary acts of speech act theory (e.g., **inform**, **request**), now called CORE SPEECH ACTS. These actions are, however, reinterpreted as multi-agent collaborative achievements, taking on their full effect only after they have been *grounded*, i.e., acknowledged. Rather than being actions performed by a speaker to a hearer, the core speech acts are joint actions; the initial speaker and the hearer (called hereafter INITIATOR and RESPONDER, respectively) each contribute actions of a more basic type, the result being the common ground assumed to be the effects of core speech acts.

In addition, Conversation Acts (CA) theory also assumes that three other kinds of speech acts are performed in conversations: acts for TURN-TAKING, GROUNDING, and more complex acts called ARGUMENTATION ACTS; Traum and Hinkelman include in this class both the macro structures of conversation often called GAMES (Carlson, 1983; Levin and Moore, 1978) and the organization of acts according to the rhetorical structure of discourse, as in elaborations. In (Poesio and Traum, 1997) the additional level of LOCUTIONARY ACTS was made explicit in addition to the four levels of the initial proposal. We will not be concerned with turn-taking and argumentation acts here.

The dialogue acts adopted in (Poesio and Traum, 1998) are those proposed in the Discourse Resource Initiative (Discourse Resource Initiative, 1997; Allen and Core, 1997), currently the most widely examined proposal for a task-independent set of dialogue acts.[3] The DRI scheme has a somewhat different conceptual organization from CA theory, but it is relatively easy to

[3]It is being developed by an international team of dialogue researchers from previous coding schemes and speech act taxonomies, and the reliability of this classification scheme has been studied in (Core and Allen, 1997; Di Eugenio *et al.*, 1997).

establish a connection.

Poesio and Traum assume that speech acts are just ordinary events, for which they adopt a Davidsonian treatment (Davidson, 1967) as usual in Discourse Representation Theory Kamp and Reyle (1993)–more specifically, the version of Davidson's theory proposed by Muskens (1995), in which eventualities are objects of type ϵ and each predicate has an extra argument for the eventuality. They adopt however the standard DRT notation, and write $e : \mathbf{p}(\overline{x})$ for $\mathbf{p}(\overline{x},e)$. Each eventuality e is associated in Muskens (1995) with a unique time interval $\vartheta(e)$; time intervals are mostly omitted below, except where necessary to specify the updates.

2.2.1 Locutionary Acts

Poesio and Traum use the ternary predicate $e : \mathbf{Utter}(A,P)$ to characterize locutionary acts, where A is an individual, P is a string, and, as mentioned above, e is an eventuality. A locutionary act may consist of an utterance of a single word, a sentence constituent such as an NP, or a complete sentence.[4]

2.2.2 Core Speech Acts

Core speech acts are dialogue acts which have to do with managing the topic of the conversation, in a general sense. Some of them play a FORWARD-LOOKING FUNCTION: they introduce new social attitudes in the conversation that have to be addressed. The forward-looking acts from the DRI dialogue act coding scheme are shown in (2.1).

(2.1)
- **Statement**
  - **Assert**
  - **Reassert**
  - **Other-statement**
- **Influencing-addressee-future-action**
  - **Open-option**
  - **Directive**
    - **Action-directive**
    - **Info-request**
- **Committing-speaker-future-action**
  - **Offer**
  - **Commit**
- **Conventional**

[4]The participants in a conversation are also assumed to share additional information about a locutionary act such as its syntactic classification if any or its meaning. See (Poesio and Traum, 1997; Poesio, 1998) for details.

  - **Opening**
  - **Closing**
- **Explicit-performative**
- **Exclamation**

In this scheme acts are hierarchically organized in classes and subclasses; sub-acts maintain all of the properties of the parent act, while also adding additional information about the act.[5] The current scheme specifies six main act types with subtypes. An initiator is committed to the veracity of her **Statement**. If the statement is used to try to achieve the belief of the addressee (regardless of its success, or the prior belief of the addressee), then it is an **Assert**. If the initiator was already previously so committed, then it is a **Reassert**. An **Other-statement** is a statement that is not an assert or reassert, such as taking a stand on a particular position, without concern to the beliefs of other conversants on this matter. The decision as to whether to classify an utterance for the **Statement** dimension, and if so which class to use, is guided by the decision tree for statements as shown in Fig. 2.2.

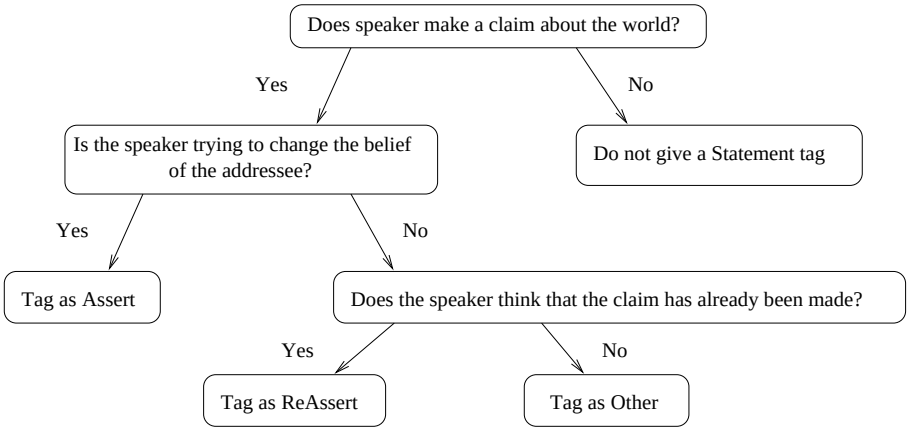


Figure 2.2: The DRI decision tree for statements

Influencing-addressee-future-action acts constrain the discourse situation to contain an option for the addressee. **Open-option** does only this; more precisely, it does not count as an attempt to get the addressee to actually do the mentioned act, merely allows it as a possibility for consideration. A **directive**, on the other hand, *does* count as such an attempt. The DRI scheme includes two types of directives to the other agent, depending on what kind of action is directed. **Info-requests** are directives to perform a statement.**Action-directives** are directives to perform another kind of action. Both types of directives also impose an obligation to address the directive itself (though not necessarily to perform the requested action) Traum and Allen (1994). A **Committing-speaker-future-action** act mentions

[5]The scheme as developed in Discourse Resource Initiative (1997) included **Info-request** as a sub-class of **Action-directive** — it is a directive in which the directed act is one of making a statement. The authors of Allen and Core (1997), subsequently decided to make **Info-request** its own main type, since it was often easy to identify using a different syntactic form than other directives (i.e., interrogative vs. imperative mood). While this change is sensible for a coding manual, for reasons of semantic simplicity, we stick with the prior formulation in this paper.

an option of the initiator. A **Commit** act means that the initiator has an obligation to perform the action. An **Offer** is a conditional commitment: if the addressee accepts, then the initiator is committed. **Explicit-performatives** are the traditional speech acts from (Austin, 1962). The DRI scheme also includes the acts **Opening** and **Closing**, which have to do with the conventional organization of conversations. We will not discuss explicit performatives, conventional acts, and exclamations here. We refer the reader to (Allen and Core, 1997) for more discussion and examples of these classes, as well as their decision trees.

Other core speech acts are instead classified in the DRI scheme as responses to previous acts: for example, the initiator may accept or reject a previous proposal, or answer a request for information. These acts are called BACKWARD-LOOKING in the DRI classification. The backward-looking acts from the DRI scheme playing a function related to the task are listed below; as we will see, other backward-looking acts play functions related to grounding. The specification of such acts always involves mention of the dialogue act(s) that they are a response to; i.e., all of these acts are implicitly anaphoric on previous speech acts. The decision tree for backward-looking acts is shown in Fig. 2.3; again we refer the reader to (Discourse Resource Initiative, 1997; Allen and Core, 1997) for discussion and examples.

- **Agreement**
  - **Accept**
  - **Accept-part**
  - **Maybe**
  - **Reject**
  - **Reject-part**
  - **Hold**
- **Answer**

Hold is the label used for any actions that do not explicitly accept or reject the act they are a response to, but merely postpone the decision.

2.2.3 Grounding Acts

The model proposed in (Poesio and Traum, 1998) inherits a fundamental assumption of theories such as (Clark and Schaefer, 1989; Traum, 1994): that information has to be GROUNDED before it becomes part of the common ground. As in (Traum, 1994), they assume that grounding is achieved by means of dialogue acts. Acts such as assertions or instructions specify CONTRIBUTIONS that have to be ACKNOWLEDGED before they become a proper part of the common ground. Acknowledgments can either be performed implicitly or explicitly, by means of linguistics expressions such as *okay* or *gotcha* but also by nodding or by means of expressions such as *uhu*. Here is an example of acknowledgment from the TRAINS-93 corpus:

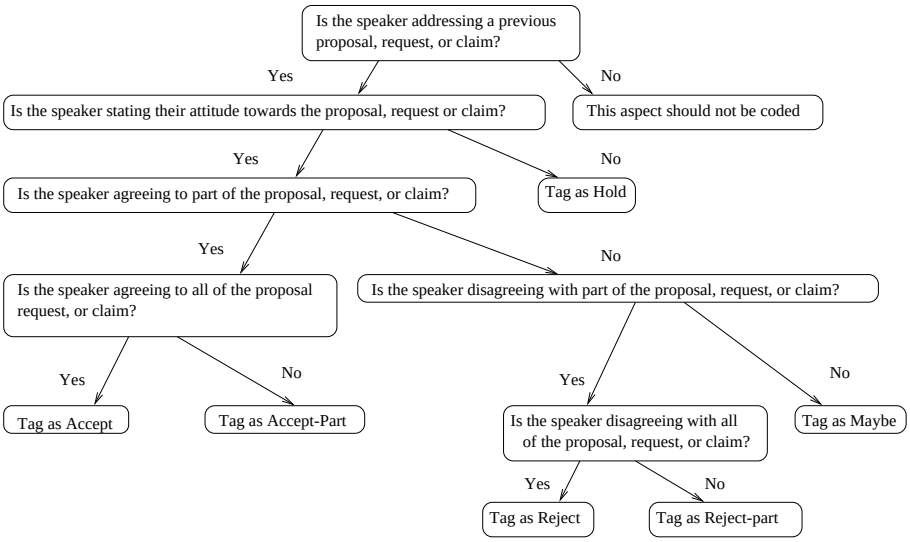


Figure 2.3: Backward-looking acts in the DRI scheme

(2.2) `utt1: s: take the Avon train to Dansville`
`utt2: u: Okay`

The participants in a conversation do not always acknowledge contributions right away: they may also signal that they did not understand, e.g., by saying *Sorry, I didn't hear that.*

Some of the backward-looking acts in the DRI classification are concerned with grounding; they are listed below.[6]

- **Understanding-act**
  - **Signal-non-understanding**
  - **Signal-understanding**
    - **Acknowledge**
    - **Repeat-rephrase**
    - **Completion**
  - **Correct-misspeaking**

2.2.4 Turn-taking Acts

The classification in (Traum and Hinkelman, 1992) also includes a class of acts having to do with the management of the turn–i.e., who is speaking at any given point. Actions in this

[6]We should note that some of the grounding acts in (Traum and Hinkelman, 1992) are not included in the DRI scheme - for example, requests for acknowledgments.

class include **take-turn**, **keep-turn**, **release-turn**, **assign-turn**. The DRI scheme does not include actions of this type at the moment.

2.2.5 Multiple Dialogue Acts

One hypothesis shared both by CA theory and by the DRI proposal is that a locutionary act may generate more than one dialogue act. For example, a locutionary act such as *okay* is typically used to perform actions at both the grounding level and at the core speech act level at the same time; but it is also possible to perform multiple actions at the core speech act level–e.g., an utterance such as *There is an engine at Avon* in the TRAINS domain can be both an **Assert** and an **Open-option**. Following (Goldman, 1970), Poesio and Traum assume that in these cases multiple events are GENERATED by a single locutionary event.

Chapter 3

Dynamic Information State in Dialogue

As mentioned in the introductory chapter, the concept of the *information state* of a dialogue is a very general one, encompassing whatever information is needed to represent the dialogue context to be able to understand and appropriately participate in the dialogue. In this chapter we present a characterization of information states as feature structures, presenting two distinct theories of task-oriented dialogue in these terms, one developed by Cooper and Larsson on the basis of work by Ginzburg (Cooper and Larsson, 1999; Ginzburg, 1998), and one developed by Poesio and Traum Poesio and Traum (1998). Examples are given with reference to the Autoroute domain on the basis of an investigation of the Autoroute corpus collected by DERA.[1]

3.1 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., (Albesano *et al.*, 1997)) can be represented in terms of feature structures as in (3.1): a list of fields which the system must fill before being able to ask a query.

$$\begin{bmatrix} l_1 & = & a_1 \\ l_2 & = & a_2 \\ \dots & & \\ l_n & = & a_n \end{bmatrix} \tag{3.1}$$

[1] 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.

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 (3.2).[2]

$$(3.2) \qquad \begin{bmatrix} \text{start} & = & \\ \text{end} & = & \\ \text{stime} & = & \end{bmatrix}$$

This notation can be interpreted in various ways. One interpretation we have adopted is that in terms of typed records as discussed in (Cooper , 1998a,b). 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 (3.1) is of the record type in (3.3).

$$(3.3) \qquad \begin{bmatrix} l_1 & : & T_1 \\ l_2 & : & T_2 \\ \ldots & & \\ l_n & : & T_n \end{bmatrix}$$

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 (3.2). 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 (3.2), 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 middle-grounds between the two for information which is

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

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:

(3.4)
$$\begin{bmatrix} \text{participant A} & : & \begin{bmatrix} \text{modality}_1 & : & \begin{bmatrix} \text{infotype}_1 & : & T_1 \\ \dots & & \\ \text{infotype}_\text{k} & : & T_k \end{bmatrix} \\ \dots & & \\ \text{modality}_\text{n} & : & [\ \dots\ ] \end{bmatrix} \\ \text{participant B} & : & [\ \dots\ ] \end{bmatrix}$$

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 (3.4), assuming T_k is the type stack, then (3.5) would be an example update operation:

(3.5) $\text{pop}(\text{participant A.modality}_1\text{.infotype}_\text{k})$

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

In the rest of this chapter we will present two different theories of information state which correspond to the general framework presented here.

3.2 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 Ginzburg (1995a,b, 1998) 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 (3.4) for this information state type is shown in (3.6).

(3.6)
$$\begin{bmatrix} \text{Private} & : & \begin{bmatrix} \text{Bel} & : & \text{Set(Prop)} \\ \text{Agenda} & : & \text{Stack(Action)} \end{bmatrix} \\ \text{Common} & : & \begin{bmatrix} \text{Bel} & : & \text{Set(Prop)} \\ \text{QUD} & : & \text{Stack(Question)} \end{bmatrix} \end{bmatrix}$$

That is, we made a division between Private and Common 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. (This feels like it should be an oversimplification and that it will be necessary to have a separate field for goals.) 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.

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)*.

(3.7) $$\frac{q\text{:Question}}{\text{respond}(q)\text{:Action}} \quad \frac{q\text{:Question}}{\text{raise}(q)\text{:Action}} \quad \frac{p\text{:Prop}}{\text{instruct}(p)\text{:Action}}$$

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.

The first Common 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 common propositions. The common 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 Common 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.

The following example shows how updates change the information state during a dialogue. (3.8) shows the information state before utterance U4 in the Autoroute dialogue presented in full in Appendix A. (3.9) shows the utterance itself and the accompanying updates. (3.10)

shows the information state after the updates.

(3.8)

$$\begin{bmatrix} A = \begin{bmatrix} \text{Private} = \begin{bmatrix} \text{Bel} = 3(\text{i}).\text{A.Private.Bel} \\ \text{Agenda} = \langle \begin{array}{l} \text{raise}(\textit{Where does B want to start?}), \\ \text{raise}(\textit{Where does B want to go?}), \\ \text{raise}(\textit{What time does B want to make the journey?}), \\ \text{raise}(\textit{Does B want the quickest or shortest route?}) \end{array} \rangle \end{bmatrix} \\ \text{Common} = \begin{bmatrix} \text{Bel} = 3(\text{i}).\text{A.Common.Bel} \cup \{\textit{B wants a route from A}\} \\ \text{QUD} = \langle\rangle \end{bmatrix} \end{bmatrix} \\ B = \begin{bmatrix} \text{Private} = \begin{bmatrix} \text{Bel} = 3(\text{i}).\text{B.Private.Bel} \\ \text{Agenda} = \langle\rangle \end{bmatrix} \\ \text{Common} = \begin{bmatrix} \text{Bel} = 3(\text{i}).\text{B.Common.Bel} \\ \text{QUD} = \langle\rangle \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

(3.9) **U4 A <Where would you like to start your journey.>**
pop(A.Private.Agenda)
push(*Where does B want to start the journey?*, A.Common.QUD)
push(respond(fst(B.Common.QUD)), B.Private.Agenda)
push(*Where does B want to start the journey*, B.Common.QUD)

(3.10)

$$\begin{bmatrix} A = \begin{bmatrix} \text{Private} = \begin{bmatrix} \text{Bel} = 3(\text{ii}).\text{A.Private.Bel} \\ \text{Agenda} = \langle \begin{array}{l} \text{raise}(\textit{Where does B want to go?}), \\ \text{raise}(\textit{What time does B want to make the journey?}), \\ \text{raise}(\textit{Does B want the quickest or shortest route?}) \end{array} \rangle \end{bmatrix} \\ \text{Common} = \begin{bmatrix} \text{Bel} = 3(\text{ii}).\text{A.Common.Bel} \\ \text{QUD} = \langle \text{Where does B want to start the journey?} \rangle \end{bmatrix} \end{bmatrix} \\ B = \begin{bmatrix} \text{Private} = \begin{bmatrix} \text{Bel} = 3(\text{ii}).\text{B.Private.Bel} \\ \text{Agenda} = \langle \text{respond(fst(B.Common.QUD))} \rangle \end{bmatrix} \\ \text{Common} = \begin{bmatrix} \text{Bel} = 3(\text{ii}).\text{B.Common.Bel} \\ \text{QUD} = \langle \textit{Where does B want to start the journey} \rangle \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Note that the Common fields are not shared between 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.

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 (3.11).

(3.11) Stack: push, pop
Set: add an element

3.3 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 Poesio and Traum (1997, 1998). One of the central concerns of this work, which builds upon previous work by Traum (1994), is the GROUNDING process, by which common ground is established (Clark and Schaefer, 1989; Traum and Hinkelman, 1992). 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, as they are called in (Traum and Hinkelman, 1992).

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 (3.12)

(3.12)
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 (3.13).

(3.13)
$$\begin{bmatrix} G & : & \text{PT-record} \\ DU_1 & : & \text{PT-record} \\ \ldots & & \\ DU_n & : & \text{PT-record} \\ UDUs & : & \text{List} \\ INT & : & \text{List} \end{bmatrix}$$

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-

[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 Poesio and Traum (1997, 1998), 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 (3.4).

Traum model there are several kinds of information kept int he shared (G) and semi-public (DUs) part of a participants information state. First, as described further in the next chapter, an explicit history of the dialogue acts 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 (3.14) (abbreviated PT-record in (3.13)).

(3.14)
$$\begin{bmatrix} \text{DH} & : & \text{List} \\ \text{OBL} & : & \text{List} \\ \text{SCP} & : & \text{List} \\ \text{OPT} & : & \text{List} \end{bmatrix}$$

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 (3.13)), 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.

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

To summarize, each information state will be of the type in (3.16).

(3.16)
$$\begin{bmatrix} \text{A:} \begin{bmatrix} \text{G:} \begin{bmatrix} \text{OBL: } < \ldots > \\ \text{DH: } < \ldots > \\ \text{SCP: } < \ldots > \\ \text{OPT: } < \ldots > \end{bmatrix} \\ \text{INT: } < \ldots > \\ \text{DUi: } \ldots \\ \text{UDUS: } < \text{DUi}, \ldots > \end{bmatrix} \\ \text{B:} \begin{bmatrix} \text{G:} \begin{bmatrix} \text{OBL: } < > \\ \text{DH: } < > \\ \text{SCP: } < \ldots > \\ \text{OPT: } < \ldots > \end{bmatrix} \\ \text{INT: } < \ldots > \\ \text{DUi: } \ldots \\ \text{UDUS: } < \text{DUi}, \ldots > \end{bmatrix} \end{bmatrix}$$

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. (3.17) 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.

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

$$\begin{bmatrix} \text{A:} \begin{bmatrix} \text{G:} \begin{bmatrix} \text{OBL: } < \text{B understanding-act 4B, A address 3C} > \\ \text{SCP: } < \text{B wants a route} > \end{bmatrix} \\ \text{INT: } < \text{get(SP), get(DEST), get(ST), get(ROUTE-TYPE), give B route(SP,DEST,ST,ROUTE-TYPE)} > \\ \text{DU4A: } \begin{bmatrix} \text{OBL: } < \text{B answer 4} > \\ \text{DH: } < 4\text{: info-request} > \end{bmatrix} \end{bmatrix} \\ \text{B:} \begin{bmatrix} \text{G:} \begin{bmatrix} \text{OBL: } < \text{B understanding-act 4B, A address 3C} > \\ \text{SCP: } < \text{B wants a route} > \end{bmatrix} \\ \text{INT: } < \text{get a route from Malvern to Edwinstowe} > \\ \text{DU4B: } \begin{bmatrix} \text{OBL: } < \text{B answer 4} > \\ \text{DH: } < 4\text{: info-request} > \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

(3.18) 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.)

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

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

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 (3.19).

(3.19) G+= DU

Chapter 4

Dialogue Moves and Information States

There are several ways in which dialogue moves, such as those discussed in Chapter 2, can play a role in the information state in dialogue. For one thing, they can be a part of the information structure itself, as in the **DH** field in Scheme 2, described above. Another important aspect depends on the updating of information state. One way in which moves can play a role in updates is to have specific updates associated with the observation or performance of particular moves. Another way is to see moves, themselves, as a bundle of update instructions. From this viewpoint coding moves is really the same thing as coding information state updates, perhaps in a more compact form. The information state models we have presented encompass both of these options. While moves are not seen as directly part of the information state in the Cooper-Larsson model, they are used as shorthand update instructions. On the other hand, in the Poesio-Traum model, there are specific update rules associated with observation of each kind of dialogue move. In this chapter we will explore each of these uses in more detail.

4.1 Moves as Update-bundles in Scheme 1

From a more theoretical perspective, we are interested in characterizing moves (such as those used in the Map Task, Carletta *et al.*, 1996, or DRI, Allen and Core, ms) in terms of transitions between information states, in a reasonably precise way without committing to a particular semantic theory. The particular preliminary formulation we present here builds on the kinds of moves used in the Map Task but with the addition of arguments indicating the agents and the contents of the utterances involved. What one notices when one begins to look at the information states is that the kind of division that seems natural when one is only thinking in terms of moves perhaps should be refined when one derives one's moves from information state transition types. For example, there is no real motivation to distinguish between *query-w* (wh-query) and *query-yn* (according to the assumptions under which we have done this

particular annotation) since the operations on information states are exactly similar except for the fact that for *query-w* it is a precondition that q is a *wh*-question whereas for *query-yn* there is a precondition that q is a yes-no-question. On the other hand some move types have to be broken down into various subtypes such as successful and unsuccessful (depending on whether the hearer accepts the other agent's response or not), suggesting perhaps that a neater analysis of moves would break them down into smaller units, including silent moves in which an agent tries to integrate the information from the last move into her information state.

query-w(A, B, q) "A asks B q"

Preconditions

fst(A.Private.Agenda) = raise(q)
whq(q)

Effects

pop(A.Private.Agenda)
push(q, A.Common.QUD)

push(respond(fst(B.Common.QUD)), B.Private.Agenda)
push(q, B.Common.QUD)

reply-w(A, B, q, p) – **successful** "A replies to B with p as a response to question q"

Preconditions

fst(A.Private.Agenda) = respond(fst(A.Common.QUD))
fst(A.Common.QUD) = q
whq(q)

Effects

pop(A.Private.Agenda)
add(p, A.Common.Bel)
pop(A.Common.QUD) # Should be: pop all questions from top of QUD to which q is an answer.

add(p, B.Common.Bel)
pop(B.Common.QUD) # Should be: pop all questions from top of QUD to which q is an answer.

reply-w(A, B, q, p, q') – **unsuccessful** "A responds to B concerning question q with response p, which B fails to integrate, generating a clarification question q' on B's Agenda"

Preconditions

fst(A.Private.Agenda) = respond(fst(A.Common.QUD))
fst(A.Common.QUD) = q

Effects

pop(A.Private.Agenda)
add(p, A.Common.Bel)
pop(A.Common.QUD) # Should be: pop all questions from top of QUD to which q is an answer.

push(raise(q', B.Private.Agenda))

query-yn(A, B, q) "A asks q of B"

Preconditions

fst(A.Private.Agenda) = raise(q)
ynq(q)

Effects

pop(A.Private.Agenda)
push(q, A.Common.QUD)

push(respond(fst(B.Common.QUD)), B.Private.Agenda)
push(q, B.Common.QUD)

reply-y(A, B, q, p) – successful "A responds to B concerning q with p"

Preconditions

fst(A.Private.Agenda) = respond(fst(A.Common.QUD))
fst(A.Common.QUD) = q
ynq(q)
p = yes(q)

Effects

pop(A.Private.Agenda)
add(p, A.Common.Bel)
pop(A.Common.QUD) # Should be: pop all questions from top of QUD to which q is an answer.

add(p, B.Common.Bel)
pop(B.Common.QUD) # Should be: pop all questions from top of QUD to which q is an answer.

4.2 Move-Based updates in Scheme 2

The Poesio-Traum theory of update, as described in Poesio and Traum (1998), uses the modified DRI set of dialogue moves, as described in Section refdri-move-sec.

Updating the Poesio-Traum framework information states due to the performance of a new action can be conceived of as a multi-step process. While we do not advocate that it is necessary or desirable (or perhaps even possible) to proceed exactly in this fashion during actual dialogue processing, it does result in consistent new states in the theory.

The general principle behind this update procedure is the following. There should be one new DU for each grounding act, and one grounding act each for (1) operations on a previous DU or (2) initiation of new material that will have to be grounded. Updates for an utterance can be performed using the following sequence of steps for each utterance:

1. create a new DU, (e.g., DU1) and push this on the top of UDUs (=CDU).
2. perform updates based on the observance of the backward grounding act i.e., for ack(DU0): G+= DU0 , remove(DU0,UDUs), push(G.DH,ack(DU0).
3. put forward looking acts and backward acts other than grounding acts into CDU.DH
4. apply update rules to all DRS's which contain newly added forward and backward acts.

Updates due to the effects of dialogue acts are summarized in (4.1). In these rules, `e: Act` gives the name `e` to an act of type `Act`, which has been performed. `K :: e` means that Act `e` is a member of the DH field in DRS `K`.

(4.1)

Action Condition		**Update**
K:: e : ForwardAct(A,B,X)	→	push(G.OBL,UnderstandingAct(B,K))
K:: s: statement(A,B,K')	→	push(K.SCP,SCP(A,K)) push(K.SCP,Accept(B,s) → SCP(B,K'))
K:: e : IAFutA(A,B,AT)	→	push(K.Opt,Opt(B,AT)) push(K.OBL, Accept(B,e) → Obliged(B,AT))
K:: di: Direct(A,B,AT)	→	push(K.OBL,Obliged(B,A,Address(di)))
K:: CSFA(A,B,AT)	→	push(K.Opt,Opt(B,AT))
K:: Commit(A,B,AT)	→	push(K.OBL,Obliged(A,B,AT))
K:: o : Offer(A,B,AT)	→	push(K.OBL, Accept(B,o) →Obliged(B,AT))
K:: q : Info-req(A,B,Q)	→	push(K.OBL, Obliged(B,Answer(q)))
K:: q : Check(A,B,K')	→	push(K.OBL,:Obliged(B,A,Answer(q))) push(K.SCP, Agree(B,s) →SCP(A,K'))
K:: Agree(A,B,K')	→	push(K.SCP, SCP(A,K'))
K:: Answer(A,B,q,K')	→	push(K.SCP, SCP(A,B,ans(q,K')))

In addition to the actual acts appearing in the DH field of Dus and G, some of the rules in (4.1) refer to more abstract action types, as indicated in Section 2.2. (4.2) shows the needed taxonomic relationships.

(4.2)

Move Sub-Type		**Move Super-Type**
e : Ack(K)	→	x : UnderstandingAct(K)
e : Accept(e')	→	e : Address(e')
e : Reject(e')	→	e : Address(e')
e : Accept(e')	→	e : BackwardAct(e')
e : Address(e')	→	e: BackwardAct(e')
e : Agree(e')	→	e : BackwardAct(e')
e : Answer(e',P)	→	e : BackwardAct(e')
e : Statement(e')	→	e : ForwardAct(e')
e : IAFutA(e')	→	e : ForwardAct(e')
e : CSFA(e')	→	e : ForwardAct(e')
e : Info-req(e')	→	e : ForwardAct(e')
e : Check(e')	→	e : Info-req(e')
e : Direct(e')	→	e : IAFutA(e')
e : Commit(e')	→	e : CSFA(e')
e : Offer(e')	→	e : CSFA(e')
e : Agree(ch), ch : check(e')	→	e : Answer(e',yes)

Chapter 5

Further Update Rules and Example Codings

In this chapter we give further specifics as to how to code information state updates, using illustrative examples from the Dialogue in Appendix A. More complete annotations may be found in Appendix B, containing the whole dialogue in scheme 1, and Appendix C, containing the first 7 utterances in scheme 2.

5.1 Updating the Cooper-Larsson Information States

In order to code information states, a number of further decisions must be made as to which updates to perform. The overall basic strategy which regulates the flow of information between the Agenda and QUD is as follows:

- **Rule 1:** If a dialogue participant A has *raise(q)* on the agenda, then A should use her turn to utter a question which expresses q and push q onto QUD.
- **Rule 2:** If a dialogue participant A has *respond(fst(A.Common.QUD))* on top of the agenda and q on the top of QUD then A should use her turn to utter an appropriate response to q, pop the Agenda and the QUD and add the response to Common Beliefs.
- **Rule 3:** If a dialogue participant A notices that q has been pushed onto QUD by the other dialogue participant then A also pushes q onto QUD and pushes *respond(fst(A.Common.QUD))* onto her agenda.
- **Rule 4:** If a dialogue participant A notices that the other dialogue participant has responded to a question with p then A should attempt to integrate p with her Private and Common Beliefs. If the integration is successful then A should add p to the Common Beliefs and pop QUD. If the integration is unsuccessful then A pushes an action to raise

a clarifying question q' onto her Agenda. (Her Common Beliefs and QUD remain unchanged.)

A final state is one in which all participants' Agenda and QUD are empty.

This basic strategy embodies OPTIMISM, with respect to grounding. As soon as A has uttered something as a response to a question, she enters the response into Common Beliefs. As soon as A raises a question, the question is entered into QUD. A cautious strategy would wait until there is some kind of feedback (which may involve simply continuing with another relevant utterance) before entering the common information. We are not sure what the consequences of pursuing an optimistic or a cautious strategy are. We think that the kind of annotation we are pursuing would allow us to experiment with annotations for different strategies and see if there are any empirical consequences (i.e. dialogue phenomena that can be accounted for by one strategy but not the other) or consequences involving computational efficiency (in terms of the number of operations that have to be performed overall).

Most updates in the dialogue can be explained either by the move definitions in Section 4.1 or by the rules given in the basic strategy described above.

5.1.1 Annotating Autoroute Dialogue 127 using Cooper-Larsson Information States

In this section, we give an annotated example of coding using Scheme 1. In Appendix B, we give the complete information states from this coding, produced using the GATE version of the TranScript tool described in Chapter 6.

We assume that once A and B have completed the initial greeting procedure, they have established contact and have a shared belief that they are attending each other. B wants assistance from A, and A has a single item on the agenda, to raise a question concerning how A can help B.

$$\left[\begin{array}{lll} \text{A} & = & \left[\begin{array}{lll} \text{PRIVATE} & = & \left[\text{AGENDA} = \left\langle\ \textbf{raise}(\text{'}\textit{What does B want}\text{'})\ \right\rangle\right] \\ \text{SHARED} & = & \left[\text{BEL} = \left\{\begin{array}{l}\text{'}\textit{B has A's attention}\text{'}\\ \text{'}\textit{A has B's attention}\text{'}\end{array}\right\}\right] \end{array}\right] \\ \text{B} & = & \left[\begin{array}{lll} \text{PRIVATE} & = & \left[\text{BEL} = \left\{\ \text{'}\textit{B wants assistance}\text{'}\ \right\}\right] \\ \text{SHARED} & = & \left[\text{BEL} = \left\{\begin{array}{l}\text{'}\textit{B has A's attention}\text{'}\\ \text{'}\textit{A has B's attention}\text{'}\end{array}\right\}\right] \end{array}\right] \end{array}\right]$$

Following rule 1 in Section 5.1, A uses the turn to raise the question "How can A help B?" on the top of the agenda by asking a **query-w** move. As a result the information state is updated with the effects of the move, as specified in the definition in Section 4.1. B also pushes a slightly different question onto QUD, the question "What does B want?", which is the question B chooses to answer. Following rule 3, B pushes on the agenda the action to respond to the question topmost on QUD.

(3)
A <How can I help you.>
(Pause: 1)

popRec(A.PRIVATE.AGENDA)
pushRec(A.SHARED.QUD, '*How can A help B?*')
pushRec(B.SHARED.QUD, '*How can A help B?*')
pushRec(B.SHARED.QUD, '*What does B want from A?*')
pushRec(B.PRIVATE.AGENDA, **respond**('*What does B want from A?*'))

$$\left[\begin{array}{lll} \text{A} & = & \left[\begin{array}{lll} \text{PRIVATE} & = & [] \\ \text{SHARED} & = & \left[\begin{array}{lll} \text{BEL} & = & \left\{\begin{array}{l} \text{'}B\text{ has }A\text{'s attention'} \\ \text{'}A\text{ has }B\text{'s attention'} \end{array}\right\} \\ \text{QUD} & = & \left\langle \text{'How can A help B?'} \right\rangle \end{array}\right] \end{array}\right] \\ \text{B} & = & \left[\begin{array}{lll} \text{PRIVATE} & = & \left[\begin{array}{lll} \text{BEL} & = & \left\{ \text{'}B\text{ wants assistance'} \right\} \\ \text{AGENDA} & = & \left\langle \textbf{respond}(\text{'What does B want from A?'}) \right\rangle \end{array}\right] \\ \text{SHARED} & = & \left[\begin{array}{lll} \text{BEL} & = & \left\{\begin{array}{l} \text{'}B\text{ has }A\text{'s attention'} \\ \text{'}A\text{ has }B\text{'s attention'} \end{array}\right\} \\ \text{QUD} & = & \left\langle\begin{array}{l} \text{'What does B want from A?'} \\ \text{'How can A help B?'} \end{array}\right\rangle \end{array}\right] \end{array}\right] \end{array}\right]$$

B now uses rule 2 and generates a **reply-w**-move to answer the question topmost on B's QUD, and B updates his information state accordingly. However, A cannot integrate the effects of this move since it doesn't answer the topmost question on A's QUD ("A route please" is not an answer to "How can A help B?"). To remedy this, A pushes the relevant question onto QUD; this can be seen as a kind of accommodation.

(4)
B <A route please.>
(Pause: 1)

pushRec(A.SHARED.QUD, '*What does B want from A?*')
popRec(B.PRIVATE.AGENDA)
popRec(B.SHARED.QUD)
popRec(B.SHARED.QUD)
addRec(B.SHARED.BEL, '*B wants a route from A*')

$$\left[\begin{array}{lll} & \text{PRIVATE} = [] & \\ \text{A} = & \text{SHARED} = \left[\begin{array}{ll} \text{BEL} = & \left\{\begin{array}{l} \text{'}B\text{ has }A\text{'s attention'} \\ \text{'}A\text{ has }B\text{'s attention'} \end{array}\right\} \\ \text{QUD} = & \left\langle\begin{array}{l} \text{'}\textit{What does B want from A?}\text{'} \\ \text{'}\textit{How can A help B?}\text{'} \end{array}\right\rangle \end{array}\right] & \\ & \text{PRIVATE} = \left[\text{BEL} = \left\{\text{'}B\text{ wants assistance'}\right\}\right] & \\ \text{B} = & \text{SHARED} = \left[\text{BEL} = \left\{\begin{array}{l} \text{'}B\text{ wants a route from }A\text{'} \\ \text{'}B\text{ has }A\text{'s attention'} \\ \text{'}A\text{ has }B\text{'s attention'} \end{array}\right\}\right] & \end{array}\right]$$

Now, A can integrate the effects of the reply-w move; A pops the agenda and adds the proposition "B wants a route from A" to the shared beliefs. As it happens, A's second question has also been answered (A can help B by giving a route to A) so A pops the QUD again. In order to be able to give a route to B, A needs to have some questions concerning destination etc. answered, so A pushes the raising of these questions onto his agenda.

popRec(A.SHARED.QUD)
addRec(A.SHARED.BEL, '*B wants a route from A*')
popRec(A.SHARED.QUD)
pushRec(A.PRIVATE.AGENDA, **raise**('*Does B want the quickest or shortest route?*'))
pushRec(A.PRIVATE.AGENDA, **raise**('*What time does B want to go?*'))
pushRec(A.PRIVATE.AGENDA, **raise**('*Where does B want to go?*'))
pushRec(A.PRIVATE.AGENDA, **raise**('*Where does B want to start?*'))

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

Since A has the turn, A follows rule 1 and utters the appropriate question, which is successfully interpreted and integrated by B according to the definition of **query-w**.

(5)
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?*'))

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

,

5.2 Updating the Poesio-Traum Information States

Most of the updating of the Poesio-Traum information states comes as a direct result of incorporating the moves and their effects into the information state, as presented in Section 4.2. There are, however several additional steps that happen between processing of moves. First, there is the calculation of inference on the basis of the new states. These include the rules in (5.1). The first one is that of Obligation Resolution, lifting obligations for performed actions. The second one is a specialization of Modus Ponens, specifically for resolving conditional attitudes when the antecedent action has been performed. The third rule is one of Intention Resolution, in which an agent drops an intention to perform actions it has already performed[1].

(5.1)

In(K.OBL,Obliged(A,B,AT)), In(K.DH,AT(A))	→	remove(K.OBL,Obliged(A,B,AT))
In(K.DH, phi), In(K.X phi → psi)	→	remove(K.X, phi → psi), push(K.X, psi)
In(K.INT,AT), In(K.DU.DH, AT)	→	remove(K.INT, AT)

Finally, after this sort of inference, there must be a process of DELIBERATION, in which the agent adopts new intentions and (perhaps) performs further utterances (which would then update the information state using the updates associated with the various dialogue moves).

[1]This third rule is actually a simplification, since an agent may not drop the intention until the agent believes the action has been grounded. To be more accurate we would need to add a layer of private beliefs, and then specific rules for when the agent will adopt such beliefs given the current groundedness situation.

Some example intention update rules are shown in (5.2), while (5.3) shows a likely rule leading to performing a dialogue move.The first rule shows how an agent adopts the most immediate obligation as an intention, thus deciding to meet its social commitment to act. SUBACT and PRECONDITION refer to bits of the agent's STATIC INFORMATION STATE, in particular, knowledge of recipes for action. The latter two rules thus represent a kind of planning of how to achieve a complex actions.

(5.2)

Top(X.G.OBL,Obliged(X, AT))	→	push(X.Int, AT)
In(X.Int(AT), Subact(AT, at1))	→	push(X.Int,at1)
In(X.Int(AT), precondition(AT, pre1)	→	push(X.Int,achieve(pre1))

(5.3)

Top(X.Int,AT), Turn(X), Cando(X,AT)	→	Perform(X,AT)

5.2.1 Annotation of Autoroute Dialogue 127 using Poesio-Traum Information States

In this section, we give an annotated example of coding using Scheme 2. In Appendix C, we give the complete information states from this coding, produced using the TranScript tool described in Chapter 6.

We assume that in the initial state there are no obligations, but that the system has an intention to be polite, which explains the first utterance, and then a persistent intention to assist, or inform, or whatever, and respond. (Strictly speaking, the latter intention might be represented throughout the dialogue as it probably underlies utterance 22. To simplify the representations, however, we have excluded it.) In addition, we assume that B has the intention of getting a route from Malvern to Edwinstowe.

$$\begin{bmatrix} A: & \begin{bmatrix} G: & \begin{bmatrix} OBL: < > \\ DH: < > \end{bmatrix} \\ INT: & < \text{GREET, OFFER HELP} > \\ UDUS: & < > \end{bmatrix} \\ B: & \begin{bmatrix} G: & \begin{bmatrix} OBL: < > \\ DH: < > \end{bmatrix} \\ UDUS: & < > \\ INT: & < \text{GET A ROUTE FROM MALVERN TO EDWINSTOWE} > \end{bmatrix} \end{bmatrix}$$

U1 [A]: Welcome to the Route Planning Service.

U2 [A]: How can I help you.

The effect of a new utterance is to create a new DU (DU2), which becomes part of the information state of both agents. Note that the contents of DU2 are different for each agent. Move 2 (an INFO-REQUEST) It also results in two obligations for B: to signal understanding, and to answer the question.

$$\left[\begin{array}{ll} \text{A:} & \left[\begin{array}{ll} \text{G:} & \left[\begin{array}{l} \text{OBL: < B understanding-act DU2A >} \\ \text{DH: < >} \end{array}\right] \\ \text{DU2A:} & \left[\begin{array}{l} \text{OBL: < B answer 2 >} \\ \text{DH: < 1: greeting, 2: info-request >} \end{array}\right] \\ \multicolumn{2}{l}{\text{UDUS: < DU2A >}} \\ \multicolumn{2}{l}{\text{INT: < >}} \end{array}\right] \\ \text{B:} & \left[\begin{array}{ll} \text{G:} & \left[\begin{array}{l} \text{OBL: < B understanding-act DU2B >} \\ \text{DH: < >} \end{array}\right] \\ \text{DU2B:} & \left[\begin{array}{l} \text{OBL: < B answer 2 >} \\ \text{DH: < 1: greeting, 2: info-request >} \end{array}\right] \\ \multicolumn{2}{l}{\text{UDUS: < DU2B >}} \\ \multicolumn{2}{l}{\text{INT: < get a route from Malvern to Edwinstowe >}} \end{array}\right] \end{array}\right]$$

As in this dialogue there are no cases of misunderstanding, we will only specify the view of the DU, of G and of UDUs by one agent (A) in what follows.

The obligations to answer the question and perform an understanding act result in B acquiring the intention to perform a directive:

$$\left[\begin{array}{ll} \text{A:} & \left[\begin{array}{ll} \text{G:} & \left[\begin{array}{l} \text{OBL: < B understanding-act DU2A, >} \\ \text{DH: < >} \end{array}\right] \\ \text{DU2A:} & \left[\begin{array}{l} \text{OBL: < B answer 2 >} \\ \text{DH: < 1: greeting, 2: info-request >} \end{array}\right] \\ \multicolumn{2}{l}{\text{UDUS: < DU2A >}} \\ \multicolumn{2}{l}{\text{INT: < >}} \end{array}\right] \\ \text{B:} & \left[\text{INT: < perform understanding act, answer, get a route from Malvern to Edwinstowe >}\right] \end{array}\right]$$

U3 [B]: A route please.

Two new DUS, DU3A and DU3B, are created. B's utterance performs an implicit acknowledgment of DU2, resulting in DU2 being added to G and in the obligation to perform an understanding act being removed. Secondly, B performs a DIRECTIVE, which results in two obligations for A: performing an understanding act and addressing the directive. Thirdly, B commits himself to the proposition that he wants a route.

One question here is when B's obligation to answer 2 disappears. The assumption here is that the obligation disappears when the fact that B performed an answer becomes part of G; hence for the moment the obligation persists.

$$\left[\begin{array}{ll} & \left[\begin{array}{ll} \text{G:} & \left[\begin{array}{l} \text{OBL: < A understanding-act DU3B, B answer 2 >} \\ \text{DH: < 1: greeting, 2: info-request, 3A: acknowledge >} \end{array}\right] \\ \text{A:} & \left[\begin{array}{ll} \text{DU3B:} & \left[\begin{array}{l} \text{OBL: < A address 3C >} \\ \text{SCP: <B wants a route >} \\ \text{DH: < 3B: answer, 3C:directive >} \end{array}\right] \\ \multicolumn{2}{l}{\text{UDUS: < DU3B >}} \\ \multicolumn{2}{l}{\text{INT: < >}} \end{array}\right] \end{array}\right] \\ \text{B:} & \left[\text{INT: < get a route from Malvern to Edwinstowe >}\right] \end{array}\right]$$

At this point, we hypothesize that A assumes an obligation to give B a route by accepting B's directive (which we assume A does before actually telling B); this in turn results in A acquiring an intention to give B a route. (An alternative explanation in terms of intentions would be to assume that A assumes B's intention as his own.) We assume here that A does this by adding the obligation to his own private mental state:

$$\left[\begin{array}{ll} & \left[\begin{array}{ll} \text{G:} & \left[\begin{array}{l} \text{OBL: < A understanding-act DU3B, B answer 2 >} \\ \text{DH: < 1: greeting, 2: info-request, 3A: acknowledge >} \end{array}\right] \\ \text{A:} & \left[\begin{array}{ll} \text{DU3B:} & \left[\begin{array}{l} \text{OBL: < A address 3C >} \\ \text{SCP: <B wants a route >} \\ \text{DH: < 3B: answer, 3C:directive >} \end{array}\right] \\ \multicolumn{2}{l}{\text{UDUS: < DU3B >}} \\ \multicolumn{2}{l}{\text{OBL: < give B a route >}} \\ \multicolumn{2}{l}{\text{INT: < perform accept >}} \end{array}\right] \end{array}\right] \\ \text{B:} & \left[\text{INT: < get a route from Malvern to Edwinstowe >}\right] \end{array}\right]$$

And then acquiring the intention:

A:
- G:
  - OBL: < A understanding-act DU3B, B answer 2 >
  - DH: < 1: greeting, 2: info-request, 3A: acknowledge >
- DU3B:
  - OBL: < A address 3C >
  - SCP: <B wants a route >
  - DH: < 3B: answer, 3C:directive >
- UDUS: < DU3B >
- INT: < perform accept, give B a route(SP,DEST,ST,ROUTE-TYPE) >

B:
- INT: < get a route from Malvern to Edwinstowe >

At this point, A reasons about what is needed in order to address the directive; this results in A's agenda being updated with the appropriate subtasks.

A:
- G:
  - OBL: < A understanding-act DU3B, B answer 2 >
  - DH: < 1: greeting, 2: info-request, 3A: acknowledge >
- DU3B:
  - OBL: < A address 3C >
  - SCP: <B wants a route >
  - DH: < 3B: answer, 3C:directive >
- UDUS: < DU3B >
- INT: < perform accept, get(SP), get(DEST), get(ST), get(ROUTE-TYPE), give B route (SP,DEST,ST,ROUTI

B:
- INT: < get a route from Malvern to Edwinstowe >

In what follows, we will skip B's INT for brevity (hence, skipping B's state altogether) and A's DH.

U4 [A]: Where would you like to start your journey.

Again, two new DUs are created. An implicit acknowledgment is performed, which results in the previously pending DU (DU3B) being merged with G, thus satisfying the obligation to perform an understanding act. Grounding DU3B results in accepting U3 as an answer to U2, so all the obligations in the previous IS are now dealt with. A's intention to perform an accept is popped.

A:
- G:
  - OBL: < B understanding-act 4B, A address 3C >
  - SCP: <B wants a route >
- INT: < get(SP), get(DEST), get(ST), get(ROUTE-TYPE), give B route(SP,DEST,ST,ROUTE-TYPE) >
- DU4B:
  - OBL: < B answer 4 >
  - DH: < 4: info-request >

U5 [B]: Malvern.

DU4A is merged with G. As in the previous case, 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.

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

There are a couple of ways of dealing with U6. We have assumed here that A needs more stringent evidence for accepting B's assertion, and therefore plans a **check** act instead of simply going on with the next question.

An issue here is that a simple acknowledgment would not be a proper answer to 6; it seems that B has to give a true answer. Yet, 6 does not look like a real yes-no question, in the sense that A seems to just want confirmation. For the moment, we have modeled this by assuming a **check**. An alternative way would be to assume that a **repeat-rephrase** results in an obligation for B to answer. (Note that U6 couldn't be an ASSERT under the formalization of assert given above - A's is clearly not committed to this fact.)

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

U6 [A]: Starting in Great Malvern.

As a result of the repeat-rephrase, DU5B gets acknowledged; but the **assert** is not yet accepted.

$$\left[\begin{array}{l} \\ \text{A:} \left[\begin{array}{l} \text{G:} \left[\begin{array}{l} \text{OBL: < B UNDERSTANDING-ACT DU6B, A ADDRESS 5B, A ADDRESS 3C >} \\ \text{SCP: <B SCP SP = MALVERN, B WANTS A ROUTE >} \end{array}\right] \\ \text{INT: < GET(SP), >} \\ \text{INT: < GET(DEST), GET(ST), GET(ROUTE-TYPE), GIVE B ROUTE(SP,DEST,ST,ROUTE-TYPE) >} \\ \text{DU6B:} \left[\begin{array}{l} \text{OBL: < B ANSWER 6B >} \\ \text{DH: < 6B: CHECK >} \end{array}\right] \end{array}\right] \\ \\ \end{array}\right]$$

U7 [B]: Yes.

U7 answers U6 positively and reasserts B's commitment to the proposition that SP= Malvern. B's answer means that A can now accept the proposition; now both B and A are committed to the proposition, so A's obligation to address 5B can be removed. B's obligation to answer 6B persists.

$$\left[\begin{array}{l} \text{A:} \left[\begin{array}{l} \text{G:} \left[\begin{array}{l} \text{OBL: < A UNDERSTANDING-ACT DU7B, A ADDRESS 3C >} \\ \text{SCP: <B SCP SP = MALVERN, A SCP SP = MALVERN, B WANTS A ROUTE >} \end{array}\right] \\ \text{INT: < GET(SP), >} \\ \text{INT: < GET(DEST), GET(ST), GET(ROUTE-TYPE), GIVE B ROUTE(SP,DEST,ST,ROUTE-TYPE) >} \\ \text{DU7A:} \left[\begin{array}{l} \text{OBL: < A ADDRESS 7B >} \\ \text{DH: < 7A: ANSWER, 7B: REASSERT >} \end{array}\right] \end{array}\right] \end{array}\right]$$

Now A's intention to get SP can finally be removed, and A can start processing the next intention, getting a value for DEST.

$$\left[\begin{array}{l} \\ \text{A:} \left[\begin{array}{l} \text{G:} \left[\begin{array}{l} \text{OBL: < A UNDERSTANDING-ACT DU7B, A ADDRESS 3C >} \\ \text{SCP: <B SCP SP = MALVERN, A SCP SP = MALVERN, B WANTS A ROUTE >} \end{array}\right] \\ \text{INT: < ACKNOWLEDGE DU7B, INFO-REQUEST, GET(DEST) >} \\ \text{INT: < GET(ST), GET(ROUTE-TYPE), GIVE B ROUTE(SP,DEST,ST,ROUTE-TYPE) >} \\ \text{DU7B:} \left[\begin{array}{l} \text{OBL: < A ADDRESS 7B >} \\ \text{DH: < 7A: ANSWER, 7B: REASSERT >} \end{array}\right] \end{array}\right] \\ \\ \end{array}\right]$$

U8 [A]: Where would you like to go.

Chapter 6

Coding Tools

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.

6.1 Annotation as scripting

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 Lager (1995). 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 743-750.

```
# range(743-750).

label(q1,"Does J have P's attention?").
label(a2,respond($q1)).
label(p3,"J has P's attention").
# pushRec(p*shared*qud,$q1).
# pushRec(j*shared*qud,$q1).
```

```
# pushRec(j*private*agenda,$a2).
# addRec(p*shared*bel,$p3).
# addRec(j*shared*bel,$p3).
# popRec(p*private*agenda).

# 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 the following example:

```
(2)
$P: Excuse me
```

pushRec(P.SHARED.QUD, '*Does P have J's attention?*')
pushRec(J.SHARED.QUD, '*Does P have J's attention?*')
pushRec(J.PRIVATE.AGENDA, **respond**('*Does P have J's attention?*'))
addRec(P.SHARED.BEL, '*J has P's attention*')
addRec(J.SHARED.BEL, '*J has P's attention*')
popRec(P.PRIVATE.AGENDA)

$$\left[\begin{array}{lll} \text{J} & = & \left[\begin{array}{lll} \text{PRIVATE} & = & \left[\begin{array}{lll} \text{BEL} & = & \{\} \\ \text{AGENDA} & = & \langle\ \textbf{respond}(\text{'}\textit{Does P have J's attention?}\text{'})\ \rangle \end{array}\right] \\ \text{SHARED} & = & \left[\begin{array}{lll} \text{BEL} & = & \{\ \text{'}\textit{J has P's attention}\text{'}\ \} \\ \text{QUD} & = & \langle\ \text{'}\textit{Does P have J's attention?}\text{'}\ \rangle \end{array}\right] \end{array}\right] \\ \text{P} & = & \left[\begin{array}{lll} \text{PRIVATE} & = & \left[\begin{array}{lll} \text{BEL} & = & \left\{\begin{array}{l} \text{'}\textit{P wants to travel by plane}\text{'} \\ \text{'}\textit{P wants to go to Paris}\text{'} \end{array}\right\} \\ \text{AGENDA} & = & \langle\rangle \end{array}\right] \\ \text{SHARED} & = & \left[\begin{array}{lll} \text{BEL} & = & \{\ \text{'}\textit{J has P's attention}\text{'}\ \} \\ \text{QUD} & = & \langle\ \text{'}\textit{Does P have J's attention?}\text{'}\ \rangle \end{array}\right] \end{array}\right] \end{array}\right]$$

6.2 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.

6.2.1 Operations and moves, `update`

Update operations or moves[1] 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.

The update operators are defined in `update_ops.pl`, and so far the following have been defined:

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`

6.2.2 `range`

The `range` command is used to indicate the range of transcription text referred to by the updates below it, until the next `range` command. A range is an expression on the form `X-Y` where `X` and `Y` are file offsets in the transcription ascii file. When annotating, these can be found e.g. by marking the relevant range in the GATE MAT annotator.

6.2.3 `print`

The `print` can be used to print any kind of text. One may want to replace `range` commands by `print`, to save the trouble of finding the proper file offsets. However, this will sever the link between the annotation file and the transcription file. In some cases one may want to keep this link in order to e.g. compare TranScript annotations by different annotators, or relate TranScript annotation to annotation using other schemas and tools.

[1] Not yet implemented.

6.2.4 `label`

As stated above, labels are used to refer to propositional contents, actions etc. in annotation clauses.

6.2.5 `comment`

The `comment` command is used to add free-text comments to updates.

6.2.6 `initial_state`

This command indicates the initial information state, e.g. the state before the first utterance in a dialogue. It also serves to define the type of information state which the annotation assumes.

6.3 Parsing TranScript files

The 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 a table:

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

The operators and moves are defined in separate files and can be supplemented or replaced

with other definitions.

6.4 Generating output

The LaTeXgenerator takes an inflated TranScript file and produces a LaTeXfile, 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.

The script `tr2ps` takes as argument a TranScript file, and produces LaTeXand PostScript output. The option `-s` will hide record fields whose values are empty sets, stacks etc..

6.5 Example

Below is the input file of codings in the Poesio-Traum theory of information states which was used to produce the output shown in Appendix C.

```
% updates for the record version of the PoesioTraum coding
%info state assumed to look like:
%Fields: OBL - list of obligations
%        DH -  list of performed dialogue acts
%        SCP - set of social commitments
%  OPT - set of options
%  INT - set of action types (realy partially ordered set, where
% top is next act to do, but intentions can be dropped
% from other places, and inserted anywhere
% UDUS - list of DUs
% DU type - record with four fields, OBL, DH, SCP, OPT
% G - a DU type record
% other DUs get added to an agent record
% Agent type : record with the following fields:
% G, INT, UDUs, BEL, and  some expandable
% number of DU type fields.
%agents: A,B
%all operations not mentioning a or b at the beginning of a path
%assumed to apply to both a and b.

transcription_file('/users/ling/sl/Jobb/TRINDI/Corpus/autoroute1').

initial_state(record([a=record([g=record([obl=list([]),dh=stack([]),
scp=set([]),opt=set([])]),int=list([]),udus=list([])]),
b=record([g=record([obl=list([]),dh=stack([]),scp=set([]),
opt=set([])]),int=list([]),udus=list([])])]).

shorthand(edu,record([obl=list([]),dh=stack([]),scp=set([]),opt=set([])])).

#range(0-27).

%starting dialogue condition
```

```
label(int1b,"Get a route from malvern to edwinstowe").
#update([pushRec(a*int,greet),pushRec(a*int,offerHelp),
        pushRec(b*int, $int1b)]).

%updates from utt1 & 2

#range(27-114).
#update([removeRec(a*int,greet),removeRec(a*int,offerHelp)]).
#update([add_fieldRec(a*du2, @edu), add_fieldRec(b*du2, @edu),
    pushRec(a*udus, du2), pushRec(b*udus, du2)]).
label(q2, "How can A help B?").
label(i2,info_request(a,b,$q2)).
#update([pushRec(a*du2*dh, greet(a,b)), pushRec(b*du2*dh, greet(a,b)),
    pushRec(a*du2*dh, i2: $i2), pushRec(b*du2*dh,  i2: $i2)]).
#update([pushRec(a*g*obl,understandingAct(b,du2)),
    pushRec(b*g*obl,understandingAct(b,du2)),
    pushRec(a*du2*obl,answer(b, i2: $i2)),
    pushRec(b*du2*obl,answer(b, i2: $i2))]).


%utterance planning for 3
#update([pushRec(b*int,understandingAct(b,du2)),
    pushRec(b*int,answer(b, i2: $i2))]).



#range(114-156).


%updates after 3
#update([add_fieldRec(a*du3, @edu), add_fieldRec(b*du3, @edu),
    pushRec(a*udus, du3), pushRec(b*udus, du3)]).
#update([peRec(a*g,a*du2),peRec(b*g,b*du2),removeRec(a*udus,du2),
    removeRec(b*udus,du2),pushRec(a*g*dh, acknowledge(b,du2)),
    pushRec(b*g*dh, acknowledge(b,du2))]).
label(d3, "Give B a route").
label(di3,direct(b,a,$d3)).
label(a3, "A can help B by providing a route").
label(ans3,answer(b,a,i2,$a3)).
label(cond3, accept(a,di3) --> obliged(a,b,$d3)).
#update([pushRec(a*du3*dh,di3: $di3),pushRec(b*du3*dh,di3: $di3),
    pushRec(a*du3*dh,$ans3),pushRec(b*du3*dh,$ans3)]).
#update([pushRec(a*g*obl,understandingAct(a,du3)),
    pushRec(b*g*obl,understandingAct(a,du3)),
    pushRec(a*du3*obl,address(a,di3: $di3)),
    pushRec(b*du3*obl,address(a,di3: $di3)),
    addRec(a*du3*opt,address(a,di3: $di3)),
    addRec(b*du3*opt,address(a,di3: $di3)),
    pushRec(a*du3*obl,$cond3),pushRec(b*du3*obl,$cond3),
    addRec(a*du3*scp,scp(b,$a3)),addRec(b*du3*scp,scp(b,$a3))]).
#update([removeRec(a*g*obl,understandingAct(b,du2)),
    removeRec(a*g*obl,understandingAct(b,du2)),
    removeRec(b*g*obl,understandingAct(b,du2)),
    removeRec(b*int,understandingAct(b,du2)),
    removeRec(b*int,answer(b,i2: $i2))]).

%utterance planning leading to 4
label(ii3, "give B route(sp,dest,st,routetype)").
#update([removeRec(a*int, address(a,di3: $di3)),pushRec(a*int,$ii3),
    pushRec(a*int,get(routetype)), pushRec(a*int,get(st)),
    pushRec(a*int,get(dest)),pushRec(a*int,get(sp)),
    pushRec(a*int, understandingAct(a,du3)),pushRec(a*int, accept(di3: $di3)),
    pushRec(a*int, address(a,di3: $di3))]).


%utterance 4
```

```
#print("A <Where would you like to start your journey?>").
#add_fieldRec(a*du4, @edu).
#add_fieldRec(b*du4, @edu).
#pushRec(a*udus, du4).
#pushRec(b*udus, du4).

#peRec(a*g,a*du3).
#peRec(b*g,b*du3).
#removeRec(a*udus,du3).
#removeRec(b*udus,du3).
#pushRec(a*g*dh, acknowledge(a,du3)).
#pushRec(b*g*dh, acknowledge(a,du3)).

label(q4, "Where does B want to start?").
label(qu4,info_request(a,b,$q4)).
#pushRec(a*du4*dh,qu4: $qu4).
#pushRec(b*du4*dh,qu4: $qu4).

#pushRec(a*g*obl,understandingAct(b,du4)).
#pushRec(b*g*obl,understandingAct(b,du4)).
#pushRec(a*du4*obl,answer(b,qu4)).
#pushRec(b*du4*obl,answer(b,qu4)).

#removeRec(a*g*obl,understandingAct(a,du3)).
#removeRec(b*g*obl,understandingAct(a,du3)).
#removeRec(a*g*obl,answer(b,i2: $i2)).
#removeRec(b*g*obl,answer(b,i2: $i2)).
#removeRec(a*g*obl,address(a,di3: $di3)).
#removeRec(b*g*obl,address(a,di3: $di3)).
#removeRec(a*g*obl,$cond3).
#removeRec(b*g*obl,$cond3).
#pushRec(a*g*obl,obliged(a,b,$d3)).
#pushRec(b*g*obl,obliged(a,b,$d3)).

#pushRec(b*int,understandingAct(b,du4)).
#pushRec(b*int,answer(b,qu4)).
#removeRec(a*int,understandingAct(a,du3)).
#removeRec(a*int,address(a,di3: $di3)).
#removeRec(a*int, accept(di3: $di3)).

#print_state.

#print("B <Malvern.>").
#add_fieldRec(a*du5, @edu).
#add_fieldRec(b*du5, @edu).
#pushRec(a*udus, du5).
#pushRec(b*udus, du5).
#pushRec(a*g*obl,understandingAct(a,du5)).
#pushRec(b*g*obl,understandingAct(a,du5)).

#peRec(a*g,a*du4).
#peRec(b*g,b*du4).
#removeRec(a*udus,du4).
#removeRec(b*udus,du4).
#pushRec(a*g*dh, acknowledge(b,du4)).
#pushRec(b*g*dh, acknowledge(b,du4)).

label(a5, "SP=Malvern").
label(ans5,answer(b,a,$q4,$a5)).
#pushRec(a*du5*dh,$ans5).
#pushRec(b*du5*dh,$ans5).
#addRec(a*du5*scp,scp(b,$a5)).
#addRec(b*du5*scp,scp(b,$a5)).
#removeRec(a*g*obl,understandingAct(b,du4)).
#removeRec(b*g*obl,understandingAct(b,du4)).
```

```
#removeRec(b*int,understandingAct(b,du4)).
#removeRec(b*int,answer(b,qu4)).

#pushRec(a*int, understandingAct(a,du5)).
#pushRec(a*int, check($ans5)).

#print_state.

#print("A <Starting in Great Malvern.>").
#add_fieldRec(a*du6, @edu).
#add_fieldRec(b*du6, @edu).
#pushRec(a*udus, du6).
#pushRec(b*udus, du6).
#pushRec(a*g*obl,understandingAct(b,du6)).
#pushRec(b*g*obl,understandingAct(b,du6)).

#peRec(a*g,a*du5).
#peRec(b*g,b*du5).
#removeRec(a*udus,du5).
#removeRec(b*udus,du5).
#pushRec(a*g*dh, acknowledge(b,du5)).
#pushRec(b*g*dh, acknowledge(b,du5)).

label(c6, "SP= Great Malvern").
label(ch6,check(a,b,$c6)).
#pushRec(a*du6*dh,ch6: $ch6).
#pushRec(b*du6*dh,ch6: $ch6).
#pushRec(a*du6*obl,answer(b,ch6)).
#pushRec(b*du6*obl,answer(b,ch6)).
label(cond6, agree(b,ch6) --> scp(a,$c6)).
#addRec(a*du6*scp,$cond6).
#addRec(b*du6*scp,$cond6).

#removeRec(a*g*obl,understandingAct(a,du5)).
#removeRec(b*g*obl,understandingAct(a,du5)).
#removeRec(a*g*obl,answer(b,qu4)).
#removeRec(b*g*obl,answer(b,qu4)).

#removeRec(a*int, understandingAct(a,du5)).
#removeRec(a*int, check($ans5)).

#pushRec(b*int,understandingAct(b,du6)).
#pushRec(b*int,agree(b,ch6)).

#print_state.

#print("B <Yes.>").

#add_fieldRec(a*du7, @edu).
#add_fieldRec(b*du7, @edu).
#pushRec(a*udus, du7).
#pushRec(b*udus, du7).
#pushRec(a*g*obl,understandingAct(a,du7)).
#pushRec(b*g*obl,understandingAct(a,du7)).

#peRec(a*g,a*du6).
#peRec(b*g,b*du6).
#removeRec(a*udus,du6).
#removeRec(b*udus,du6).
#pushRec(a*g*dh, acknowledge(b,du6)).
#pushRec(b*g*dh, acknowledge(b,du6)).
label(ag7,agree(b,a,ch6)).
#pushRec(a*du7*dh,$ag7).
#pushRec(b*du7*dh,$ag7).

#pushRec(b*du7*dh,$ag7).
```

```
#addRec(a*du7*scp,scp(b,$c6)).
#addRec(b*du7*scp,scp(b,$c6)).
#removeRec(a*g*obl,understandingAct(b,du6)).
#removeRec(b*g*obl,understandingAct(b,du6)).

%#removeRec(a*g*obl,answer(b,ch6)).
%#removeRec(b*g*obl,answer(b,ch6)).
%#removeRec(a*g*scp,$cond6).
%#removeRec(b*g*scp,$cond6).
%#pushRec(a*g*scp, scp(a,b,$c6)).
%#pushRec(b*g*scp, scp(a,b,$c6)).


#removeRec(a*int,get(sp)).
#print_state.
```

6.6 Using the GATE system for annotation

As an alternative to using TranScript, one can annotate for information state updates using the MAT annotator in the GATE system Gaizauskas (1998). Given a script for translating GATE annotations to the ITR format[2], one can then use the same output generators as for TranScript coding. Two MAT annotation schemes have been designed for annotating updates to information states. The **infostate** scheme has the following attributes:

- **Participant**: The participant whose information state is updated.
- **Operation**: This is the type of operation to be performed, e.g. `push`, `pop` (for stacks), `add` and `delete` (for sets).
- **Field**: The fields are shorthand names for paths in the information state record, such as `qud` (for *common.qud*), `agenda` (for *private.agenda*) etc.
- **Content**: The value of this attribute is a reference to an annotation produced by the **label** scheme. Contents are currently sentences of natural language. Eventually, one might want want to complement this with a more formal representation of content.
- **Action**: This attribute is used only for pushes to the agenda, as in `push(A.private.agenda, raise(label-12))`. The value of this attribute is a reference to an annotation produced by the **label** scheme.
- **Order** is a natural number indicating when an update is to be performed in relation to other updates caused by a single utterance (segment). It is used in cases where a single segment is annotated with several order-dependent updates. For example, if an utterance is annotated with several `push`es to `A.shared.qud`,the resulting information state depends on the order in which these are executed.

[2]This script is under development at the time of writing.

The **label** scheme is used to annotate the dialogue transcription with e.g. (natural language) paraphrases of the contents. These paraphrases (as all annotations) are assigned indexes, which can then be used as values of the **label** attribute of the **infostate** scheme. The scheme can also be used to annotate actions, which consist of an action type and a content. The action types are `raise`, `respond` and `instruct`.

For example, part of the annotation shown in Section 6.1 might look like this when produced by GATE:

```
ID      TYPE            START   END     ATTRIBUTES
19      label                           (string:Does P have J's attention?)
20      label                           (action:respond) (content:19)
23      label                           (string:P has J's attention)
37      infostate       743     750     (content:19) (field:qud)
                                        (operation:push) (participant:J)
37      infostate       743     750     (content:19) (field:qud)
                                        (operation:push) (participant:P)
40      infostate       743     750     (operation:push) (participant:J)
                                        (field:agenda) (action:20)
...
```

(Note that the contents annotation has no specific range.) The number 19 in the **content** fields are references to the ID number of the **label** annotation.

Chapter 7

Discussion

The methodology we are developing, as described in this report has several potential uses, including,

- empirical application of theories of dialogue semantics
- fine-tuning detailed aspects of these theories
- detailed comparison of the mechanics of seemingly different theories of dialogue
- a testbed for dialogue system prototyping

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. Second, doing coding like this forces one to address certain detailed questions about the theories, e.g., whether an *answer* act or an obligation to answer contains as an argument a *question* as a linguistic object (e.g., as might be put on the QUD), or an *info-request* as the dialogue act which introduces the question. Settling this issue has implications for the resulting information state after a repair or re-interpretation of the act. A related issue is how much of the previous information state is it necessary to keep available, especially for cases of repair? Issues such as these that we have encountered in developing these coding schemes for information state will be considered in the next phase of the project, in developing more formal and broader coverage versions of these theories of information state.

Also, the kind of annotation of information state updates is not suitable for large-scale annotation: both because it is time consuming, and tedious to get the details correct, even when the rules are well specified. The tools described in Chapter 6 definitely make coding

and analysis much easier than doing it by hand, but do not go far enough. Coding with "moves" representing stereotyped sequences of updates is also helpful, however, as described and illustrated in Chapter 4, the connection between the type of moves that are useful for updating information states and the moves used in dialogue move annotation often diverge. What is required instead are the kinds of update rules which are sensitive not just to the moves themselves, but other aspects of the information state.

Moving in the direction of more sophisticated tools for automatically updating information states on the basis of pre-defined rules also moves in the direction of implementing a dialogue manager for task oriented dialogue. In the next phase of the project we will also explore this connection, developing rules which can be used equally as part of a "dialogue move engine" for allowing a system to engage in dialogue as well as being part of a tool for semi-automated dialogue annotation.

Bibliography

Ahrenberg, L., Dahlbäck, N., and Jönsson, A. (1995). Coding Schemes for Studies of Natural Language Dialogue. In *Working Notes from AAAI Spring Symposium*, Stanford.

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

Allen, J. and Core, M. (1997). DAMSL: Dialogue act markup in several layers. Draft contribution for the Discourse Resource Initiative.

Allwood, J., Nivre, J. and Ahlsn, E. (1994). Semantics and Spoken Language: Manual for Coding Interaction Management. Report from the HSFR project Semantik och talsprk.

Austin, J. L. (1962). *How to Do Things with Words.* Harvard University Press, Cambridge, MA.

Bunt, H. C. (1995). Dialogue control functions and interaction design. In R. Beun, M. Baker, and M. Reiner, editors, *Dialogue in Instruction*, pages 197–214. Springer Verlag.

Carletta, J., Isard, A., Isard, S., Kowtko, J., Doherty-Sneddon, G., and Anderson, A. H. (1997). The reliability of a dialogue structure coding scheme. *Computational Linguistics*, **23**(1), 13–32.

Carlson, L. (1983). *Dialogue Games.* D. Reidel, Dordrecht.

Clark, H. H. and Schaefer, E. F. (1989). Contributing to discourse. *Cognitive Science*, **13**, 259 – 94.

Cooper, Robin (1998a). Mixing Situation Theory and Type Theory to Formalize Information States in Dialogue Exchanges. In *Proceedings of TWLT 13/Twendial '98: Formal Semantics and Pragmatics of Dialogue.* Also available as GPCL 98-2 at `http://www.ling.gu.se/publications/GPCL.html`.

Cooper, Robin (1998b). Information States, Attitudes and Dialogue, In *Proceedings of ITALLC-98*. Also available as GPCL 98-5 at `http://www.ling.gu.se/publications/GPCL.html`.

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

Core, M. G. and Allen, J. F. (1997). Coding dialogs with the DAMSL scheme. In *Working Notes of the AAAI Fall Symposium on Communicative Action in Humans and Machines*, Boston, MA. AAAI.

Davidson, D. (1967). The logical form of action sentences. In N. Rescher, editor, *The Logic of Decision and Action*, pages 81–95. University of Pittsburgh Press, Pittsburgh.

Di Eugenio, B., Jordan, P. W., Thomason, R. T., and Moore, J. D. (1997). Reconstructed intentions in collaborative problem solving dialogues. In *Working Notes of the AAAI Fall Symposium on Communicative Action in Humans and Machines*, Boston, MA. AAAI.

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

Gaizauskas, R. e. a. (1998). *GATE User Guide*. Institute for Language, Speech and Hearing (ILASH) , and the Department of Computer Science , University of Sheffield, UK, 1.5.0-1 edition.

Ginzburg, J. (1995a). Resolving questions, i. *Linguistics and Philosophy*, **18**(5), 567–609.

Ginzburg, J. (1995b). Resolving questions, ii. *Linguistics and Philosophy*, **18**(6), 567–609.

Ginzburg, J. (1996). Dynamics and the semantics of dialogue. In J. Seligman and D. Westerståhl, editors, *Logic, Language and Computation, Vol. 1*, volume 1. CSLI Publications.

Ginzburg, J. (1997). On some semantic consequences of turn-taking. In A. Benz and G. Jäger, editors, *Proc. of the Munich Workshop on Formal Semantics and Pragmatics of Dialogue*. University of Munich.

Ginzburg, J. (1998). 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. Universiteit Twente, Faculteit Informatica.

Goldman, A. (1970). *A Theory of Human Action*. Princeton University Press, Princeton, NJ.

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

Kamp, H. and Reyle, U. (1993). *From Discourse to Logic*. D. Reidel, Dordrecht.

Kowtko, J. C., Isard, S. D., and Doherty, G. M. (1992). Conversational games within dialogue. Research Paper HCRC/RP-31, Human Communication Research Centre.

Lager, T. (1995). *A Logical Approach to Computational Corpus Linguistics*. Ph.D. thesis, Detp. of Linguistics, Göteborgh Universtity.

Levin, J. A. and Moore, J. A. (1978). Dialogue games: Metacommunication strategies for natural language interaction. *Cognitive Science*, **1**(4), 395–420.

Muskens, R. (1995). Tense and the logic of change. In U. Egli, P. Pause, C. Schwarze, A. von Stechow, and G. Wienold, editors, *Lexical Knowledge in the Organization of Language*, pages 147–183. John Benjamins, Amsterdam / Philadelphia.

Novick, D. (1988). *Control of Mixed-Initiative Discourse Through Meta-Locutionary Acts: A Computational Model.* Ph.D. thesis, University of Oregon. also available as U. Oregon Computer and Information Science Tech Report CIS-TR-88-18.

Poesio, M. (1998). Utterance processing and semantic underspecification. Submitted.

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

Poesio, M. and Traum, D. (1998). 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. Universiteit Twente, Faculteit Informatica.

Traum, D. R. (1994). *A Computational Theory of Grounding in Natural Language Conversation.* Ph.D. thesis, University of Rochester, Department of Computer Science, Rochester, NY.

Traum, D. R. and Allen, J. F. (1994). Discourse obligations in dialogue processing. In *Proc. of the 32nd Annual Meeting of the Association for Computational Linguistics*, pages 1–8, New Mexico.

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

Appendix A

An Autoroute Dialogue

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.

Appendix B

Appendix: Full annotation of the Autoroute Dialogue in Scheme 1

(1)
Dialogue 127
%—start—-%

push(a.private.agenda, raise(*What does B want?*))
push(a.private.agenda, respond(*Does B have A's attention?*))
add(a.shared.bel, *A has B's attention*)
push(a.shared.qud, *Does B have A's attention?*)
add(b.shared.bel, *B wants assistance*)
add(b.shared.bel, *A has B's attention*)
push(b.shared.qud, *Does B have A's attention?*)

$$\left[\begin{array}{lll} \text{a} & = & \left[\begin{array}{lll} \text{private} & = & \left[\begin{array}{lll} \text{bel} & = & \{\} \\ \text{agenda} & = & \left\langle \begin{array}{l} \text{respond}(\textit{Does B have A's attention?}\ ) \\ \text{raise}(\textit{What does B want?}\ ) \end{array} \right\rangle \end{array}\right] \\ \text{shared} & = & \left[\begin{array}{lll} \text{bel} & = & \{\ \textit{A has B's attention}\ \} \\ \text{qud} & = & \langle\ \textit{Does B have A's attention?}\ \rangle \end{array}\right] \end{array}\right] \\ \text{b} & = & \left[\begin{array}{lll} \text{private} & = & \left[\begin{array}{lll} \text{bel} & = & \{\} \\ \text{agenda} & = & \langle\rangle \end{array}\right] \\ \text{shared} & = & \left[\begin{array}{lll} \text{bel} & = & \left\{ \begin{array}{l} \textit{A has B's attention} \\ \textit{B wants assistance} \end{array} \right\} \\ \text{qud} & = & \langle\ \textit{Does B have A's attention?}\ \rangle \end{array}\right] \end{array}\right] \end{array}\right]$$

(2)

A <Welcome to the Route Planning Service.>

pop(a.shared.qud)
add(a.shared.bel, *B has A's attention*)

add(b.shared.bel, *B has A's attention*)
pop(b.shared.qud)
pop(a.private.agenda)

$$\left[\begin{array}{lll} & & \left[\begin{array}{lll} \text{private} & = & \left[\begin{array}{lll} \text{bel} & = & \{\} \\ \text{agenda} & = & \langle\ \text{raise}(\textit{What does B want?})\ \rangle \end{array}\right] \\ \text{shared} & = & \left[\begin{array}{lll} \text{bel} & = & \left\{\begin{array}{l} \textit{B has A's attention} \\ \textit{A has B's attention} \end{array}\right\} \\ \text{qud} & = & \langle\rangle \end{array}\right] \end{array}\right] \\ \text{a} & = & \\ & & \left[\begin{array}{lll} \text{private} & = & \left[\begin{array}{lll} \text{bel} & = & \{\} \\ \text{agenda} & = & \langle\rangle \end{array}\right] \\ \text{shared} & = & \left[\begin{array}{lll} \text{bel} & = & \left\{\begin{array}{l} \textit{B has A's attention} \\ \textit{A has B's attention} \\ \textit{B wants assistance} \end{array}\right\} \\ \text{qud} & = & \langle\rangle \end{array}\right] \end{array}\right] \\ \text{b} & = & \end{array}\right]$$

(3)
<How can I help you.>

pop(a.private.agenda)
push(a.shared.qud, *How can A help B?*)
push(b.private.agenda, respond(*What does B want from A?*))
push(b.shared.qud, *How can A help B?*)
push(b.shared.qud, *What does B want from A?*)

$$\left[\begin{array}{lll} & & \left[\begin{array}{lll} \text{private} & = & \left[\begin{array}{lll} \text{bel} & = & \{\} \\ \text{agenda} & = & \langle\rangle \end{array}\right] \\ \text{shared} & = & \left[\begin{array}{lll} \text{bel} & = & \left\{\begin{array}{l} \textit{B has A's attention} \\ \textit{A has B's attention} \end{array}\right\} \\ \text{qud} & = & \langle\ \textit{How can A help B?}\ \rangle \end{array}\right] \end{array}\right] \\ \text{a} & = & \\ & & \left[\begin{array}{lll} \text{private} & = & \left[\begin{array}{lll} \text{bel} & = & \{\} \\ \text{agenda} & = & \langle\ \text{respond}(\textit{What does B want from A?})\ \rangle \end{array}\right] \\ \text{shared} & = & \left[\begin{array}{lll} \text{bel} & = & \left\{\begin{array}{l} \textit{B has A's attention} \\ \textit{A has B's attention} \\ \textit{B wants assistance} \end{array}\right\} \\ \text{qud} & = & \left\langle\begin{array}{l} \textit{What does B want from A?} \\ \textit{How can A help B?} \end{array}\right\rangle \end{array}\right] \end{array}\right] \\ \text{b} & = & \end{array}\right]$$

(4)

(Pause: 1)
B <A route please.>

pop(b.private.agenda)
pop(b.shared.qud)

pop(b.shared.qud)
push(a.shared.qud, *What does B want from A?*)
add(b.shared.bel, *B wants a route from A*)

$$\left[\begin{array}{lll} & & \left[\begin{array}{lll} \text{private} & = & \left[\begin{array}{lll} \text{bel} & = & \{\} \\ \text{agenda} & = & \langle\rangle \end{array}\right] \\ \text{shared} & = & \left[\begin{array}{lll} \text{bel} & = & \left\{\begin{array}{l} \textit{B has A's attention} \\ \textit{A has B's attention} \end{array}\right\} \\ \text{qud} & = & \left\langle\begin{array}{l} \textit{What does B want from A?} \\ \textit{How can A help B?} \end{array}\right\rangle \end{array}\right] \end{array}\right] \\ \text{a} & = & \\ & & \left[\begin{array}{lll} \text{private} & = & \left[\begin{array}{lll} \text{bel} & = & \{\} \\ \text{agenda} & = & \langle\rangle \end{array}\right] \\ \text{shared} & = & \left[\begin{array}{lll} \text{bel} & = & \left\{\begin{array}{l} \textit{B wants a route from A} \\ \textit{B has A's attention} \\ \textit{A has B's attention} \\ \textit{B wants assistance} \end{array}\right\} \\ \text{qud} & = & \langle\rangle \end{array}\right] \end{array}\right] \\ \text{b} & = & \end{array}\right]$$

(5)

pop(a.shared.qud)
pop(a.shared.qud)
push(a.private.agenda, raise(*Does B want the quickest or shortest route?*))
push(a.private.agenda, raise(*What time does B want to make the journey?*))
push(a.private.agenda, raise(*Where does B want to go?*))
push(a.private.agenda, raise(*Where does B want to start?*))

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

(6)
(Pause: 1)
A <Where would you like to start your journey.>

pop(a.private.agenda)
push(a.shared.qud, *Where does B want to start?*)
push(b.private.agenda, respond(*Where does B want to start?*))
push(b.shared.qud, *Where does B want to start?*)

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

(7)

B <Malvern.>

push(a.private.agenda, raise(*Does B want to start the journey in great Malvern?*))
pop(b.private.agenda)
add(b.shared.bel, *B wants to start the journey in Malvern*)
pop(b.shared.qud)

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

(8)

A <Starting in Great Malvern.>

pop(a.private.agenda)
push(a.shared.qud, *Does B want to start the journey in great Malvern?*)
push(b.shared.qud, *Where does B want to start?*)
delete(b.shared.bel, *B wants to start the journey in Malvern*)
push(b.shared.qud, *Does B want to start the journey in great Malvern?*)
push(b.private.agenda, respond(*Does B want to start the journey in great Malvern?*))

$$\left[\begin{array}{lll} & & \left[\begin{array}{lll} \text{bel} & = & \{\} \\ \text{agenda} & = & \left\langle \begin{array}{l} \text{raise}(\textit{Where does B want to go?}) \\ \text{raise}(\textit{What time does B want to make the journey?}) \\ \text{raise}(\textit{Does B want the quickest or shortest route?}) \end{array} \right\rangle \end{array}\right] \\ \text{a} = & \text{private} = & \\ & \text{shared} = & \left[\begin{array}{lll} \text{bel} & = & \left\{ \begin{array}{l} \textit{B has A's attention} \\ \textit{A has B's attention} \end{array} \right\} \\ \text{qud} & = & \left\langle \begin{array}{l} \textit{Does B want to start the journey in great Malvern?} \\ \textit{Where does B want to start?} \end{array} \right\rangle \end{array}\right] \end{array}\right]$$

$$\left[\begin{array}{lll} & & \left[\begin{array}{lll} \text{bel} & = & \{\} \\ \text{agenda} & = & \left\langle \text{respond}(\textit{Does B want to start the journey in great Malvern?}) \right\rangle \end{array}\right] \\ \text{b} = & \text{private} = & \\ & \text{shared} = & \left[\begin{array}{lll} \text{bel} & = & \left\{ \begin{array}{l} \textit{B wants a route from A} \\ \textit{B has A's attention} \\ \textit{A has B's attention} \\ \textit{B wants assistance} \end{array} \right\} \\ \text{qud} & = & \left\langle \begin{array}{l} \textit{Does B want to start the journey in great Malvern?} \\ \textit{Where does B want to start?} \end{array} \right\rangle \end{array}\right] \end{array}\right]$$

(9)

(Pause: 1)
B <Yes.>

pop(a.shared.qud)
pop(a.shared.qud)
pop(b.private.agenda)
pop(b.shared.qud)
pop(b.shared.qud)
add(a.shared.bel, *B wants to start the journey in Great Malvern*)
add(b.shared.bel, *B wants to start the journey in Great Malvern*)

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

(10)

A <Where would you like to go.>

pop(a.private.agenda)
push(a.shared.qud, *Where does B want to go?*)
push(b.private.agenda, respond(*Where does B want to go?*))
push(b.shared.qud, *Where does B want to go?*)

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

(11)

B <Edwinstowe.>

push(a.private.agenda, raise(*Does B want to go to Edwinstowe?*))
pop(b.private.agenda)

add(b.shared.bel, *B wants to go to Edwinstowe*)
pop(b.shared.qud)

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

(12)

(Pause: 1)
A <Edwinstowe.>

pop(a.private.agenda)
push(a.shared.qud, *Does B want to go to Edwinstowe?*)
delete(b.shared.bel, *B wants to go to Edwinstowe*)
push(b.private.agenda, respond(*Does B want to go to Edwinstowe?*))
push(b.shared.qud, *Where does B want to go?*)
push(b.shared.qud, *Does B want to go to Edwinstowe?*)

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

(13)

B <Yes.>

push(a.private.agenda, instruct(*B waits*))
pop(a.shared.qud)
pop(b.private.agenda)
add(b.shared.bel, *B wants to go to Edwinstowe*)
pop(b.shared.qud)
pop(b.shared.qud)

$$
\left[\begin{array}{lll}
 & & \left[\begin{array}{lll}
\text{private} & = & \left[\begin{array}{lll}
\text{bel} & = & \{\} \\
\text{agenda} & = & \left\langle \begin{array}{l} \text{instruct}(\textit{B waits}\ ) \\ \text{raise}(\textit{What time does B want to make the journey?}\ ) \\ \text{raise}(\textit{Does B want the quickest or shortest route?}\ ) \end{array} \right\rangle
\end{array}\right] \\
\text{shared} & = & \left[\begin{array}{lll}
\text{bel} & = & \left\{ \begin{array}{l} \textit{B wants to start the journey in Great Malvern} \\ \textit{B has A's attention} \\ \textit{A has B's attention} \end{array} \right\} \\
\text{qud} & = & \left\langle \textit{Where does B want to go?} \right\rangle
\end{array}\right]
\end{array}\right] \\
\text{a} = & & \\
 & & \left[\begin{array}{lll}
\text{private} & = & \left[\begin{array}{lll}
\text{bel} & = & \{\} \\
\text{agenda} & = & \langle\rangle
\end{array}\right] \\
\text{shared} & = & \left[\begin{array}{lll}
\text{bel} & = & \left\{ \begin{array}{l} \textit{B wants to go to Edwinstowe} \\ \textit{B wants to start the journey in Great Malvern} \\ \textit{B wants a route from A} \\ \textit{B has A's attention} \\ \textit{A has B's attention} \\ \textit{B wants assistance} \end{array} \right\} \\
\text{qud} & = & \langle\rangle
\end{array}\right]
\end{array}\right]
\end{array}\right]
$$

(14)

A <Please wait.>

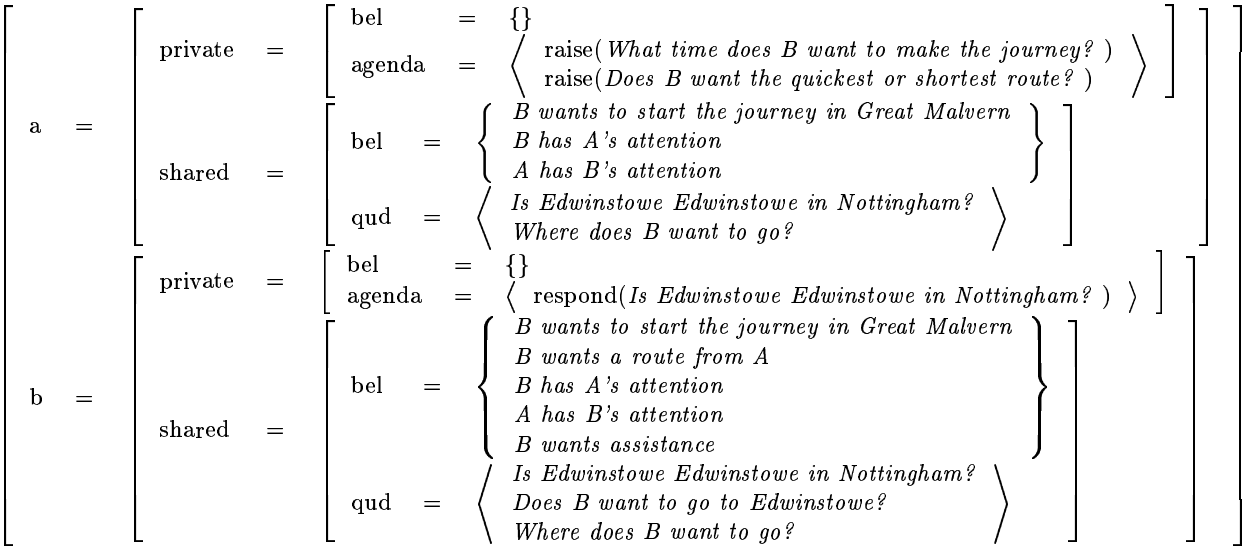
pop(a.private.agenda)
push(a.private.agenda, raise(*Is Edwinstowe Edwinstowe in Nottingham?*))

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

(15)

(Pause: 5)
<Is that Edwinstowe in Nottingham.>

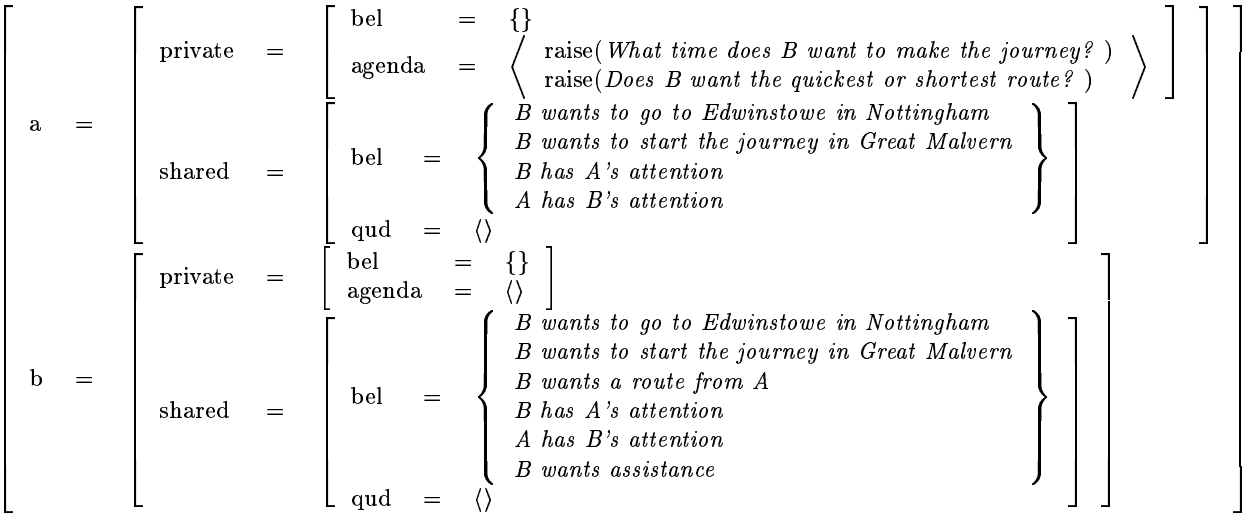
pop(a.private.agenda)
push(a.shared.qud, *Is Edwinstowe Edwinstowe in Nottingham?*)
delete(b.shared.bel, *B wants to go to Edwinstowe*)
push(b.shared.qud, *Where does B want to go?*)
push(b.shared.qud, *Does B want to go to Edwinstowe?*)
push(b.private.agenda, respond(*Is Edwinstowe Edwinstowe in Nottingham?*))
push(b.shared.qud, *Is Edwinstowe Edwinstowe in Nottingham?*)



(16)

B <Yes.>

add(a.shared.bel, *B wants to go to Edwinstowe in Nottingham*)
pop(a.shared.qud)
pop(a.shared.qud)
pop(b.private.agenda)
add(b.shared.bel, *B wants to go to Edwinstowe in Nottingham*)
pop(b.shared.qud)
pop(b.shared.qud)
pop(b.shared.qud)



(17)

(Pause: 2)
A <What time would you like to make your journey.>

push(a.shared.qud, *What time does B want to make the journey?*)
pop(a.private.agenda)
push(b.shared.qud, *What time does B want to make the journey?*)
push(b.private.agenda, respond(*What time does B want to make the journey?*))

$$\left[\begin{array}{lll} & & \left[\begin{array}{lll} \text{private} & = & \left[\begin{array}{lll} \text{bel} & = & \{\} \\ \text{agenda} & = & \langle\ \text{raise}(\textit{Does B want the quickest or shortest route?}\ )\ \rangle \end{array}\right] \\ \text{shared} & = & \left[\begin{array}{lll} \text{bel} & = & \left\{\begin{array}{l} \textit{B wants to go to Edwinstowe in Nottingham} \\ \textit{B wants to start the journey in Great Malvern} \\ \textit{B has A's attention} \\ \textit{A has B's attention} \end{array}\right\} \\ \text{qud} & = & \langle\ \textit{What time does B want to make the journey?}\ \rangle \end{array}\right] \end{array}\right] \\ \text{a} = & & \\ & & \left[\begin{array}{lll} \text{private} & = & \left[\begin{array}{lll} \text{bel} & = & \{\} \\ \text{agenda} & = & \langle\ \text{respond}(\textit{What time does B want to make the journey?}\ )\ \rangle \end{array}\right] \\ \text{shared} & = & \left[\begin{array}{lll} \text{bel} & = & \left\{\begin{array}{l} \textit{B wants to go to Edwinstowe in Nottingham} \\ \textit{B wants to start the journey in Great Malvern} \\ \textit{B wants a route from A} \\ \textit{B has A's attention} \\ \textit{A has B's attention} \\ \textit{B wants assistance} \end{array}\right\} \\ \text{qud} & = & \langle\ \textit{What time does B want to make the journey?}\ \rangle \end{array}\right] \end{array}\right] \\ \text{b} = & & \end{array}\right]$$

(18)

B <Six p.m.>

push(a.private.agenda, raise(*Does B want to leave at 6 p.m.?*))
pop(b.private.agenda)
add(b.shared.bel, *B wants to make the journey at 6 p.m.*)
pop(b.shared.qud)

$$
\left[\begin{array}{lll}
 & & \left[\begin{array}{lll}
\text{private} & = & \left[\begin{array}{lll}
\text{bel} & = & \{\} \\
\text{agenda} & = & \left\langle \begin{array}{l} \text{raise}(\textit{Does B want to leave at 6 p.m.?}) \\ \text{raise}(\textit{Does B want the quickest or shortest route?}) \end{array} \right\rangle
\end{array}\right] \\
\text{shared} & = & \left[\begin{array}{lll}
\text{bel} & = & \left\{ \begin{array}{l} \textit{B wants to go to Edwinstowe in Nottingham} \\ \textit{B wants to start the journey in Great Malvern} \\ \textit{B has A's attention} \\ \textit{A has B's attention} \end{array} \right\} \\
\text{qud} & = & \left\langle \textit{What time does B want to make the journey?} \right\rangle
\end{array}\right]
\end{array}\right] \\
\text{a} = & & \\
 & & \left[\begin{array}{lll}
\text{private} & = & \left[\begin{array}{lll}
\text{bel} & = & \{\} \\
\text{agenda} & = & \langle\rangle
\end{array}\right] \\
\text{shared} & = & \left[\begin{array}{lll}
\text{bel} & = & \left\{ \begin{array}{l} \textit{B wants to make the journey at 6 p.m.} \\ \textit{B wants to go to Edwinstowe in Nottingham} \\ \textit{B wants to start the journey in Great Malvern} \\ \textit{B wants a route from A} \\ \textit{B has A's attention} \\ \textit{A has B's attention} \\ \textit{B wants assistance} \end{array} \right\} \\
\text{qud} & = & \langle\rangle
\end{array}\right]
\end{array}\right] \\
\text{b} = & &
\end{array}\right]
$$

(19)

(Pause: 1)
A <Leaving at six p.m.>

pop(a.private.agenda)
push(a.shared.qud, *Does B want to leave at 6 p.m.?*)
delete(b.shared.bel, *B wants to make the journey at 6 p.m.*)
push(b.shared.qud, *What time does B want to make the journey?*)
push(b.shared.qud, *Does B want to leave at 6 p.m.?*)
push(b.private.agenda, respond(*Does B want to leave at 6 p.m.?*))

$$
\left[\begin{array}{lll}
 & & \left[\begin{array}{lll}
\text{private} & = & \left[\begin{array}{lll}
\text{bel} & = & \{\} \\
\text{agenda} & = & \left\langle \text{raise}(\textit{Does B want the quickest or shortest route?}) \right\rangle
\end{array}\right] \\
\text{shared} & = & \left[\begin{array}{lll}
\text{bel} & = & \left\{ \begin{array}{l} \textit{B wants to go to Edwinstowe in Nottingham} \\ \textit{B wants to start the journey in Great Malvern} \\ \textit{B has A's attention} \\ \textit{A has B's attention} \end{array} \right\} \\
\text{qud} & = & \left\langle \begin{array}{l} \textit{Does B want to leave at 6 p.m.?} \\ \textit{What time does B want to make the journey?} \end{array} \right\rangle
\end{array}\right]
\end{array}\right] \\
\text{a} = & & \\
 & & \left[\begin{array}{lll}
\text{private} & = & \left[\begin{array}{lll}
\text{bel} & = & \{\} \\
\text{agenda} & = & \left\langle \text{respond}(\textit{Does B want to leave at 6 p.m.?}) \right\rangle
\end{array}\right] \\
\text{shared} & = & \left[\begin{array}{lll}
\text{bel} & = & \left\{ \begin{array}{l} \textit{B wants to go to Edwinstowe in Nottingham} \\ \textit{B wants to start the journey in Great Malvern} \\ \textit{B wants a route from A} \\ \textit{B has A's attention} \\ \textit{A has B's attention} \\ \textit{B wants assistance} \end{array} \right\} \\
\text{qud} & = & \left\langle \begin{array}{l} \textit{Does B want to leave at 6 p.m.?} \\ \textit{What time does B want to make the journey?} \end{array} \right\rangle
\end{array}\right]
\end{array}\right] \\
\text{b} = & &
\end{array}\right]
$$

(20)

B <Yes.>

add(a.shared.bel, *B wants to make the journey at 6 p.m.*)
pop(a.shared.qud)
pop(a.shared.qud)
pop(b.private.agenda)
add(b.shared.bel, *B wants to make the journey at 6 p.m.*)
pop(b.shared.qud)
pop(b.shared.qud)

$$
\left[\begin{array}{ll}
\text{a} = & \left[\begin{array}{ll}
\text{private} = & \left[\begin{array}{ll}
\text{bel} = & \{\} \\
\text{agenda} = & \langle\ \text{raise}(\textit{Does B want the quickest or shortest route?}\ )\ \rangle
\end{array}\right] \\
\text{shared} = & \left[\begin{array}{ll}
\text{bel} = & \left\{\begin{array}{l}
\textit{B wants to make the journey at 6 p.m.} \\
\textit{B wants to go to Edwinstowe in Nottingham} \\
\textit{B wants to start the journey in Great Malvern} \\
\textit{B has A's attention} \\
\textit{A has B's attention}
\end{array}\right\} \\
\text{qud} = & \langle\rangle
\end{array}\right]
\end{array}\right] \\
\text{b} = & \left[\begin{array}{ll}
\text{private} = & \left[\begin{array}{ll}
\text{bel} = & \{\} \\
\text{agenda} = & \langle\rangle
\end{array}\right] \\
\text{shared} = & \left[\begin{array}{ll}
\text{bel} = & \left\{\begin{array}{l}
\textit{B wants to make the journey at 6 p.m.} \\
\textit{B wants to go to Edwinstowe in Nottingham} \\
\textit{B wants to start the journey in Great Malvern} \\
\textit{B wants a route from A} \\
\textit{B has A's attention} \\
\textit{A has B's attention} \\
\textit{B wants assistance}
\end{array}\right\} \\
\text{qud} = & \langle\rangle
\end{array}\right]
\end{array}\right]
\end{array}\right]
$$

(21)

(Pause: 4)
A <Would you like the quickest or the shortest route.>

pop(a.private.agenda)
push(a.shared.qud, *Does B want the quickest or shortest route?*)
push(b.shared.qud, *Does B want the quickest or shortest route?*)
push(b.private.agenda, respond(*Does B want the quickest or shortest route?*))

$$
\left[\begin{array}{lll}
 & \left[\begin{array}{lll}
\text{private} & = & \left[\begin{array}{lll}
\text{bel} & = & \{\} \\
\text{agenda} & = & \langle\rangle
\end{array}\right] \\
\text{shared} & = & \left[\begin{array}{lll}
\text{bel} & = & \left\{\begin{array}{l}
\textit{B wants to make the journey at 6 p.m.} \\
\textit{B wants to go to Edwinstowe in Nottingham} \\
\textit{B wants to start the journey in Great Malvern} \\
\textit{B has A's attention} \\
\textit{A has B's attention}
\end{array}\right\} \\
\text{qud} & = & \langle\ \textit{Does B want the quickest or shortest route?}\ \rangle
\end{array}\right]
\end{array}\right] \\
\text{a} = & \\
 & \left[\begin{array}{lll}
\text{private} & = & \left[\begin{array}{lll}
\text{bel} & = & \{\} \\
\text{agenda} & = & \langle\ \text{respond}(\textit{Does B want the quickest or shortest route?})\ \rangle
\end{array}\right] \\
\text{shared} & = & \left[\begin{array}{lll}
\text{bel} & = & \left\{\begin{array}{l}
\textit{B wants to make the journey at 6 p.m.} \\
\textit{B wants to go to Edwinstowe in Nottingham} \\
\textit{B wants to start the journey in Great Malvern} \\
\textit{B wants a route from A} \\
\textit{B has A's attention} \\
\textit{A has B's attention} \\
\textit{B wants assistance}
\end{array}\right\} \\
\text{qud} & = & \langle\ \textit{Does B want the quickest or shortest route?}\ \rangle
\end{array}\right]
\end{array}\right] \\
\text{b} = &
\end{array}\right]
$$

(22)

B <Quickest.>

push(a.private.agenda, instruct(*Wait while route from Malvern to Edwinstowe is calculated*))
add(a.shared.bel, *B wants the quickest route*)
pop(a.shared.qud)
pop(b.private.agenda)
add(b.shared.bel, *B wants the quickest route*)
pop(b.shared.qud)

$$
\left[\begin{array}{lll}
 & \left[\begin{array}{lll}
\text{private} & = & \left[\begin{array}{lll}
\text{bel} & = & \{\} \\
\text{agenda} & = & \langle\ \text{instruct}(\textit{Wait while route from Malvern to Edwinstowe is calculated})\ \rangle
\end{array}\right] \\
\text{shared} & = & \left[\begin{array}{lll}
\text{bel} & = & \left\{\begin{array}{l}
\textit{B wants the quickest route} \\
\textit{B wants to make the journey at 6 p.m.} \\
\textit{B wants to go to Edwinstowe in Nottingham} \\
\textit{B wants to start the journey in Great Malvern} \\
\textit{B has A's attention} \\
\textit{A has B's attention}
\end{array}\right\} \\
\text{qud} & = & \langle\rangle
\end{array}\right]
\end{array}\right] \\
\text{a} = & \\
 & \left[\begin{array}{lll}
\text{private} & = & \left[\begin{array}{lll}
\text{bel} & = & \{\} \\
\text{agenda} & = & \langle\rangle
\end{array}\right] \\
\text{shared} & = & \left[\begin{array}{lll}
\text{bel} & = & \left\{\begin{array}{l}
\textit{B wants the quickest route} \\
\textit{B wants to make the journey at 6 p.m.} \\
\textit{B wants to go to Edwinstowe in Nottingham} \\
\textit{B wants to start the journey in Great Malvern} \\
\textit{B wants a route from A} \\
\textit{B has A's attention} \\
\textit{A has B's attention} \\
\textit{B wants assistance}
\end{array}\right\} \\
\text{qud} & = & \langle\rangle
\end{array}\right]
\end{array}\right] \\
\text{b} = &
\end{array}\right]
$$

(23)

(Pause: 2)
A <Please wait> <while your route from Malvern to Edwinstowe is calculated.>

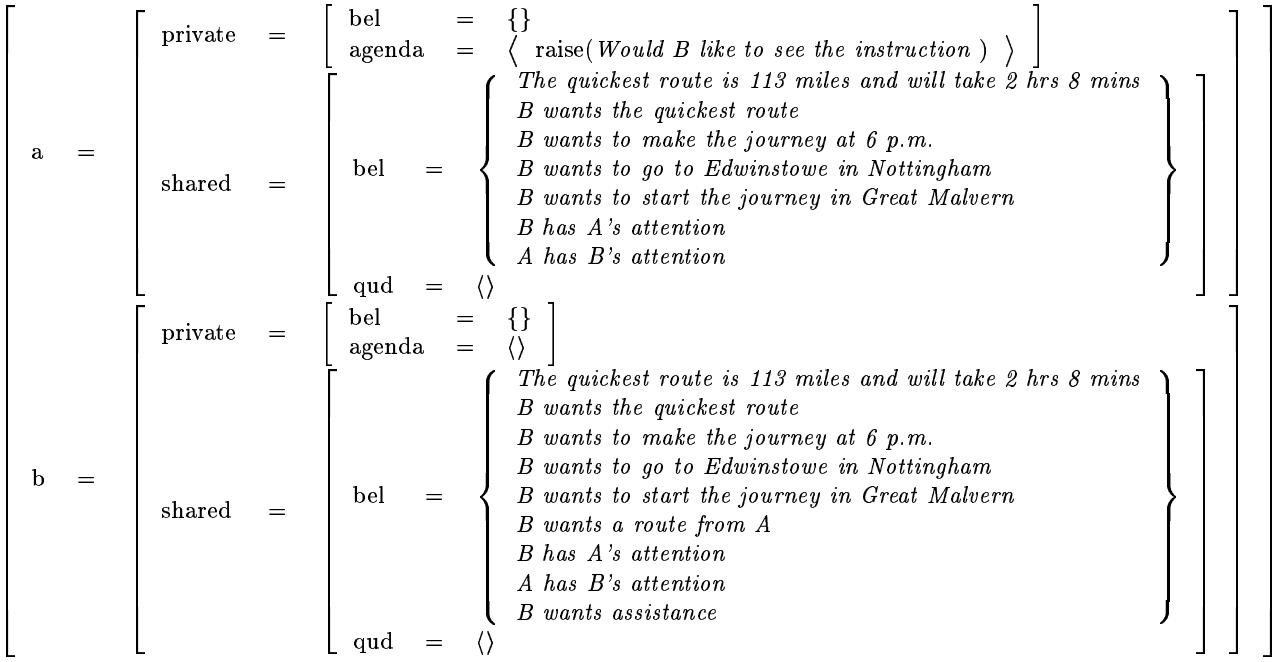
pop(a.private.agenda)
push(a.shared.qud, *How long is the quickest route and how long will it take?*)
push(a.private.agenda, respond(*How long is the quickest route and how long will it take?*))
push(b.shared.qud, *How long is the quickest route and how long will it take?*)

$$
\left[\begin{array}{lll}
\text{a} & = & \left[\begin{array}{lll}
\text{private} & = & \left[\begin{array}{lll}
\text{bel} & = & \{\} \\
\text{agenda} & = & \langle\ \text{respond}(\textit{How long is the quickest route and how long will it take?}\ )\ \rangle
\end{array}\right] \\
\text{shared} & = & \left[\begin{array}{lll}
\text{bel} & = & \left\{\begin{array}{l}
\textit{B wants the quickest route} \\
\textit{B wants to make the journey at 6 p.m.} \\
\textit{B wants to go to Edwinstowe in Nottingham} \\
\textit{B wants to start the journey in Great Malvern} \\
\textit{B has A's attention} \\
\textit{A has B's attention}
\end{array}\right\} \\
\text{qud} & = & \langle\ \textit{How long is the quickest route and how long will it take?}\ \rangle
\end{array}\right]
\end{array}\right] \\
\text{b} & = & \left[\begin{array}{lll}
\text{private} & = & \left[\begin{array}{lll}
\text{bel} & = & \{\} \\
\text{agenda} & = & \langle\rangle
\end{array}\right] \\
\text{shared} & = & \left[\begin{array}{lll}
\text{bel} & = & \left\{\begin{array}{l}
\textit{B wants the quickest route} \\
\textit{B wants to make the journey at 6 p.m.} \\
\textit{B wants to go to Edwinstowe in Nottingham} \\
\textit{B wants to start the journey in Great Malvern} \\
\textit{B wants a route from A} \\
\textit{B has A's attention} \\
\textit{A has B's attention} \\
\textit{B wants assistance}
\end{array}\right\} \\
\text{qud} & = & \langle\ \textit{How long is the quickest route and how long will it take?}\ \rangle
\end{array}\right]
\end{array}\right]
\end{array}\right]
$$

(24)

(Pause: 3)
<The quickest route is one hundred and thirteen miles and will take two hours eight minutes.>

pop(a.private.agenda)
push(a.private.agenda, raise(*Would B like to see the instruction*))
add(a.shared.bel, *The quickest route is 113 miles and will take 2 hrs 8 mins*)
pop(a.shared.qud)
add(b.shared.bel, *The quickest route is 113 miles and will take 2 hrs 8 mins*)
pop(b.shared.qud)



(25)

(Pause: 1)
<Would you like me to send the instructions to you.>

pop(a.private.agenda)
push(a.shared.qud, *Would B like to see the instruction*)
push(b.shared.qud, *Would B like to see the instruction*)
push(b.private.agenda, respond(*Would B like to see the instruction*))

$$
\left[\begin{array}{lll}
 & \left[\begin{array}{lll}
\text{private} & = & \left[\begin{array}{lll}
\text{bel} & = & \{\} \\
\text{agenda} & = & \langle\rangle
\end{array}\right] \\
\text{shared} & = & \left[\begin{array}{lll}
\text{bel} & = & \left\{\begin{array}{l}
\textit{The quickest route is 113 miles and will take 2 hrs 8 mins} \\
\textit{B wants the quickest route} \\
\textit{B wants to make the journey at 6 p.m.} \\
\textit{B wants to go to Edwinstowe in Nottingham} \\
\textit{B wants to start the journey in Great Malvern} \\
\textit{B has A's attention} \\
\textit{A has B's attention}
\end{array}\right\} \\
\text{qud} & = & \langle\ \textit{Would B like to see the instruction}\ \rangle
\end{array}\right]
\end{array}\right] \\
\text{a} = & \\
 & \left[\begin{array}{lll}
\text{private} & = & \left[\begin{array}{lll}
\text{bel} & = & \{\} \\
\text{agenda} & = & \langle\ \text{respond}(\textit{Would B like to see the instruction})\ \rangle
\end{array}\right] \\
\text{shared} & = & \left[\begin{array}{lll}
\text{bel} & = & \left\{\begin{array}{l}
\textit{The quickest route is 113 miles and will take 2 hrs 8 mins} \\
\textit{B wants the quickest route} \\
\textit{B wants to make the journey at 6 p.m.} \\
\textit{B wants to go to Edwinstowe in Nottingham} \\
\textit{B wants to start the journey in Great Malvern} \\
\textit{B wants a route from A} \\
\textit{B has A's attention} \\
\textit{A has B's attention} \\
\textit{B wants assistance}
\end{array}\right\} \\
\text{qud} & = & \langle\ \textit{Would B like to see the instruction}\ \rangle
\end{array}\right]
\end{array}\right] \\
\text{b} = &
\end{array}\right]
$$

(26)

B <No.>

push(a.private.agenda, raise(*Does B require any further information?*))
add(a.shared.bel, *B would not like A to send the instructions*)
pop(a.shared.qud)
pop(b.private.agenda)
add(b.shared.bel, *B would not like A to send the instructions*)
pop(b.shared.qud)

$$
\left[\begin{array}{lll}
 & \left[\begin{array}{lll}
\text{private} & = & \left[\begin{array}{lll}
\text{bel} & = & \{\} \\
\text{agenda} & = & \langle\ \text{raise}(\textit{Does B require any further information?}\ )\ \rangle
\end{array}\right] \\
\text{shared} & = & \left[\begin{array}{lll}
\text{bel} & = & \left\{\begin{array}{l}
\textit{B would not like A to send the instructions} \\
\textit{The quickest route is 113 miles and will take 2 hrs 8 mins} \\
\textit{B wants the quickest route} \\
\textit{B wants to make the journey at 6 p.m.} \\
\textit{B wants to go to Edwinstowe in Nottingham} \\
\textit{B wants to start the journey in Great Malvern} \\
\textit{B has A's attention} \\
\textit{A has B's attention}
\end{array}\right\} \\
\text{qud} & = & \langle\rangle
\end{array}\right]
\end{array}\right] \\
\text{a} = & & \\
 & \left[\begin{array}{lll}
\text{private} & = & \left[\begin{array}{lll}
\text{bel} & = & \{\} \\
\text{agenda} & = & \langle\rangle
\end{array}\right] \\
\text{shared} & = & \left[\begin{array}{lll}
\text{bel} & = & \left\{\begin{array}{l}
\textit{B would not like A to send the instructions} \\
\textit{The quickest route is 113 miles and will take 2 hrs 8 mins} \\
\textit{B wants the quickest route} \\
\textit{B wants to make the journey at 6 p.m.} \\
\textit{B wants to go to Edwinstowe in Nottingham} \\
\textit{B wants to start the journey in Great Malvern} \\
\textit{B wants a route from A} \\
\textit{B has A's attention} \\
\textit{A has B's attention} \\
\textit{B wants assistance}
\end{array}\right\} \\
\text{qud} & = & \langle\rangle
\end{array}\right]
\end{array}\right] \\
\text{b} = & &
\end{array}\right]
$$

(27)

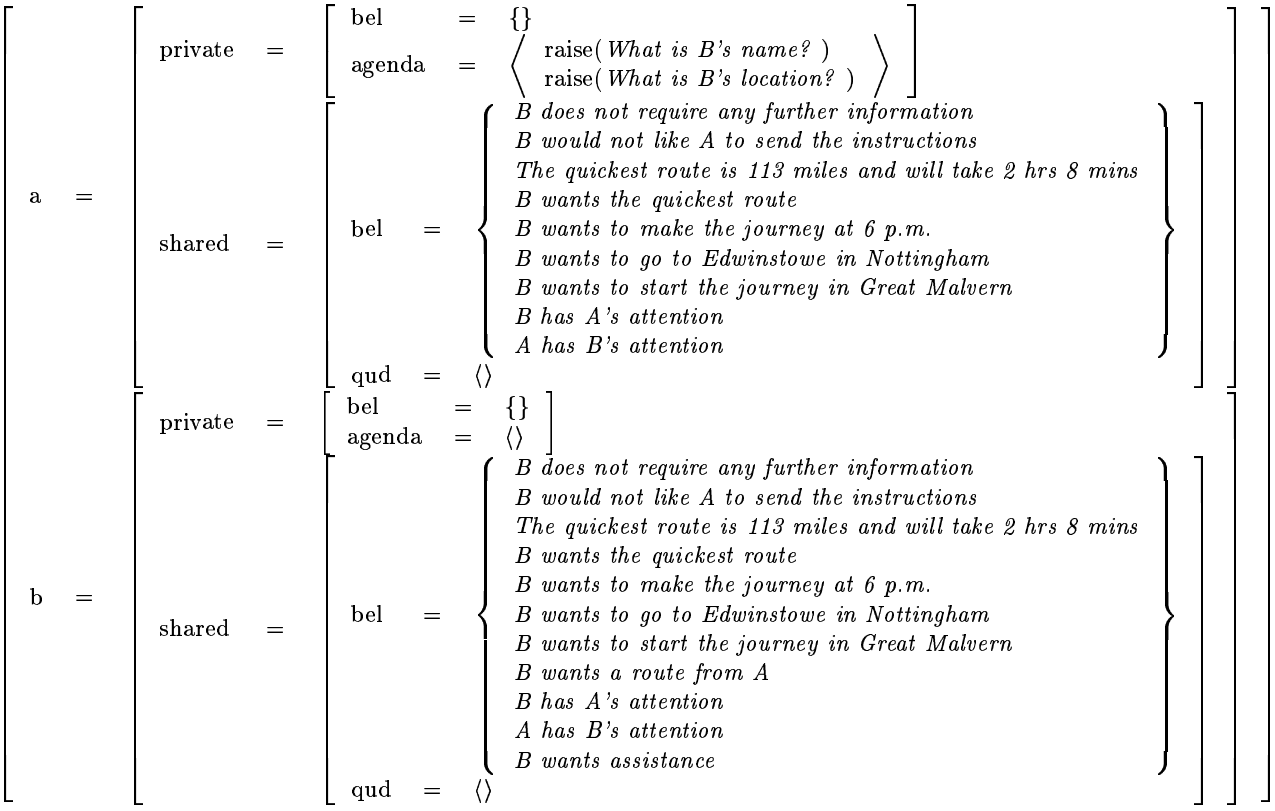
(Pause: 2)
A <Do you require any further information now.>
B

pop(a.private.agenda)
push(a.shared.qud, *Does B require any further information?*)
push(b.shared.qud, *Does B require any further information?*)
push(b.private.agenda, respond(*Does B require any further information?*))

$$
\left[\begin{array}{lll}
 & \left[\begin{array}{lll}
\text{private} & = & \left[\begin{array}{lll}
\text{bel} & = & \{\} \\
\text{agenda} & = & \langle\rangle
\end{array}\right] \\
\text{shared} & = & \left[\begin{array}{lll}
\text{bel} & = & \left\{\begin{array}{l}
\textit{B would not like A to send the instructions} \\
\textit{The quickest route is 113 miles and will take 2 hrs 8 mins} \\
\textit{B wants the quickest route} \\
\textit{B wants to make the journey at 6 p.m.} \\
\textit{B wants to go to Edwinstowe in Nottingham} \\
\textit{B wants to start the journey in Great Malvern} \\
\textit{B has A's attention} \\
\textit{A has B's attention}
\end{array}\right\} \\
\text{qud} & = & \left\langle\ \textit{Does B require any further information?}\ \right\rangle
\end{array}\right]
\end{array}\right] \\
\text{a} = & \\
 & \left[\begin{array}{lll}
\text{private} & = & \left[\begin{array}{lll}
\text{bel} & = & \{\} \\
\text{agenda} & = & \left\langle\ \text{respond}(\textit{Does B require any further information?})\ \right\rangle
\end{array}\right] \\
\text{shared} & = & \left[\begin{array}{lll}
\text{bel} & = & \left\{\begin{array}{l}
\textit{B would not like A to send the instructions} \\
\textit{The quickest route is 113 miles and will take 2 hrs 8 mins} \\
\textit{B wants the quickest route} \\
\textit{B wants to make the journey at 6 p.m.} \\
\textit{B wants to go to Edwinstowe in Nottingham} \\
\textit{B wants to start the journey in Great Malvern} \\
\textit{B wants a route from A} \\
\textit{B has A's attention} \\
\textit{A has B's attention} \\
\textit{B wants assistance}
\end{array}\right\} \\
\text{qud} & = & \left\langle\ \textit{Does B require any further information?}\ \right\rangle
\end{array}\right]
\end{array}\right] \\
\text{b} = &
\end{array}\right]
$$

(28)
B <No.>

add(a.shared.bel, *B does not require any further information*)
pop(a.shared.qud)
push(a.private.agenda, raise(*What is B's location?*))
push(a.private.agenda, raise(*What is B's name?*))
add(b.shared.bel, *B does not require any further information*)
pop(b.private.agenda)
pop(b.shared.qud)



(29)

(Pause: 1)
A <Can I have your name please.>

pop(a.private.agenda)
push(a.shared.qud, *What is B's name?*)
push(b.shared.qud, *What is B's name?*)
push(b.private.agenda, respond(*What is B's name?*))

$$
\left[\begin{array}{lll}
 & \left[\begin{array}{lll}
\text{private} & = & \left[\begin{array}{lll}
\text{bel} & = & \{\} \\
\text{agenda} & = & \langle\ \text{raise}(\textit{What is B's location?}\ )\ \rangle
\end{array}\right] \\
\text{shared} & = & \left[\begin{array}{lll}
\text{bel} & = & \left\{\begin{array}{l}
\textit{B does not require any further information} \\
\textit{B would not like A to send the instructions} \\
\textit{The quickest route is 113 miles and will take 2 hrs 8 mins} \\
\textit{B wants the quickest route} \\
\textit{B wants to make the journey at 6 p.m.} \\
\textit{B wants to go to Edwinstowe in Nottingham} \\
\textit{B wants to start the journey in Great Malvern} \\
\textit{B has A's attention} \\
\textit{A has B's attention}
\end{array}\right\} \\
\text{qud} & = & \langle\ \textit{What is B's name?}\ \rangle
\end{array}\right]
\end{array}\right] \\
\text{a} = & \\
 & \left[\begin{array}{lll}
\text{private} & = & \left[\begin{array}{lll}
\text{bel} & = & \{\} \\
\text{agenda} & = & \langle\ \text{respond}(\textit{What is B's name?}\ )\ \rangle
\end{array}\right] \\
\text{shared} & = & \left[\begin{array}{lll}
\text{bel} & = & \left\{\begin{array}{l}
\textit{B does not require any further information} \\
\textit{B would not like A to send the instructions} \\
\textit{The quickest route is 113 miles and will take 2 hrs 8 mins} \\
\textit{B wants the quickest route} \\
\textit{B wants to make the journey at 6 p.m.} \\
\textit{B wants to go to Edwinstowe in Nottingham} \\
\textit{B wants to start the journey in Great Malvern} \\
\textit{B wants a route from A} \\
\textit{B has A's attention} \\
\textit{A has B's attention} \\
\textit{B wants assistance}
\end{array}\right\} \\
\text{qud} & = & \langle\ \textit{What is B's name?}\ \rangle
\end{array}\right]
\end{array}\right]
\end{array}\right]
$$

(30)

(Pause: 1)
B <Mr Smith>

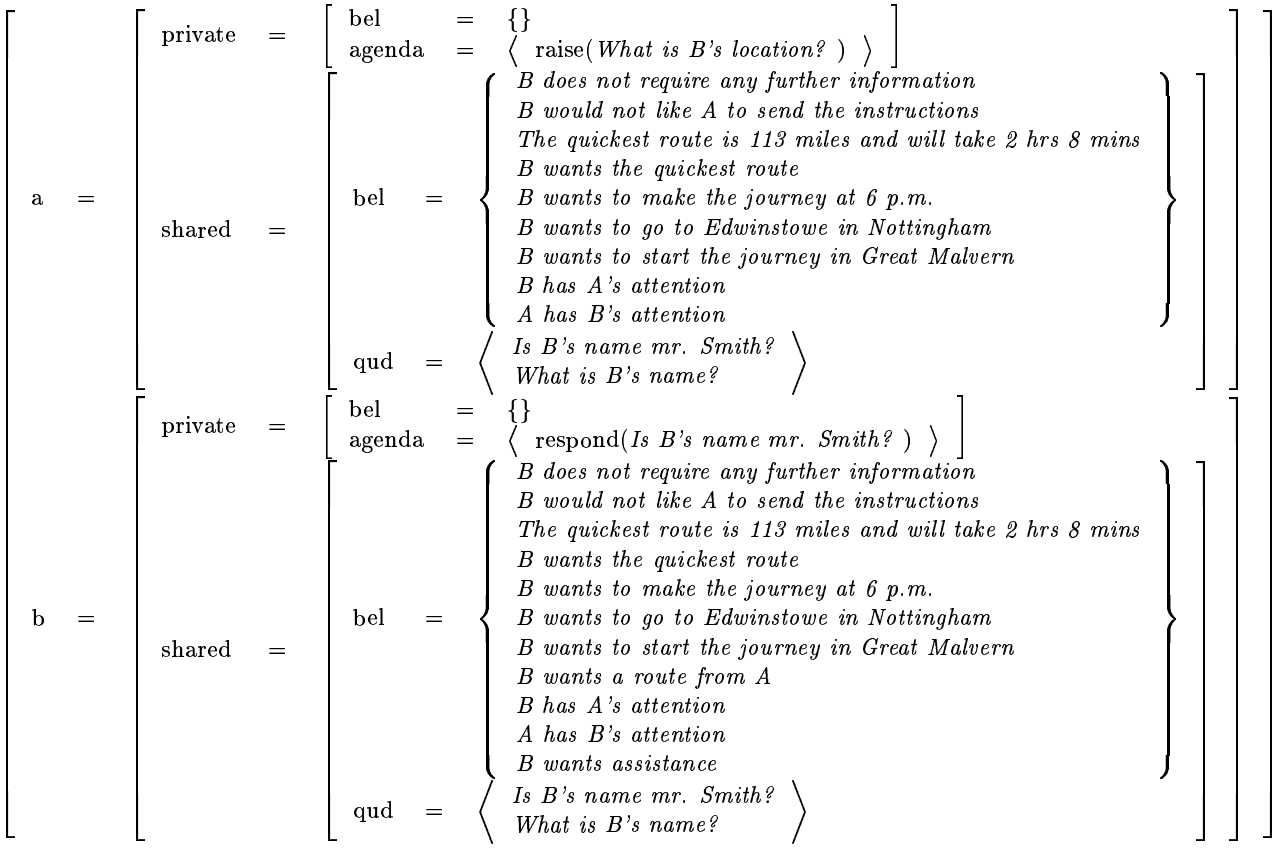
push(a.private.agenda, raise(*Is B's name mr. Smith?*))
pop(b.private.agenda)
add(b.shared.bel, *B's name is mr. Smith*)
pop(b.shared.qud)

$$
\left[\begin{array}{lll}
 & \left[\begin{array}{lll}
\text{private} & = & \left[\begin{array}{lll}
\text{bel} & = & \{\} \\
\text{agenda} & = & \left\langle \begin{array}{l} \text{raise}(\textit{Is B's name mr. Smith?}\ ) \\ \text{raise}(\textit{What is B's location?}\ ) \end{array} \right\rangle
\end{array}\right] \\
\text{shared} & = & \left[\begin{array}{lll}
\text{bel} & = & \left\{\begin{array}{l}
\textit{B does not require any further information} \\
\textit{B would not like A to send the instructions} \\
\textit{The quickest route is 113 miles and will take 2 hrs 8 mins} \\
\textit{B wants the quickest route} \\
\textit{B wants to make the journey at 6 p.m.} \\
\textit{B wants to go to Edwinstowe in Nottingham} \\
\textit{B wants to start the journey in Great Malvern} \\
\textit{B has A's attention} \\
\textit{A has B's attention}
\end{array}\right\} \\
\text{qud} & = & \left\langle\ \textit{What is B's name?}\ \right\rangle
\end{array}\right]
\end{array}\right] \\
\text{a} = & \\
 & \left[\begin{array}{lll}
\text{private} & = & \left[\begin{array}{ll}
\text{bel} & = \{\} \\
\text{agenda} & = \langle\rangle
\end{array}\right] \\
\text{shared} & = & \left[\begin{array}{lll}
\text{bel} & = & \left\{\begin{array}{l}
\textit{B's name is mr. Smith} \\
\textit{B does not require any further information} \\
\textit{B would not like A to send the instructions} \\
\textit{The quickest route is 113 miles and will take 2 hrs 8 mins} \\
\textit{B wants the quickest route} \\
\textit{B wants to make the journey at 6 p.m.} \\
\textit{B wants to go to Edwinstowe in Nottingham} \\
\textit{B wants to start the journey in Great Malvern} \\
\textit{B wants a route from A} \\
\textit{B has A's attention} \\
\textit{A has B's attention} \\
\textit{B wants assistance}
\end{array}\right\} \\
\text{qud} & = & \langle\rangle
\end{array}\right]
\end{array}\right] \\
\text{b} = &
\end{array}\right]
$$

(31)

(Pause: 1)
A <Mr Smith>

pop(a.private.agenda)
push(a.shared.qud, *Is B's name mr. Smith?*)
delete(b.shared.bel, *B's name is mr. Smith*)
push(b.shared.qud, *What is B's name?*)
push(b.shared.qud, *Is B's name mr. Smith?*)
push(b.private.agenda, respond(*Is B's name mr. Smith?*))



(32)

B <Yes=>

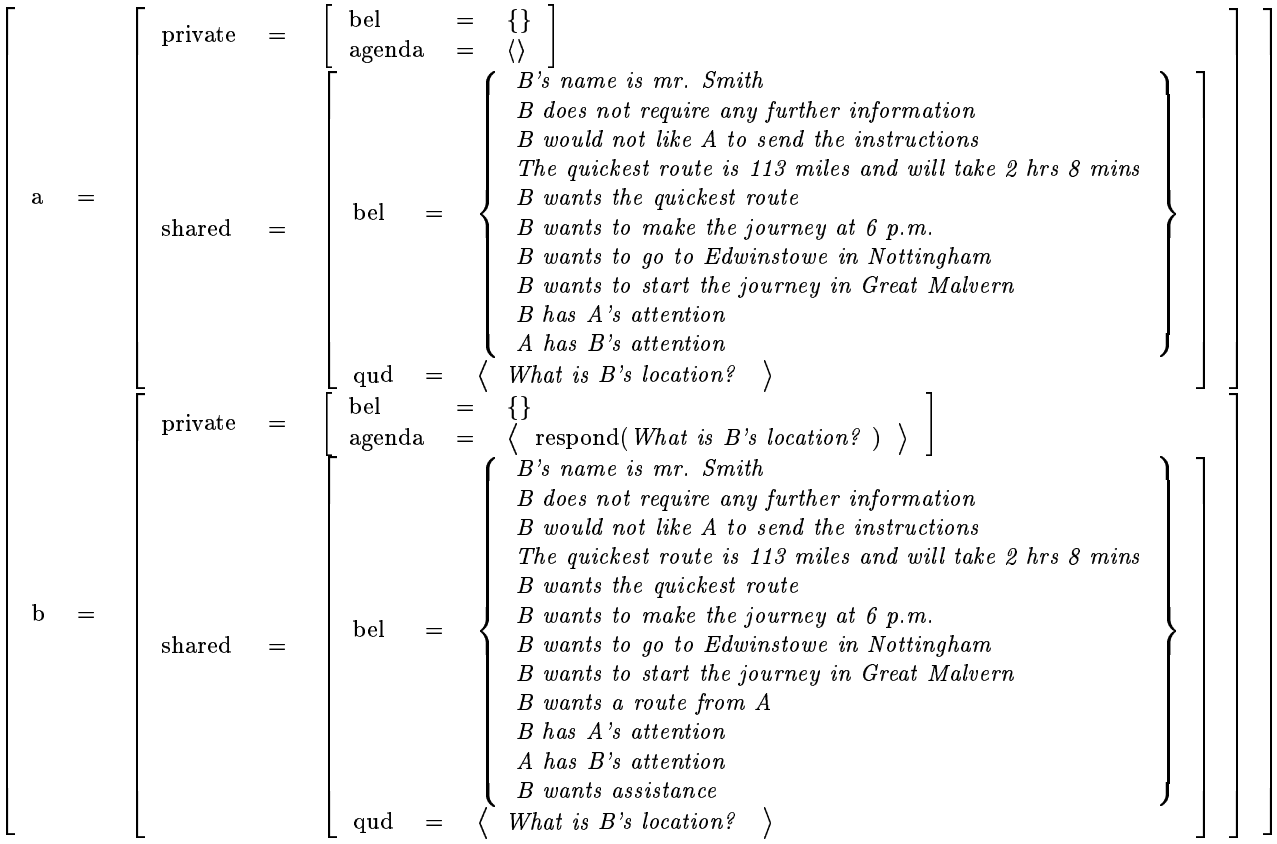
add(a.shared.bel, *B's name is mr. Smith*)
pop(a.shared.qud)
pop(a.shared.qud)
pop(b.private.agenda)
add(b.shared.bel, *B's name is mr. Smith*)
pop(b.shared.qud)
pop(b.shared.qud)

$$
\left[\begin{array}{lll}
 & \left[\begin{array}{lll}
\text{private} & = & \left[\begin{array}{lll}
\text{bel} & = & \{\} \\
\text{agenda} & = & \langle\ \text{raise}(\textit{What is B's location?}\ )\ \rangle
\end{array}\right] \\
\text{shared} & = & \left[\begin{array}{lll}
\text{bel} & = & \left\{\begin{array}{l}
\textit{B's name is mr. Smith} \\
\textit{B does not require any further information} \\
\textit{B would not like A to send the instructions} \\
\textit{The quickest route is 113 miles and will take 2 hrs 8 mins} \\
\textit{B wants the quickest route} \\
\textit{B wants to make the journey at 6 p.m.} \\
\textit{B wants to go to Edwinstowe in Nottingham} \\
\textit{B wants to start the journey in Great Malvern} \\
\textit{B has A's attention} \\
\textit{A has B's attention}
\end{array}\right\} \\
\text{qud} & = & \langle\rangle
\end{array}\right]
\end{array}\right] \\
\text{a} = & \\
 & \left[\begin{array}{lll}
\text{private} & = & \left[\begin{array}{lll}
\text{bel} & = & \{\} \\
\text{agenda} & = & \langle\rangle
\end{array}\right] \\
\text{shared} & = & \left[\begin{array}{lll}
\text{bel} & = & \left\{\begin{array}{l}
\textit{B's name is mr. Smith} \\
\textit{B does not require any further information} \\
\textit{B would not like A to send the instructions} \\
\textit{The quickest route is 113 miles and will take 2 hrs 8 mins} \\
\textit{B wants the quickest route} \\
\textit{B wants to make the journey at 6 p.m.} \\
\textit{B wants to go to Edwinstowe in Nottingham} \\
\textit{B wants to start the journey in Great Malvern} \\
\textit{B wants a route from A} \\
\textit{B has A's attention} \\
\textit{A has B's attention} \\
\textit{B wants assistance}
\end{array}\right\} \\
\text{qud} & = & \langle\rangle
\end{array}\right]
\end{array}\right] \\
\text{b} = &
\end{array}\right]
$$

(33)

A <=And your location please.>

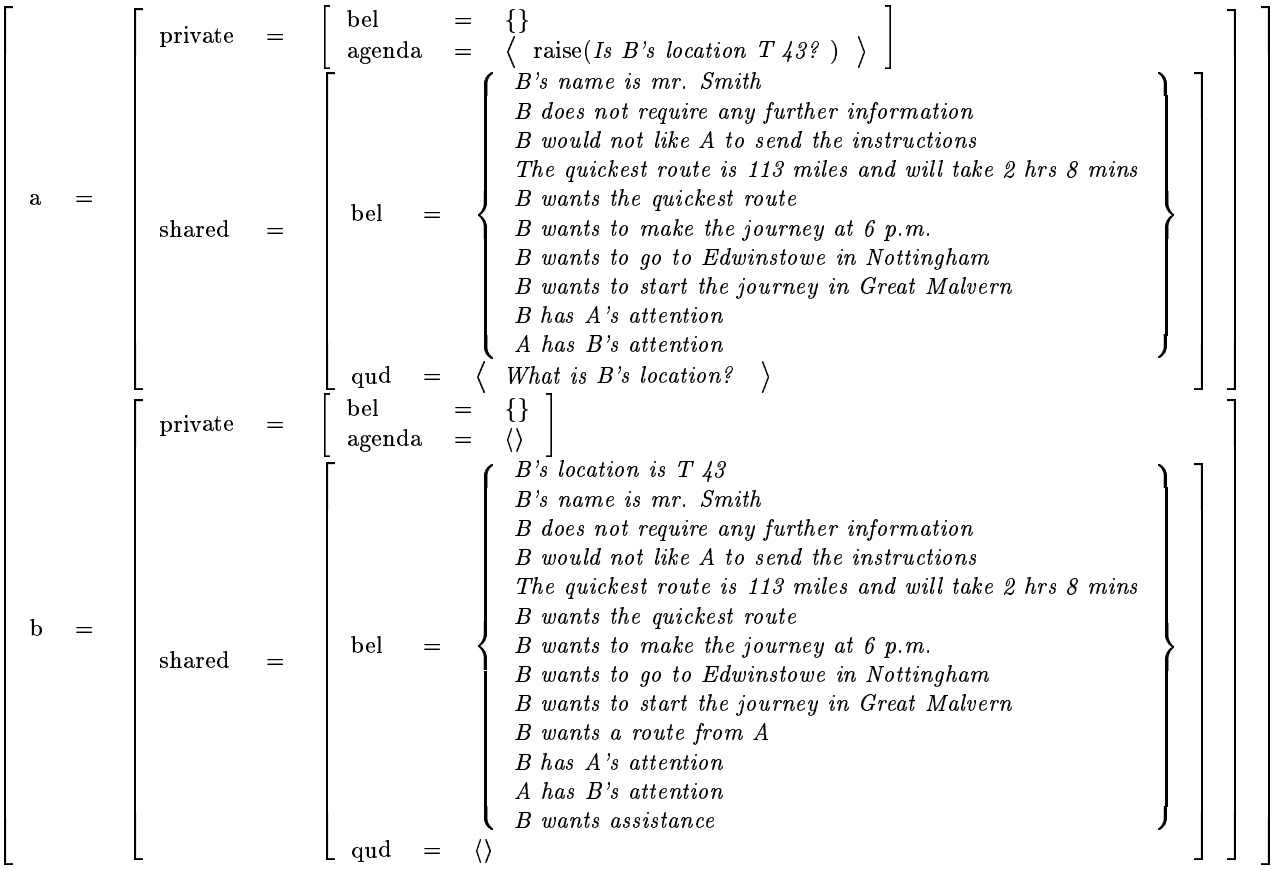
pop(a.private.agenda)
push(a.shared.qud, *What is B's location?*)
push(b.shared.qud, *What is B's location?*)
push(b.private.agenda, respond(*What is B's location?*))



(34)

B <T (.) 43>

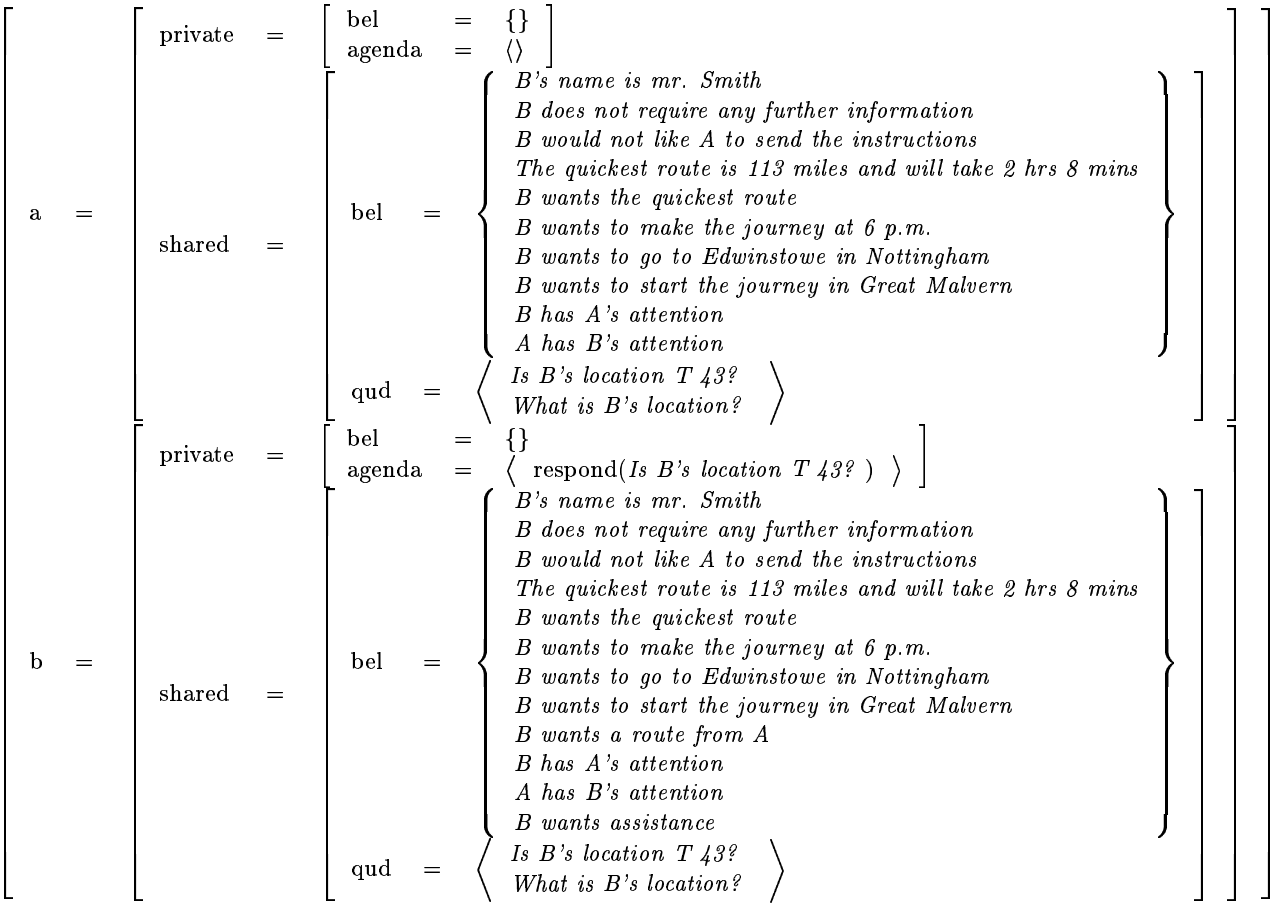
push(a.private.agenda, raise(*Is B's location T 43?*))
pop(b.private.agenda)
add(b.shared.bel, *B's location is T 43*)
pop(b.shared.qud)



(35)

A <T 43>

pop(a.private.agenda)
push(a.shared.qud, *Is B's location T 43?*)
delete(b.shared.bel, *B's location is T 43*)
push(b.shared.qud, *What is B's location?*)
push(b.shared.qud, *Is B's location T 43?*)
push(b.private.agenda, respond(*Is B's location T 43?*))



(36)

B <Yes.>

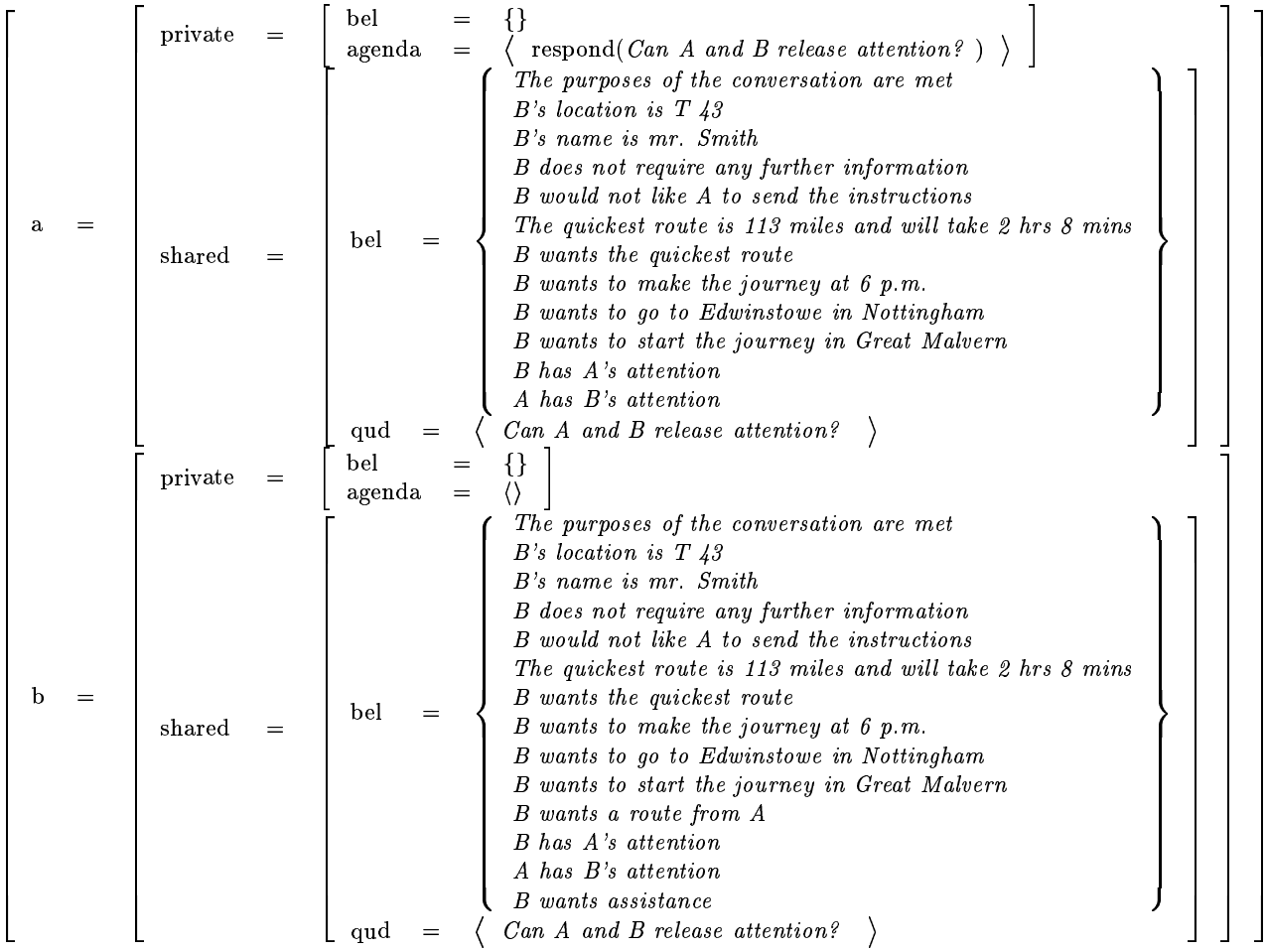
add(a.shared.bel, *B's location is T 43*)
pop(a.shared.qud)
pop(a.shared.qud)
push(a.shared.qud, *Can A and B release attention?*)
push(a.shared.qud, *Are the purposes of the conversation met?*)
push(a.private.agenda, respond(*Are the purposes of the conversation met?*))
pop(b.private.agenda)
add(b.shared.bel, *B's location is T 43*)
pop(b.shared.qud)
pop(b.shared.qud)

$$
\left[\begin{array}{lll}
 & & \left[\begin{array}{lll}
\text{private} & = & \left[\begin{array}{lll}
\text{bel} & = & \{\} \\
\text{agenda} & = & \left\langle \text{ respond}(\textit{Are the purposes of the conversation met?}\ ) \right\rangle
\end{array}\right] \\
\text{shared} & = & \left[\begin{array}{lll}
\text{bel} & = & \left\{\begin{array}{l}
\textit{B's location is T 43} \\
\textit{B's name is mr. Smith} \\
\textit{B does not require any further information} \\
\textit{B would not like A to send the instructions} \\
\textit{The quickest route is 113 miles and will take 2 hrs 8 mins} \\
\textit{B wants the quickest route} \\
\textit{B wants to make the journey at 6 p.m.} \\
\textit{B wants to go to Edwinstowe in Nottingham} \\
\textit{B wants to start the journey in Great Malvern} \\
\textit{B has A's attention} \\
\textit{A has B's attention}
\end{array}\right\} \\
\text{qud} & = & \left\langle \begin{array}{l}
\textit{Are the purposes of the conversation met?} \\
\textit{Can A and B release attention?}
\end{array} \right\rangle
\end{array}\right]
\end{array}\right] \\
\text{a} = & & \\
 & & \left[\begin{array}{lll}
\text{private} & = & \left[\begin{array}{lll}
\text{bel} & = & \{\} \\
\text{agenda} & = & \langle\rangle
\end{array}\right] \\
\text{shared} & = & \left[\begin{array}{lll}
\text{bel} & = & \left\{\begin{array}{l}
\textit{B's location is T 43} \\
\textit{B's name is mr. Smith} \\
\textit{B does not require any further information} \\
\textit{B would not like A to send the instructions} \\
\textit{The quickest route is 113 miles and will take 2 hrs 8 mins} \\
\textit{B wants the quickest route} \\
\textit{B wants to make the journey at 6 p.m.} \\
\textit{B wants to go to Edwinstowe in Nottingham} \\
\textit{B wants to start the journey in Great Malvern} \\
\textit{B wants a route from A} \\
\textit{B has A's attention} \\
\textit{A has B's attention} \\
\textit{B wants assistance}
\end{array}\right\} \\
\text{qud} & = & \langle\rangle
\end{array}\right]
\end{array}\right] \\
\text{b} = & &
\end{array}\right]
$$

(37)

A <OK.>

pop(a.private.agenda)
pop(a.shared.qud)
push(a.private.agenda, respond(*Can A and B release attention?*))
add(a.shared.bel, *The purposes of the conversation are met*)
push(b.shared.qud, *Are the purposes of the conversation met?*)
add(b.shared.bel, *The purposes of the conversation are met*)
pop(b.shared.qud)
push(b.shared.qud, *Can A and B release attention?*)



(38)
(.) <Thank you for calling.>
(Pause: 2)
<Goodbye.>

pop(a.private.agenda)
pop(a.shared.qud)
pop(b.shared.qud)
delete(a.shared.bel, *B has A's attention*)
delete(a.shared.bel, *A has B's attention*)
delete(b.shared.bel, *B has A's attention*)
delete(b.shared.bel, *A has B's attention*)

$$
\left[\begin{array}{lll}
 & & \left[\begin{array}{lll}
\text{private} & = & \left[\begin{array}{lll} \text{bel} & = & \{\} \\ \text{agenda} & = & \langle\rangle \end{array}\right] \\
\text{shared} & = & \left[\begin{array}{lll} \text{bel} & = & \left\{\begin{array}{l} \textit{The purposes of the conversation are met} \\ \textit{B's location is T 43} \\ \textit{B's name is mr. Smith} \\ \textit{B does not require any further information} \\ \textit{B would not like A to send the instructions} \\ \textit{The quickest route is 113 miles and will take 2 hrs 8 mins} \\ \textit{B wants the quickest route} \\ \textit{B wants to make the journey at 6 p.m.} \\ \textit{B wants to go to Edwinstowe in Nottingham} \\ \textit{B wants to start the journey in Great Malvern} \end{array}\right\} \\ \text{qud} & = & \langle\rangle \end{array}\right]
\end{array}\right] \\
\text{a} = & & \\
 & & \left[\begin{array}{lll}
\text{private} & = & \left[\begin{array}{lll} \text{bel} & = & \{\} \\ \text{agenda} & = & \langle\rangle \end{array}\right] \\
\text{shared} & = & \left[\begin{array}{lll} \text{bel} & = & \left\{\begin{array}{l} \textit{The purposes of the conversation are met} \\ \textit{B's location is T 43} \\ \textit{B's name is mr. Smith} \\ \textit{B does not require any further information} \\ \textit{B would not like A to send the instructions} \\ \textit{The quickest route is 113 miles and will take 2 hrs 8 mins} \\ \textit{B wants the quickest route} \\ \textit{B wants to make the journey at 6 p.m.} \\ \textit{B wants to go to Edwinstowe in Nottingham} \\ \textit{B wants to start the journey in Great Malvern} \\ \textit{B wants a route from A} \\ \textit{B wants assistance} \end{array}\right\} \\ \text{qud} & = & \langle\rangle \end{array}\right]
\end{array}\right] \\
\text{b} = & &
\end{array}\right]
$$

Appendix C

Appendix: Annotation of the Autoroute Dialogue in Scheme 2

(1)
Dialogue 127
%---start----%

pushRec(A.INT, GREET)
pushRec(A.INT, OFFERHELP)
pushRec(B.INT, *'Get a route from malvern to edwinstowe'*)

$$\left[\begin{array}{lll} \text{A} & = & \left[\begin{array}{lll} \text{G} & = & [] \\ \text{INT} & = & \left[\begin{array}{l} \text{OFFERHELP} \\ \text{GREET} \end{array}\right] \end{array}\right] \\ \text{B} & = & \left[\begin{array}{lll} \text{G} & = & [] \\ \text{INT} & = & [\ \textit{'Get a route from malvern to edwinstowe'}\ ] \end{array}\right] \end{array}\right]$$

(2)

A <Welcome to the Route Planning Service.> <How can I help you.>
(Pause: 1)

removeRec(A.INT, GREET)
removeRec(A.INT, OFFERHELP)

$$\left[\begin{array}{lcl} \textsc{a} & = & \left[\begin{array}{lcl} \textsc{g} & = & [] \end{array}\right] \\ \textsc{b} & = & \left[\begin{array}{lcl} \textsc{g} & = & [] \\ \textsc{int} & = & \left[\begin{array}{l} \textit{'Get a route from malvern to edwinstowe'} \end{array}\right] \end{array}\right] \end{array}\right]$$

add_fieldRec(A.DU2, [])
add_fieldRec(B.DU2, [])
pushRec(A.UDUS, DU2)
pushRec(B.UDUS, DU2)

$$\left[\begin{array}{lcl} \textsc{a} & = & \left[\begin{array}{lcl} \textsc{g} & = & [] \\ \textsc{udus} & = & \left[\begin{array}{l} \textsc{du2} \end{array}\right] \\ \textsc{du2} & = & [] \end{array}\right] \\ \textsc{b} & = & \left[\begin{array}{lcl} \textsc{g} & = & [] \\ \textsc{int} & = & \left[\begin{array}{l} \textit{'Get a route from malvern to edwinstowe'} \end{array}\right] \\ \textsc{udus} & = & \left[\begin{array}{l} \textsc{du2} \end{array}\right] \\ \textsc{du2} & = & [] \end{array}\right] \end{array}\right]$$

pushRec(A.DU2.DH, **greet**(A,B))
pushRec(B.DU2.DH, **greet**(A,B))
pushRec(A.DU2.DH, I2:**info_request**(A,B,*'How can A help B?'*))
pushRec(B.DU2.DH, I2:**info_request**(A,B,*'How can A help B?'*))

$$\left[\begin{array}{lcl} \textsc{a} & = & \left[\begin{array}{lcl} \textsc{g} & = & [] \\ \textsc{udus} & = & \left[\begin{array}{l} \textsc{du2} \end{array}\right] \\ \textsc{du2} & = & \left[\begin{array}{lcl} \textsc{dh} & = & \left\langle \begin{array}{l} \textsc{i2}{:}\textbf{info\_request}(\textsc{a},\textsc{b},\textit{'How can A help B?'}) \\ \textbf{greet}(\textsc{a},\textsc{b}) \end{array} \right\rangle \end{array}\right] \end{array}\right] \\ \textsc{b} & = & \left[\begin{array}{lcl} \textsc{g} & = & [] \\ \textsc{int} & = & \left[\begin{array}{l} \textit{'Get a route from malvern to edwinstowe'} \end{array}\right] \\ \textsc{udus} & = & \left[\begin{array}{l} \textsc{du2} \end{array}\right] \\ \textsc{du2} & = & \left[\begin{array}{lcl} \textsc{dh} & = & \left\langle \begin{array}{l} \textsc{i2}{:}\textbf{info\_request}(\textsc{a},\textsc{b},\textit{'How can A help B?'}) \\ \textbf{greet}(\textsc{a},\textsc{b}) \end{array} \right\rangle \end{array}\right] \end{array}\right] \end{array}\right]$$

pushRec(A.G.OBL, **understandingAct**(B,DU2))
pushRec(B.G.OBL, **understandingAct**(B,DU2))
pushRec(A.DU2.OBL, **answer**(B,I2:**info_request**(A,B,*'How can A help B?'*)))
pushRec(B.DU2.OBL, **answer**(B,I2:**info_request**(A,B,*'How can A help B?'*)))

$$\left[\begin{array}{lll}
 & \left[\begin{array}{lll}
\text{G} & = & \left[\begin{array}{lll}\text{OBL} & = & \left[\ \textbf{understandingAct}(\text{B,DU2})\ \right]\end{array}\right] \\
\text{UDUS} & = & \left[\ \text{DU2}\ \right] \\
\text{DU2} & = & \left[\begin{array}{lll}\text{OBL} & = & \left[\ \textbf{answer}(\text{B,I2:}\textbf{info_request}(\text{A,B,}\textit{'How can A help B?'}))\ \right] \\ \text{DH} & = & \left\langle\begin{array}{l}\text{I2:}\textbf{info_request}(\text{A,B,}\textit{'How can A help B?'}) \\ \textbf{greet}(\text{A,B})\end{array}\right\rangle\end{array}\right]
\end{array}\right] \\
\text{A} = & \\
 & \left[\begin{array}{lll}
\text{G} & = & \left[\begin{array}{lll}\text{OBL} & = & \left[\ \textbf{understandingAct}(\text{B,DU2})\ \right]\end{array}\right] \\
\text{INT} & = & \left[\ \textit{'Get a route from malvern to edwinstowe'}\ \right] \\
\text{UDUS} & = & \left[\ \text{DU2}\ \right] \\
\text{DU2} & = & \left[\begin{array}{lll}\text{OBL} & = & \left[\ \textbf{answer}(\text{B,I2:}\textbf{info_request}(\text{A,B,}\textit{'How can A help B?'}))\ \right] \\ \text{DH} & = & \left\langle\begin{array}{l}\text{I2:}\textbf{info_request}(\text{A,B,}\textit{'How can A help B?'}) \\ \textbf{greet}(\text{A,B})\end{array}\right\rangle\end{array}\right]
\end{array}\right] \\
\text{B} = &
\end{array}\right]$$

pushRec(B.INT, **understandingAct**(B,DU2))
pushRec(B.INT, **answer**(B,I2:**info_request**(A,B,*'How can A help B?'*)))

$$\left[\begin{array}{lll}
 & \left[\begin{array}{lll}
\text{G} & = & \left[\begin{array}{lll}\text{OBL} & = & \left[\ \textbf{understandingAct}(\text{B,DU2})\ \right]\end{array}\right] \\
\text{UDUS} & = & \left[\ \text{DU2}\ \right] \\
\text{DU2} & = & \left[\begin{array}{lll}\text{OBL} & = & \left[\ \textbf{answer}(\text{B,I2:}\textbf{info_request}(\text{A,B,}\textit{'How can A help B?'}))\ \right] \\ \text{DH} & = & \left\langle\begin{array}{l}\text{I2:}\textbf{info_request}(\text{A,B,}\textit{'How can A help B?'}) \\ \textbf{greet}(\text{A,B})\end{array}\right\rangle\end{array}\right]
\end{array}\right] \\
\text{A} = & \\
 & \left[\begin{array}{lll}
\text{G} & = & \left[\begin{array}{lll}\text{OBL} & = & \left[\ \textbf{understandingAct}(\text{B,DU2})\ \right]\end{array}\right] \\
\text{INT} & = & \left[\begin{array}{l}\textbf{answer}(\text{B,I2:}\textbf{info_request}(\text{A,B,}\textit{'How can A help B?'})) \\ \textbf{understandingAct}(\text{B,DU2}) \\ \textit{'Get a route from malvern to edwinstowe'}\end{array}\right] \\
\text{UDUS} & = & \left[\ \text{DU2}\ \right] \\
\text{DU2} & = & \left[\begin{array}{lll}\text{OBL} & = & \left[\ \textbf{answer}(\text{B,I2:}\textbf{info_request}(\text{A,B,}\textit{'How can A help B?'}))\ \right] \\ \text{DH} & = & \left\langle\begin{array}{l}\text{I2:}\textbf{info_request}(\text{A,B,}\textit{'How can A help B?'}) \\ \textbf{greet}(\text{A,B})\end{array}\right\rangle\end{array}\right]
\end{array}\right] \\
\text{B} = &
\end{array}\right]$$

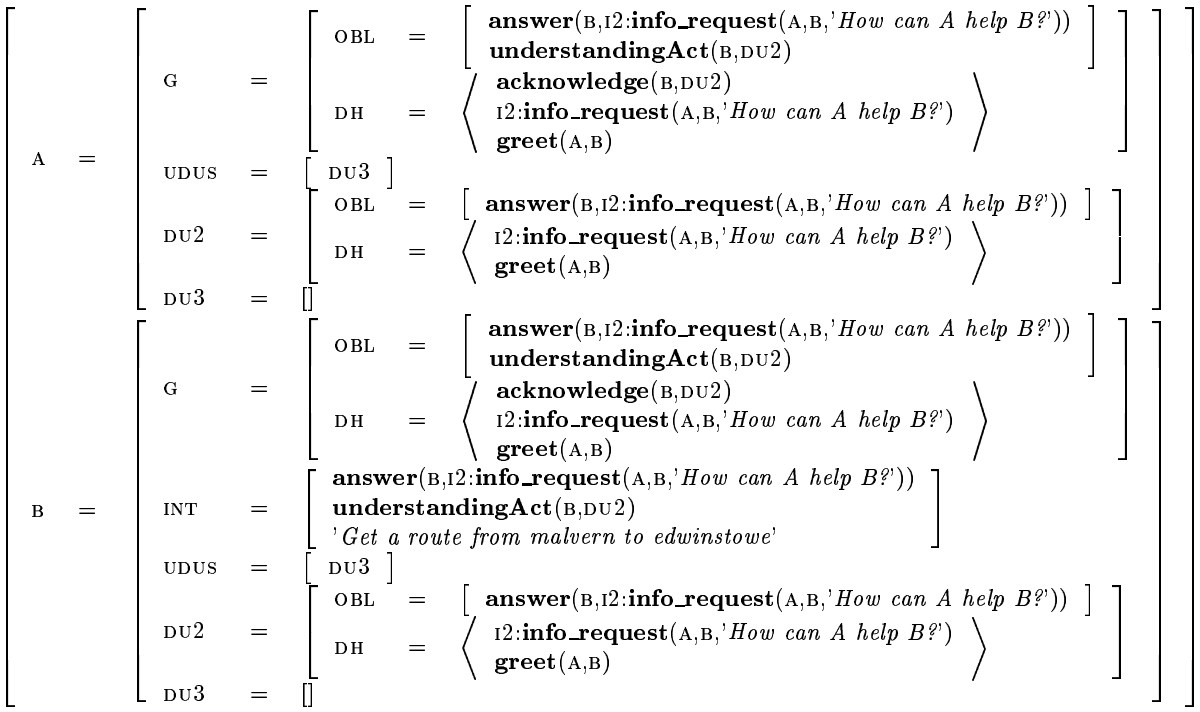
(3)

B <A route please.>
(Pause: 1)

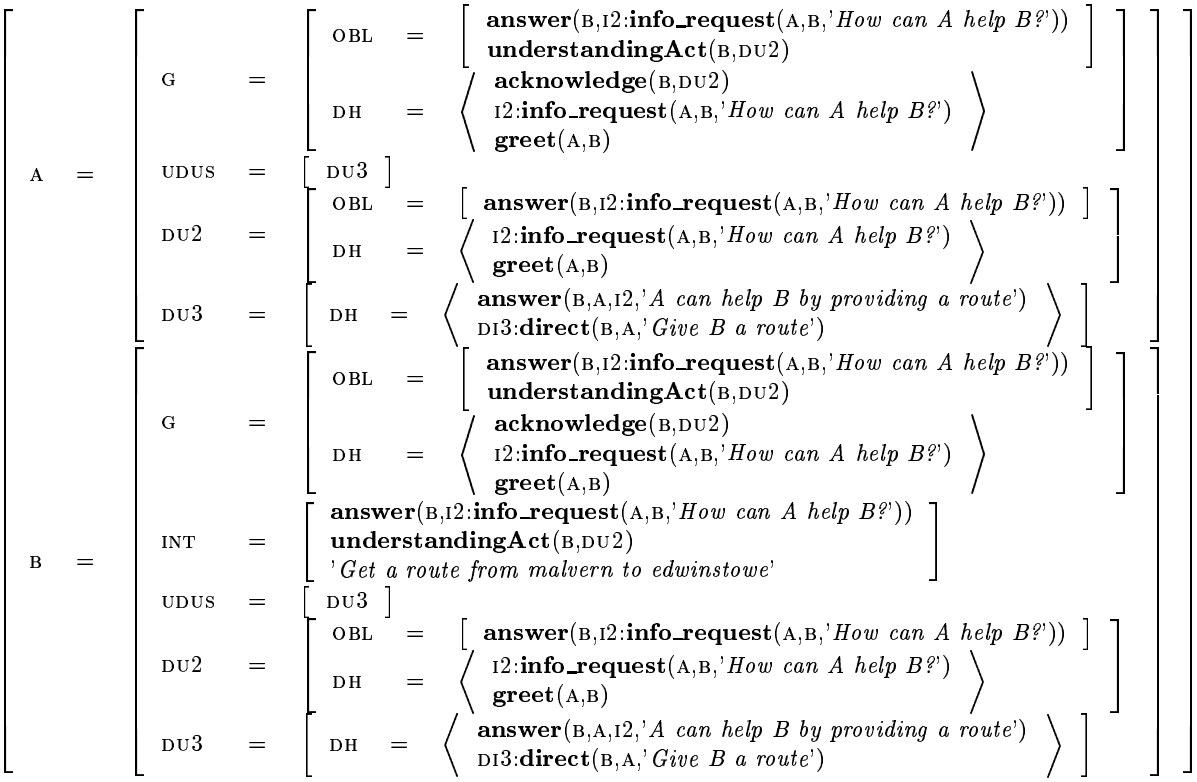
add_fieldRec(A.DU3, [])
add_fieldRec(B.DU3, [])
pushRec(A.UDUS, DU3)
pushRec(B.UDUS, DU3)

$$
\left[\begin{array}{lll}
 & \text{G} & = \left[\begin{array}{ll} \text{OBL} & = \left[\begin{array}{l} \textbf{understandingAct}(\text{B,DU2}) \end{array}\right] \end{array}\right] \\
 & \text{UDUS} & = \left[\begin{array}{l} \text{DU3} \\ \text{DU2} \end{array}\right] \\
\text{A} = & \text{DU2} & = \left[\begin{array}{ll} \text{OBL} & = \left[\begin{array}{l} \textbf{answer}(\text{B,I2:}\textbf{info\_request}(\text{A,B,}\textit{'How can A help B?'})) \end{array}\right] \\ \text{DH} & = \left\langle\begin{array}{l} \text{I2:}\textbf{info\_request}(\text{A,B,}\textit{'How can A help B?'}) \\ \textbf{greet}(\text{A,B}) \end{array}\right\rangle \end{array}\right] \\
 & \text{DU3} & = [] \\
 & \text{G} & = \left[\begin{array}{ll} \text{OBL} & = \left[\begin{array}{l} \textbf{understandingAct}(\text{B,DU2}) \end{array}\right] \end{array}\right] \\
 & \text{INT} & = \left[\begin{array}{l} \textbf{answer}(\text{B,I2:}\textbf{info\_request}(\text{A,B,}\textit{'How can A help B?'})) \\ \textbf{understandingAct}(\text{B,DU2}) \\ \textit{'Get a route from malvern to edwinstowe'} \end{array}\right] \\
\text{B} = & \text{UDUS} & = \left[\begin{array}{l} \text{DU3} \\ \text{DU2} \end{array}\right] \\
 & \text{DU2} & = \left[\begin{array}{ll} \text{OBL} & = \left[\begin{array}{l} \textbf{answer}(\text{B,I2:}\textbf{info\_request}(\text{A,B,}\textit{'How can A help B?'})) \end{array}\right] \\ \text{DH} & = \left\langle\begin{array}{l} \text{I2:}\textbf{info\_request}(\text{A,B,}\textit{'How can A help B?'}) \\ \textbf{greet}(\text{A,B}) \end{array}\right\rangle \end{array}\right] \\
 & \text{DU3} & = []
\end{array}\right]
$$

peRec(A.G, A.DU2)
peRec(B.G, B.DU2)
removeRec(A.UDUS, DU2)
removeRec(B.UDUS, DU2)
pushRec(A.G.DH, **acknowledge**(B,DU2))
pushRec(B.G.DH, **acknowledge**(B,DU2))



pushRec(A.DU3.DH, DI3:**direct**(B,A,'*Give B a route*'))
pushRec(B.DU3.DH, DI3:**direct**(B,A,'*Give B a route*'))
pushRec(A.DU3.DH, **answer**(B,A,I2,'*A can help B by providing a route*'))
pushRec(B.DU3.DH, **answer**(B,A,I2,'*A can help B by providing a route*'))


A = G = OBL = answer(B,I2:info_request(A,B,'How can A help B?')) understandingAct(B,DU2)
DH = acknowledge(B,DU2) I2:info_request(A,B,'How can A help B?') greet(A,B)
UDUS = DU3
DU2 = OBL = answer(B,I2:info_request(A,B,'How can A help B?'))
DH = I2:info_request(A,B,'How can A help B?') greet(A,B)
DU3 = DH = answer(B,A,I2,'A can help B by providing a route') DI3:direct(B,A,'Give B a route')
B = G = OBL = answer(B,I2:info_request(A,B,'How can A help B?')) understandingAct(B,DU2)
DH = acknowledge(B,DU2) I2:info_request(A,B,'How can A help B?') greet(A,B)
INT = answer(B,I2:info_request(A,B,'How can A help B?')) understandingAct(B,DU2) 'Get a route from malvern to edwinstowe'
UDUS = DU3
DU2 = OBL = answer(B,I2:info_request(A,B,'How can A help B?'))
DH = I2:info_request(A,B,'How can A help B?') greet(A,B)
DU3 = DH = answer(B,A,I2,'A can help B by providing a route') DI3:direct(B,A,'Give B a route')

pushRec(A.G.OBL, **understandingAct**(A,DU3))
pushRec(B.G.OBL, **understandingAct**(A,DU3))
pushRec(A.DU3.OBL, **address**(A,DI3:**direct**(B,A,'*Give B a route*')))
pushRec(B.DU3.OBL, **address**(A,DI3:**direct**(B,A,'*Give B a route*')))
addRec(A.DU3.OPT, **address**(A,DI3:**direct**(B,A,'*Give B a route*')))
addRec(B.DU3.OPT, **address**(A,DI3:**direct**(B,A,'*Give B a route*')))
pushRec(A.DU3.OBL, **accept**(A,DI3)→**obliged**(A,B,'*Give B a route*'))
pushRec(B.DU3.OBL, **accept**(A,DI3)→**obliged**(A,B,'*Give B a route*'))
addRec(A.DU3.SCP, **scp**(B,'*A can help B by providing a route*'))
addRec(B.DU3.SCP, **scp**(B,'*A can help B by providing a route*'))

```
A = [
  G = [
    OBL = [ understandingAct(A,DU3)
            answer(B,I2:info_request(A,B,'How can A help B?'))
            understandingAct(B,DU2) ]
    DH  = < acknowledge(B,DU2)
            I2:info_request(A,B,'How can A help B?')
            greet(A,B) >
  ]
  UDUS = [ DU3 ]
  DU2 = [
    OBL = [ answer(B,I2:info_request(A,B,'How can A help B?')) ]
    DH  = < I2:info_request(A,B,'How can A help B?')
            greet(A,B) >
  ]
  DU3 = [
    OBL = [ accept(A,DI3)→obliged(A,B,'Give B a route')
            address(A,DI3:direct(B,A,'Give B a route')) ]
    DH  = < answer(B,A,I2,'A can help B by providing a route')
            DI3:direct(B,A,'Give B a route') >
    SCP = { scp(B,'A can help B by providing a route') }
    OPT = { address(A,DI3:direct(B,A,'Give B a route')) }
  ]
]
B = [
  G = [
    OBL = [ understandingAct(A,DU3)
            answer(B,I2:info_request(A,B,'How can A help B?'))
            understandingAct(B,DU2) ]
    DH  = < acknowledge(B,DU2)
            I2:info_request(A,B,'How can A help B?')
            greet(A,B) >
  ]
  INT = [ answer(B,I2:info_request(A,B,'How can A help B?'))
          understandingAct(B,DU2)
          'Get a route from malvern to edwinstowe' ]
  UDUS = [ DU3 ]
  DU2 = [
    OBL = [ answer(B,I2:info_request(A,B,'How can A help B?')) ]
    DH  = < I2:info_request(A,B,'How can A help B?')
            greet(A,B) >
  ]
  DU3 = [
    OBL = [ accept(A,DI3)→obliged(A,B,'Give B a route')
            address(A,DI3:direct(B,A,'Give B a route')) ]
    DH  = < answer(B,A,I2,'A can help B by providing a route')
            DI3:direct(B,A,'Give B a route') >
    SCP = { scp(B,'A can help B by providing a route') }
    OPT = { address(A,DI3:direct(B,A,'Give B a route')) }
  ]
]
```

removeRec(A.G.OBL, **understandingAct**(B,DU2))
removeRec(A.G.OBL, **understandingAct**(B,DU2))
removeRec(B.G.OBL, **understandingAct**(B,DU2))
removeRec(B.INT, **understandingAct**(B,DU2))
removeRec(B.INT, **answer**(B,I2:**info_request**(A,B,'*How can A help B?*')))

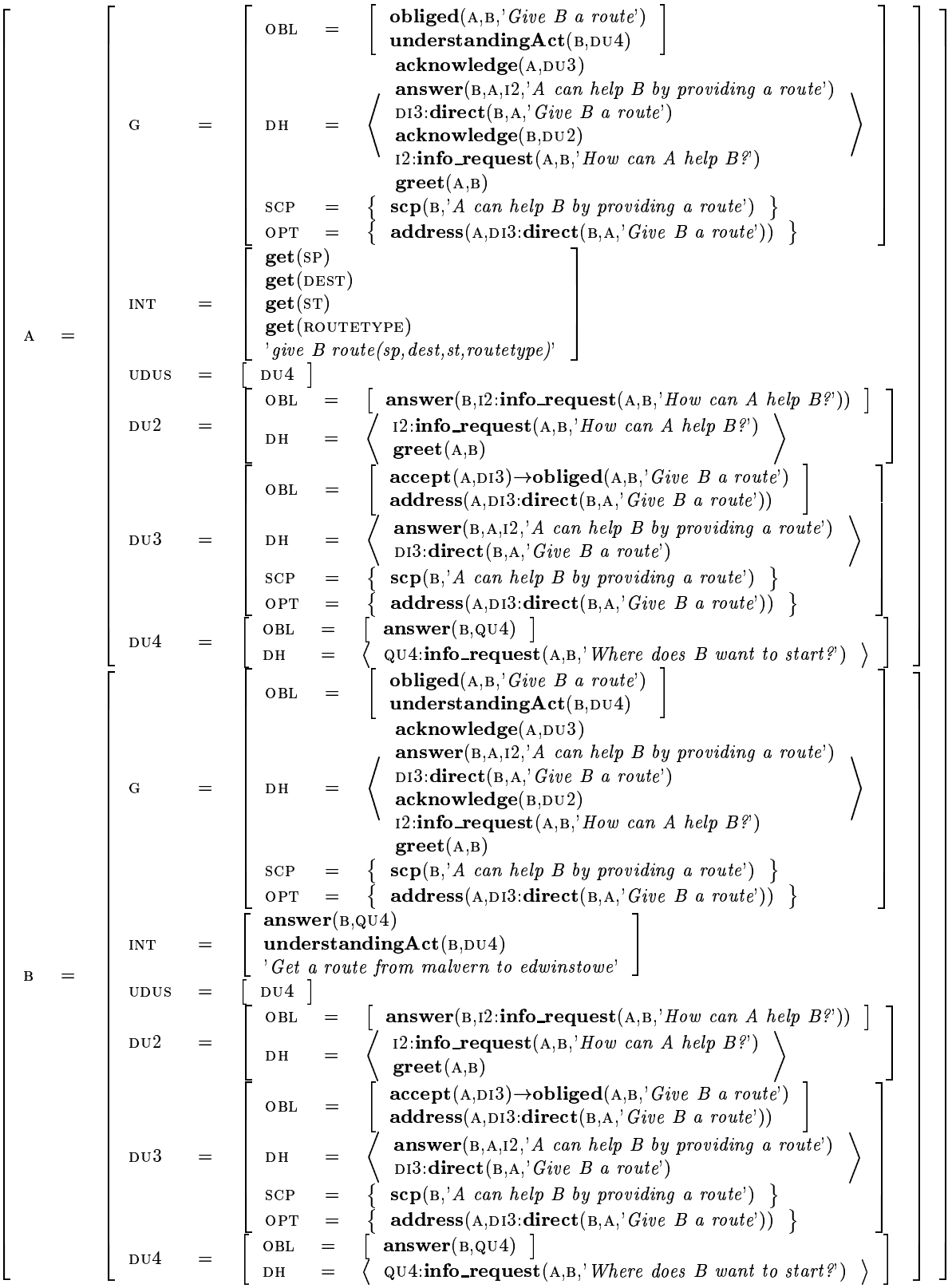
$$
\left[\begin{array}{lll}
\text{A} & = & \left[\begin{array}{lll}
\text{G} & = & \left[\begin{array}{lll}
\text{OBL} & = & \left[\begin{array}{l}
\textbf{understandingAct}(\text{A,DU3}) \\
\textbf{answer}(\text{B,I2:}\textbf{info\_request}(\text{A,B,'}\textit{How can A help B?}\text{'}))
\end{array}\right] \\
\text{DH} & = & \left\langle\begin{array}{l}
\textbf{acknowledge}(\text{B,DU2}) \\
\text{I2:}\textbf{info\_request}(\text{A,B,'}\textit{How can A help B?}\text{'}) \\
\textbf{greet}(\text{A,B})
\end{array}\right\rangle
\end{array}\right] \\
\text{UDUS} & = & \left[\begin{array}{l}\text{DU3}\end{array}\right] \\
\text{DU2} & = & \left[\begin{array}{lll}
\text{OBL} & = & \left[\begin{array}{l}\textbf{answer}(\text{B,I2:}\textbf{info\_request}(\text{A,B,'}\textit{How can A help B?}\text{'}))\end{array}\right] \\
\text{DH} & = & \left\langle\begin{array}{l}
\text{I2:}\textbf{info\_request}(\text{A,B,'}\textit{How can A help B?}\text{'}) \\
\textbf{greet}(\text{A,B})
\end{array}\right\rangle
\end{array}\right] \\
\text{DU3} & = & \left[\begin{array}{lll}
\text{OBL} & = & \left[\begin{array}{l}
\textbf{accept}(\text{A,DI3})\rightarrow\textbf{obliged}(\text{A,B,'}\textit{Give B a route}\text{'}) \\
\textbf{address}(\text{A,DI3:}\textbf{direct}(\text{B,A,'}\textit{Give B a route}\text{'}))
\end{array}\right] \\
\text{DH} & = & \left\langle\begin{array}{l}
\textbf{answer}(\text{B,A,I2,'}\textit{A can help B by providing a route}\text{'}) \\
\text{DI3:}\textbf{direct}(\text{B,A,'}\textit{Give B a route}\text{'})
\end{array}\right\rangle \\
\text{SCP} & = & \left\{\begin{array}{l}\textbf{scp}(\text{B,'}\textit{A can help B by providing a route}\text{'})\end{array}\right\} \\
\text{OPT} & = & \left\{\begin{array}{l}\textbf{address}(\text{A,DI3:}\textbf{direct}(\text{B,A,'}\textit{Give B a route}\text{'}))\end{array}\right\}
\end{array}\right]
\end{array}\right] \\
\text{B} & = & \left[\begin{array}{lll}
\text{G} & = & \left[\begin{array}{lll}
\text{OBL} & = & \left[\begin{array}{l}
\textbf{understandingAct}(\text{A,DU3}) \\
\textbf{answer}(\text{B,I2:}\textbf{info\_request}(\text{A,B,'}\textit{How can A help B?}\text{'}))
\end{array}\right] \\
\text{DH} & = & \left\langle\begin{array}{l}
\textbf{acknowledge}(\text{B,DU2}) \\
\text{I2:}\textbf{info\_request}(\text{A,B,'}\textit{How can A help B?}\text{'}) \\
\textbf{greet}(\text{A,B})
\end{array}\right\rangle
\end{array}\right] \\
\text{INT} & = & \left[\begin{array}{l}\text{'}\textit{Get a route from malvern to edwinstowe}\text{'}\end{array}\right] \\
\text{UDUS} & = & \left[\begin{array}{l}\text{DU3}\end{array}\right] \\
\text{DU2} & = & \left[\begin{array}{lll}
\text{OBL} & = & \left[\begin{array}{l}\textbf{answer}(\text{B,I2:}\textbf{info\_request}(\text{A,B,'}\textit{How can A help B?}\text{'}))\end{array}\right] \\
\text{DH} & = & \left\langle\begin{array}{l}
\text{I2:}\textbf{info\_request}(\text{A,B,'}\textit{How can A help B?}\text{'}) \\
\textbf{greet}(\text{A,B})
\end{array}\right\rangle
\end{array}\right] \\
\text{DU3} & = & \left[\begin{array}{lll}
\text{OBL} & = & \left[\begin{array}{l}
\textbf{accept}(\text{A,DI3})\rightarrow\textbf{obliged}(\text{A,B,'}\textit{Give B a route}\text{'}) \\
\textbf{address}(\text{A,DI3:}\textbf{direct}(\text{B,A,'}\textit{Give B a route}\text{'}))
\end{array}\right] \\
\text{DH} & = & \left\langle\begin{array}{l}
\textbf{answer}(\text{B,A,I2,'}\textit{A can help B by providing a route}\text{'}) \\
\text{DI3:}\textbf{direct}(\text{B,A,'}\textit{Give B a route}\text{'})
\end{array}\right\rangle \\
\text{SCP} & = & \left\{\begin{array}{l}\textbf{scp}(\text{B,'}\textit{A can help B by providing a route}\text{'})\end{array}\right\} \\
\text{OPT} & = & \left\{\begin{array}{l}\textbf{address}(\text{A,DI3:}\textbf{direct}(\text{B,A,'}\textit{Give B a route}\text{'}))\end{array}\right\}
\end{array}\right]
\end{array}\right]
\end{array}\right]
$$

removeRec(A.INT, **address**(A,DI3:**direct**(B,A,'*Give B a route*')))
pushRec(A.INT, '*give B route(sp,dest,st,routetype)*')
pushRec(A.INT, **get**(ROUTETYPE))
pushRec(A.INT, **get**(ST))
pushRec(A.INT, **get**(DEST))
pushRec(A.INT, **get**(SP))
pushRec(A.INT, **understandingAct**(A,DU3))
pushRec(A.INT, **accept**(DI3:**direct**(B,A,'*Give B a route*')))
pushRec(A.INT, **address**(A,DI3:**direct**(B,A,'*Give B a route*')))

$$
\left[\begin{array}{lll}
\multirow{10}{*}{A} & = & \left[\begin{array}{lll}
 & \text{G} & = \left[\begin{array}{lll} \text{OBL} & = & \left[\begin{array}{l} \textbf{understandingAct}(\text{A,DU3}) \\ \textbf{answer}(\text{B,I2:}\textbf{info\_request}(\text{A,B,}\textit{'How can A help B?'})) \end{array}\right] \\ \text{DH} & = & \left\langle\begin{array}{l} \textbf{acknowledge}(\text{B,DU2}) \\ \text{I2:}\textbf{info\_request}(\text{A,B,}\textit{'How can A help B?'}) \\ \textbf{greet}(\text{A,B}) \end{array}\right\rangle \end{array}\right] \\
 & \text{INT} & = \left[\begin{array}{l} \textbf{address}(\text{A,DI3:}\textbf{direct}(\text{B,A,}\textit{'Give B a route'})) \\ \textbf{accept}(\text{DI3:}\textbf{direct}(\text{B,A,}\textit{'Give B a route'})) \\ \textbf{understandingAct}(\text{A,DU3}) \\ \textbf{get}(\text{SP}) \\ \textbf{get}(\text{DEST}) \\ \textbf{get}(\text{ST}) \\ \textbf{get}(\text{ROUTETYPE}) \\ \textit{'give B route(sp,dest,st,routetype)'} \end{array}\right] \\
 & \text{UDUS} & = \left[\begin{array}{l} \text{DU3} \end{array}\right] \\
 & \text{DU2} & = \left[\begin{array}{lll} \text{OBL} & = & \left[\begin{array}{l} \textbf{answer}(\text{B,I2:}\textbf{info\_request}(\text{A,B,}\textit{'How can A help B?'})) \end{array}\right] \\ \text{DH} & = & \left\langle\begin{array}{l} \text{I2:}\textbf{info\_request}(\text{A,B,}\textit{'How can A help B?'}) \\ \textbf{greet}(\text{A,B}) \end{array}\right\rangle \end{array}\right] \\
 & \text{DU3} & = \left[\begin{array}{lll} \text{OBL} & = & \left[\begin{array}{l} \textbf{accept}(\text{A,DI3})\rightarrow\textbf{obliged}(\text{A,B,}\textit{'Give B a route'}) \\ \textbf{address}(\text{A,DI3:}\textbf{direct}(\text{B,A,}\textit{'Give B a route'})) \end{array}\right] \\ \text{DH} & = & \left\langle\begin{array}{l} \textbf{answer}(\text{B,A,I2,}\textit{'A can help B by providing a route'}) \\ \text{DI3:}\textbf{direct}(\text{B,A,}\textit{'Give B a route'}) \end{array}\right\rangle \\ \text{SCP} & = & \left\{\begin{array}{l} \textbf{scp}(\text{B,}\textit{'A can help B by providing a route'}) \end{array}\right\} \\ \text{OPT} & = & \left\{\begin{array}{l} \textbf{address}(\text{A,DI3:}\textbf{direct}(\text{B,A,}\textit{'Give B a route'})) \end{array}\right\} \end{array}\right] \\
\multirow{10}{*}{B} & = & \left[\begin{array}{lll}
 & \text{G} & = \left[\begin{array}{lll} \text{OBL} & = & \left[\begin{array}{l} \textbf{understandingAct}(\text{A,DU3}) \\ \textbf{answer}(\text{B,I2:}\textbf{info\_request}(\text{A,B,}\textit{'How can A help B?'})) \end{array}\right] \\ \text{DH} & = & \left\langle\begin{array}{l} \textbf{acknowledge}(\text{B,DU2}) \\ \text{I2:}\textbf{info\_request}(\text{A,B,}\textit{'How can A help B?'}) \\ \textbf{greet}(\text{A,B}) \end{array}\right\rangle \end{array}\right] \\
 & \text{INT} & = \left[\begin{array}{l} \textit{'Get a route from malvern to edwinstowe'} \end{array}\right] \\
 & \text{UDUS} & = \left[\begin{array}{l} \text{DU3} \end{array}\right] \\
 & \text{DU2} & = \left[\begin{array}{lll} \text{OBL} & = & \left[\begin{array}{l} \textbf{answer}(\text{B,I2:}\textbf{info\_request}(\text{A,B,}\textit{'How can A help B?'})) \end{array}\right] \\ \text{DH} & = & \left\langle\begin{array}{l} \text{I2:}\textbf{info\_request}(\text{A,B,}\textit{'How can A help B?'}) \\ \textbf{greet}(\text{A,B}) \end{array}\right\rangle \end{array}\right] \\
 & \text{DU3} & = \left[\begin{array}{lll} \text{OBL} & = & \left[\begin{array}{l} \textbf{accept}(\text{A,DI3})\rightarrow\textbf{obliged}(\text{A,B,}\textit{'Give B a route'}) \\ \textbf{address}(\text{A,DI3:}\textbf{direct}(\text{B,A,}\textit{'Give B a route'})) \end{array}\right] \\ \text{DH} & = & \left\langle\begin{array}{l} \textbf{answer}(\text{B,A,I2,}\textit{'A can help B by providing a route'}) \\ \text{DI3:}\textbf{direct}(\text{B,A,}\textit{'Give B a route'}) \end{array}\right\rangle \\ \text{SCP} & = & \left\{\begin{array}{l} \textbf{scp}(\text{B,}\textit{'A can help B by providing a route'}) \end{array}\right\} \\ \text{OPT} & = & \left\{\begin{array}{l} \textbf{address}(\text{A,DI3:}\textbf{direct}(\text{B,A,}\textit{'Give B a route'})) \end{array}\right\} \end{array}\right]
\end{array}\right]
$$

(4)
A <Where would you like to start your journey?>

add_fieldRec(A.DU4, [])
add_fieldRec(B.DU4, [])
pushRec(A.UDUS, DU4)
pushRec(B.UDUS, DU4)
peRec(A.G, A.DU3)
peRec(B.G, B.DU3)
removeRec(A.UDUS, DU3)
removeRec(B.UDUS, DU3)
pushRec(A.G.DH, **acknowledge**(A,DU3))
pushRec(B.G.DH, **acknowledge**(A,DU3))
pushRec(A.DU4.DH, QU4:**info_request**(A,B,'*Where does B want to start?*'))
pushRec(B.DU4.DH, QU4:**info_request**(A,B,'*Where does B want to start?*'))
pushRec(A.G.OBL, **understandingAct**(B,DU4))
pushRec(B.G.OBL, **understandingAct**(B,DU4))
pushRec(A.DU4.OBL, **answer**(B,QU4))
pushRec(B.DU4.OBL, **answer**(B,QU4))
removeRec(A.G.OBL, **understandingAct**(A,DU3))
removeRec(B.G.OBL, **understandingAct**(A,DU3))
removeRec(A.G.OBL, **answer**(B,I2:**info_request**(A,B,'*How can A help B?*')))
removeRec(B.G.OBL, **answer**(B,I2:**info_request**(A,B,'*How can A help B?*')))
removeRec(A.G.OBL, **address**(A,DI3:**direct**(B,A,'*Give B a route*')))
removeRec(B.G.OBL, **address**(A,DI3:**direct**(B,A,'*Give B a route*')))
removeRec(A.G.OBL, **accept**(A,DI3)→**obliged**(A,B,'*Give B a route*'))
removeRec(B.G.OBL, **accept**(A,DI3)→**obliged**(A,B,'*Give B a route*'))
pushRec(A.G.OBL, **obliged**(A,B,'*Give B a route*'))
pushRec(B.G.OBL, **obliged**(A,B,'*Give B a route*'))
pushRec(B.INT, **understandingAct**(B,DU4))
pushRec(B.INT, **answer**(B,QU4))
removeRec(A.INT, **understandingAct**(A,DU3))
removeRec(A.INT, **address**(A,DI3:**direct**(B,A,'*Give B a route*')))
removeRec(A.INT, **accept**(DI3:**direct**(B,A,'*Give B a route*')))


A =
G =
OBL = obliged(A,B,'Give B a route')
understandingAct(B,DU4)
acknowledge(A,DU3)
DH = answer(B,A,I2,'A can help B by providing a route')
DI3:direct(B,A,'Give B a route')
acknowledge(B,DU2)
I2:info_request(A,B,'How can A help B?')
greet(A,B)
SCP = scp(B,'A can help B by providing a route')
OPT = address(A,DI3:direct(B,A,'Give B a route'))
INT = get(SP)
get(DEST)
get(ST)
get(ROUTETYPE)
'give B route(sp,dest,st,routetype)'
UDUS = DU4
DU2 = OBL = answer(B,I2:info_request(A,B,'How can A help B?'))
DH = I2:info_request(A,B,'How can A help B?')
greet(A,B)
DU3 = OBL = accept(A,DI3)→obliged(A,B,'Give B a route')
address(A,DI3:direct(B,A,'Give B a route'))
DH = answer(B,A,I2,'A can help B by providing a route')
DI3:direct(B,A,'Give B a route')
SCP = scp(B,'A can help B by providing a route')
OPT = address(A,DI3:direct(B,A,'Give B a route'))
DU4 = OBL = answer(B,QU4)
DH = QU4:info_request(A,B,'Where does B want to start?')
B =
G =
OBL = obliged(A,B,'Give B a route')
understandingAct(B,DU4)
acknowledge(A,DU3)
DH = answer(B,A,I2,'A can help B by providing a route')
DI3:direct(B,A,'Give B a route')
acknowledge(B,DU2)
I2:info_request(A,B,'How can A help B?')
greet(A,B)
SCP = scp(B,'A can help B by providing a route')
OPT = address(A,DI3:direct(B,A,'Give B a route'))
INT = answer(B,QU4)
understandingAct(B,DU4)
'Get a route from malvern to edwinstowe'
UDUS = DU4
DU2 = OBL = answer(B,I2:info_request(A,B,'How can A help B?'))
DH = I2:info_request(A,B,'How can A help B?')
greet(A,B)
DU3 = OBL = accept(A,DI3)→obliged(A,B,'Give B a route')
address(A,DI3:direct(B,A,'Give B a route'))
DH = answer(B,A,I2,'A can help B by providing a route')
DI3:direct(B,A,'Give B a route')
SCP = scp(B,'A can help B by providing a route')
OPT = address(A,DI3:direct(B,A,'Give B a route'))
DU4 = OBL = answer(B,QU4)
DH = QU4:info_request(A,B,'Where does B want to start?')

(5)
B <Malvern.>

add_fieldRec(A.DU5, [])
add_fieldRec(B.DU5, [])
pushRec(A.UDUS, DU5)
pushRec(B.UDUS, DU5)
pushRec(A.G.OBL, **understandingAct**(A,DU5))
pushRec(B.G.OBL, **understandingAct**(A,DU5))
peRec(A.G, A.DU4)
peRec(B.G, B.DU4)
removeRec(A.UDUS, DU4)
removeRec(B.UDUS, DU4)
pushRec(A.G.DH, **acknowledge**(B,DU4))
pushRec(B.G.DH, **acknowledge**(B,DU4))
pushRec(A.DU5.DH, **answer**(B,A,'*Where does B want to start?*','*SP=Malvern*'))
pushRec(B.DU5.DH, **answer**(B,A,'*Where does B want to start?*','*SP=Malvern*'))
addRec(A.DU5.SCP, **scp**(B,'*SP=Malvern*'))
addRec(B.DU5.SCP, **scp**(B,'*SP=Malvern*'))
removeRec(A.G.OBL, **understandingAct**(B,DU4))
removeRec(B.G.OBL, **understandingAct**(B,DU4))
removeRec(B.INT, **understandingAct**(B,DU4))
removeRec(B.INT, **answer**(B,QU4))
pushRec(A.INT, **understandingAct**(A,DU5))
pushRec(A.INT, **check**(**answer**(B,A,'*Where does B want to start?*','*SP=Malvern*')))

$$
\left[\begin{array}{lll}
\text{A} = & \text{G} = & \left[\begin{array}{ll}
\text{OBL} = & \left[\begin{array}{l}
\textbf{answer}(\text{B,QU4}) \\
\textbf{understandingAct}(\text{A,DU5}) \\
\textbf{obliged}(\text{A,B,}'\textit{Give B a route}')
\end{array}\right] \\
\text{DH} = & \left\langle\begin{array}{l}
\textbf{acknowledge}(\text{B,DU4}) \\
\text{QU4:}\textbf{info\_request}(\text{A,B,}'\textit{Where does B want to start?}') \\
\textbf{acknowledge}(\text{A,DU3}) \\
\textbf{answer}(\text{B,A,I2,}'\textit{A can help B by providing a route}') \\
\text{DI3:}\textbf{direct}(\text{B,A,}'\textit{Give B a route}') \\
\textbf{acknowledge}(\text{B,DU2}) \\
\text{I2:}\textbf{info\_request}(\text{A,B,}'\textit{How can A help B?}') \\
\textbf{greet}(\text{A,B})
\end{array}\right\rangle \\
\text{SCP} = & \{\ \textbf{scp}(\text{B,}'\textit{A can help B by providing a route}')\ \} \\
\text{OPT} = & \{\ \textbf{address}(\text{A,DI3:}\textbf{direct}(\text{B,A,}'\textit{Give B a route}'))\ \}
\end{array}\right] \\
& \text{INT} = & \left[\begin{array}{l}
\textbf{check}(\textbf{answer}(\text{B,A,}'\textit{Where does B want to start?}'\text{,}'\textit{SP=Malvern}')) \\
\textbf{understandingAct}(\text{A,DU5}) \\
\textbf{get}(\text{SP}) \\
\textbf{get}(\text{DEST}) \\
\textbf{get}(\text{ST}) \\
\textbf{get}(\text{ROUTETYPE}) \\
'\textit{give B route(sp,dest,st,routetype)}'
\end{array}\right] \\
& \text{UDUS} = & [\ \text{DU5}\ ] \\
& \text{DU2} = & \left[\begin{array}{ll}
\text{OBL} = & [\ \textbf{answer}(\text{B,I2:}\textbf{info\_request}(\text{A,B,}'\textit{How can A help B?}'))\ ] \\
\text{DH} = & \left\langle\begin{array}{l}
\text{I2:}\textbf{info\_request}(\text{A,B,}'\textit{How can A help B?}') \\
\textbf{greet}(\text{A,B})
\end{array}\right\rangle
\end{array}\right] \\
& \text{DU3} = & \left[\begin{array}{ll}
\text{OBL} = & \left[\begin{array}{l}
\textbf{accept}(\text{A,DI3})\rightarrow\textbf{obliged}(\text{A,B,}'\textit{Give B a route}') \\
\textbf{address}(\text{A,DI3:}\textbf{direct}(\text{B,A,}'\textit{Give B a route}'))
\end{array}\right] \\
\text{DH} = & \left\langle\begin{array}{l}
\textbf{answer}(\text{B,A,I2,}'\textit{A can help B by providing a route}') \\
\text{DI3:}\textbf{direct}(\text{B,A,}'\textit{Give B a route}')
\end{array}\right\rangle \\
\text{SCP} = & \{\ \textbf{scp}(\text{B,}'\textit{A can help B by providing a route}')\ \} \\
\text{OPT} = & \{\ \textbf{address}(\text{A,DI3:}\textbf{direct}(\text{B,A,}'\textit{Give B a route}'))\ \}
\end{array}\right] \\
& \text{DU4} = & \left[\begin{array}{ll}
\text{OBL} = & [\ \textbf{answer}(\text{B,QU4})\ ] \\
\text{DH} = & \langle\ \text{QU4:}\textbf{info\_request}(\text{A,B,}'\textit{Where does B want to start?}')\ \rangle
\end{array}\right] \\
& \text{DU5} = & \left[\begin{array}{ll}
\text{DH} = & \langle\ \textbf{answer}(\text{B,A,}'\textit{Where does B want to start?}'\text{,}'\textit{SP=Malvern}')\ \rangle \\
\text{SCP} = & \{\ \textbf{scp}(\text{B,}'\textit{SP=Malvern}')\ \}
\end{array}\right]
\end{array}\right]
$$

(contd. on next page)

$$
\left[
\begin{array}{lll}
\textsc{b} & = &
\left[
\begin{array}{lll}
\textsc{g} & = &
\left[
\begin{array}{lll}
\textsc{obl} & = & \left[
\begin{array}{l}
\textbf{answer}(\textsc{b},\textsc{qu}4) \\
\textbf{understandingAct}(\textsc{a},\textsc{du}5) \\
\textbf{obliged}(\textsc{a},\textsc{b},\text{'}\textit{Give B a route}\text{'})
\end{array}
\right] \\
\textsc{dh} & = & \left\langle
\begin{array}{l}
\textbf{acknowledge}(\textsc{b},\textsc{du}4) \\
\textsc{qu}4{:}\textbf{info\_request}(\textsc{a},\textsc{b},\text{'}\textit{Where does B want to start?}\text{'}) \\
\textbf{acknowledge}(\textsc{a},\textsc{du}3) \\
\textbf{answer}(\textsc{b},\textsc{a},\textsc{i}2,\text{'}\textit{A can help B by providing a route}\text{'}) \\
\textsc{di}3{:}\textbf{direct}(\textsc{b},\textsc{a},\text{'}\textit{Give B a route}\text{'}) \\
\textbf{acknowledge}(\textsc{b},\textsc{du}2) \\
\textsc{i}2{:}\textbf{info\_request}(\textsc{a},\textsc{b},\text{'}\textit{How can A help B?}\text{'}) \\
\textbf{greet}(\textsc{a},\textsc{b})
\end{array}
\right\rangle \\
\textsc{scp} & = & \left\{ \textbf{scp}(\textsc{b},\text{'}\textit{A can help B by providing a route}\text{'}) \right\} \\
\textsc{opt} & = & \left\{ \textbf{address}(\textsc{a},\textsc{di}3{:}\textbf{direct}(\textsc{b},\textsc{a},\text{'}\textit{Give B a route}\text{'})) \right\}
\end{array}
\right] \\
\textsc{int} & = & \left[\text{'}\textit{Get a route from malvern to edwinstowe}\text{'} \right] \\
\textsc{udus} & = & \left[\textsc{du}5 \right] \\
\textsc{du}2 & = &
\left[
\begin{array}{lll}
\textsc{obl} & = & \left[\textbf{answer}(\textsc{b},\textsc{i}2{:}\textbf{info\_request}(\textsc{a},\textsc{b},\text{'}\textit{How can A help B?}\text{'})) \right] \\
\textsc{dh} & = & \left\langle
\begin{array}{l}
\textsc{i}2{:}\textbf{info\_request}(\textsc{a},\textsc{b},\text{'}\textit{How can A help B?}\text{'}) \\
\textbf{greet}(\textsc{a},\textsc{b})
\end{array}
\right\rangle
\end{array}
\right] \\
\textsc{du}3 & = &
\left[
\begin{array}{lll}
\textsc{obl} & = & \left[
\begin{array}{l}
\textbf{accept}(\textsc{a},\textsc{di}3)\rightarrow\textbf{obliged}(\textsc{a},\textsc{b},\text{'}\textit{Give B a route}\text{'}) \\
\textbf{address}(\textsc{a},\textsc{di}3{:}\textbf{direct}(\textsc{b},\textsc{a},\text{'}\textit{Give B a route}\text{'}))
\end{array}
\right] \\
\textsc{dh} & = & \left\langle
\begin{array}{l}
\textbf{answer}(\textsc{b},\textsc{a},\textsc{i}2,\text{'}\textit{A can help B by providing a route}\text{'}) \\
\textsc{di}3{:}\textbf{direct}(\textsc{b},\textsc{a},\text{'}\textit{Give B a route}\text{'})
\end{array}
\right\rangle \\
\textsc{scp} & = & \left\{ \textbf{scp}(\textsc{b},\text{'}\textit{A can help B by providing a route}\text{'}) \right\} \\
\textsc{opt} & = & \left\{ \textbf{address}(\textsc{a},\textsc{di}3{:}\textbf{direct}(\textsc{b},\textsc{a},\text{'}\textit{Give B a route}\text{'})) \right\}
\end{array}
\right] \\
\textsc{du}4 & = &
\left[
\begin{array}{lll}
\textsc{obl} & = & \left[\textbf{answer}(\textsc{b},\textsc{qu}4) \right] \\
\textsc{dh} & = & \left\langle \textsc{qu}4{:}\textbf{info\_request}(\textsc{a},\textsc{b},\text{'}\textit{Where does B want to start?}\text{'}) \right\rangle
\end{array}
\right] \\
\textsc{du}5 & = &
\left[
\begin{array}{lll}
\textsc{dh} & = & \left\langle \textbf{answer}(\textsc{b},\textsc{a},\text{'}\textit{Where does B want to start?}\text{'},\text{'}\textit{SP=Malvern}\text{'}) \right\rangle \\
\textsc{scp} & = & \left\{ \textbf{scp}(\textsc{b},\text{'}\textit{SP=Malvern}\text{'}) \right\}
\end{array}
\right]
\end{array}
\right]
\end{array}
\right]
$$

(6)
A <Starting in Great Malvern.>

add_fieldRec(A.DU6, [])
add_fieldRec(B.DU6, [])
pushRec(A.UDUS, DU6)
pushRec(B.UDUS, DU6)
pushRec(A.G.OBL, **understandingAct**(B,DU6))
pushRec(B.G.OBL, **understandingAct**(B,DU6))
peRec(A.G, A.DU5)
peRec(B.G, B.DU5)
removeRec(A.UDUS, DU5)
removeRec(B.UDUS, DU5)
pushRec(A.G.DH, **acknowledge**(B,DU5))
pushRec(B.G.DH, **acknowledge**(B,DU5))
pushRec(A.DU6.DH, CH6:**check**(A,B,'*SP= Great Malvern*'))
pushRec(B.DU6.DH, CH6:**check**(A,B,'*SP= Great Malvern*'))
pushRec(A.DU6.OBL, **answer**(B,CH6))
pushRec(B.DU6.OBL, **answer**(B,CH6))
addRec(A.DU6.SCP, **agree**(B,CH6)→**scp**(A,'*SP= Great Malvern*'))
addRec(B.DU6.SCP, **agree**(B,CH6)→**scp**(A,'*SP= Great Malvern*'))
removeRec(A.G.OBL, **understandingAct**(A,DU5))
removeRec(B.G.OBL, **understandingAct**(A,DU5))
removeRec(A.G.OBL, **answer**(B,QU4))
removeRec(B.G.OBL, **answer**(B,QU4))
removeRec(A.INT, **understandingAct**(A,DU5))
removeRec(A.INT, **check**(**answer**(B,A,'*Where does B want to start?*','*SP=Malvern*')))
pushRec(B.INT, **understandingAct**(B,DU6))
pushRec(B.INT, **agree**(B,CH6))

$$
\left[\begin{array}{lll}
\text{A} = & \text{G} = & \left[\begin{array}{ll}
\text{OBL} = & \left[\begin{array}{l} \textbf{understandingAct}(\text{B,DU6}) \\ \textbf{obliged}(\text{A,B,}'\textit{Give B a route}') \end{array}\right] \\
\text{DH} = & \left\langle\begin{array}{l} \textbf{acknowledge}(\text{B,DU5}) \\ \textbf{answer}(\text{B,A,}'\textit{Where does B want to start?}'\text{,}'\textit{SP=Malvern}') \\ \textbf{acknowledge}(\text{B,DU4}) \\ \text{QU4:}\textbf{info\_request}(\text{A,B,}'\textit{Where does B want to start?}') \\ \textbf{acknowledge}(\text{A,DU3}) \\ \textbf{answer}(\text{B,A,I2,}'\textit{A can help B by providing a route}') \\ \text{DI3:}\textbf{direct}(\text{B,A,}'\textit{Give B a route}') \\ \textbf{acknowledge}(\text{B,DU2}) \\ \text{I2:}\textbf{info\_request}(\text{A,B,}'\textit{How can A help B?}') \\ \textbf{greet}(\text{A,B}) \end{array}\right\rangle \\
\text{SCP} = & \left\{\begin{array}{l} \textbf{scp}(\text{B,}'\textit{SP=Malvern}') \\ \textbf{scp}(\text{B,}'\textit{A can help B by providing a route}') \end{array}\right\} \\
\text{OPT} = & \left\{ \textbf{address}(\text{A,DI3:}\textbf{direct}(\text{B,A,}'\textit{Give B a route}')) \right\}
\end{array}\right] \\
& \text{INT} = & \left[\begin{array}{l} \textbf{get}(\text{SP}) \\ \textbf{get}(\text{DEST}) \\ \textbf{get}(\text{ST}) \\ \textbf{get}(\text{ROUTETYPE}) \\ '\textit{give B route(sp,dest,st,routetype)}' \end{array}\right] \\
& \text{UDUS} = & \left[\text{DU6} \right] \\
& \text{DU2} = & \left[\begin{array}{ll}
\text{OBL} = & \left[\textbf{answer}(\text{B,I2:}\textbf{info\_request}(\text{A,B,}'\textit{How can A help B?}')) \right] \\
\text{DH} = & \left\langle\begin{array}{l} \text{I2:}\textbf{info\_request}(\text{A,B,}'\textit{How can A help B?}') \\ \textbf{greet}(\text{A,B}) \end{array}\right\rangle
\end{array}\right] \\
& \text{DU3} = & \left[\begin{array}{ll}
\text{OBL} = & \left[\begin{array}{l} \textbf{accept}(\text{A,DI3})\rightarrow\textbf{obliged}(\text{A,B,}'\textit{Give B a route}') \\ \textbf{address}(\text{A,DI3:}\textbf{direct}(\text{B,A,}'\textit{Give B a route}')) \end{array}\right] \\
\text{DH} = & \left\langle\begin{array}{l} \textbf{answer}(\text{B,A,I2,}'\textit{A can help B by providing a route}') \\ \text{DI3:}\textbf{direct}(\text{B,A,}'\textit{Give B a route}') \end{array}\right\rangle \\
\text{SCP} = & \left\{ \textbf{scp}(\text{B,}'\textit{A can help B by providing a route}') \right\} \\
\text{OPT} = & \left\{ \textbf{address}(\text{A,DI3:}\textbf{direct}(\text{B,A,}'\textit{Give B a route}')) \right\}
\end{array}\right] \\
& \text{DU4} = & \left[\begin{array}{ll}
\text{OBL} = & \left[\textbf{answer}(\text{B,QU4}) \right] \\
\text{DH} = & \left\langle \text{QU4:}\textbf{info\_request}(\text{A,B,}'\textit{Where does B want to start?}') \right\rangle
\end{array}\right] \\
& \text{DU5} = & \left[\begin{array}{ll}
\text{DH} = & \left\langle \textbf{answer}(\text{B,A,}'\textit{Where does B want to start?}'\text{,}'\textit{SP=Malvern}') \right\rangle \\
\text{SCP} = & \left\{ \textbf{scp}(\text{B,}'\textit{SP=Malvern}') \right\}
\end{array}\right] \\
& \text{DU6} = & \left[\begin{array}{ll}
\text{OBL} = & \left[\textbf{answer}(\text{B,CH6}) \right] \\
\text{DH} = & \left\langle \text{CH6:}\textbf{check}(\text{A,B,}'\textit{SP= Great Malvern}') \right\rangle \\
\text{SCP} = & \left\{ \textbf{agree}(\text{B,CH6})\rightarrow\textbf{scp}(\text{A,}'\textit{SP= Great Malvern}') \right\}
\end{array}\right]
\end{array}\right]
$$

(contd. on next page)


B =
G =
OBL =
understandingAct(B,DU6)
obliged(A,B,'Give B a route')
DH =
acknowledge(B,DU5)
answer(B,A,'Where does B want to start?','SP=Malvern')
acknowledge(B,DU4)
QU4:info_request(A,B,'Where does B want to start?')
acknowledge(A,DU3)
answer(B,A,I2,'A can help B by providing a route')
DI3:direct(B,A,'Give B a route')
acknowledge(B,DU2)
I2:info_request(A,B,'How can A help B?')
greet(A,B)
SCP =
scp(B,'SP=Malvern')
scp(B,'A can help B by providing a route')
OPT =
address(A,DI3:direct(B,A,'Give B a route'))
INT =
agree(B,CH6)
understandingAct(B,DU6)
'Get a route from malvern to edwinstowe'
UDUS =
DU6
DU2 =
OBL =
answer(B,I2:info_request(A,B,'How can A help B?'))
DH =
I2:info_request(A,B,'How can A help B?')
greet(A,B)
DU3 =
OBL =
accept(A,DI3)→obliged(A,B,'Give B a route')
address(A,DI3:direct(B,A,'Give B a route'))
DH =
answer(B,A,I2,'A can help B by providing a route')
DI3:direct(B,A,'Give B a route')
SCP =
scp(B,'A can help B by providing a route')
OPT =
address(A,DI3:direct(B,A,'Give B a route'))
DU4 =
OBL =
answer(B,QU4)
DH =
QU4:info_request(A,B,'Where does B want to start?')
DU5 =
DH =
answer(B,A,'Where does B want to start?','SP=Malvern')
SCP =
scp(B,'SP=Malvern')
DU6 =
OBL =
answer(B,CH6)
DH =
CH6:check(A,B,'SP= Great Malvern')
SCP =
agree(B,CH6)→scp(A,'SP= Great Malvern')

(7)
B <Yes.>

add_fieldRec(A.DU7, [])
add_fieldRec(B.DU7, [])
pushRec(A.UDUS, DU7)
pushRec(B.UDUS, DU7)
pushRec(A.G.OBL, **understandingAct**(A,DU7))
pushRec(B.G.OBL, **understandingAct**(A,DU7))
peRec(A.G, A.DU6)
peRec(B.G, B.DU6)
removeRec(A.UDUS, DU6)
removeRec(B.UDUS, DU6)
pushRec(A.G.DH, **acknowledge**(B,DU6))
pushRec(B.G.DH, **acknowledge**(B,DU6))
pushRec(A.DU7.DH, **agree**(B,A,CH6))
pushRec(B.DU7.DH, **agree**(B,A,CH6))
pushRec(B.DU7.DH, **agree**(B,A,CH6))
addRec(A.DU7.SCP, **scp**(B,'*SP= Great Malvern*'))
addRec(B.DU7.SCP, **scp**(B,'*SP= Great Malvern*'))
removeRec(A.G.OBL, **understandingAct**(B,DU6))
removeRec(B.G.OBL, **understandingAct**(B,DU6))
removeRec(A.INT, **get**(SP))

$$
\begin{array}{l}
\text{A} = \\
\quad \text{G} = \\
\qquad \text{OBL} = \\
\qquad \text{DH} = \\
\qquad \text{SCP} = \\
\qquad \text{OPT} = \\
\quad \text{INT} = \\
\quad \text{UDUS} = \\
\quad \text{DU2} = \\
\qquad \text{OBL} = \\
\qquad \text{DH} = \\
\quad \text{DU3} = \\
\qquad \text{OBL} = \\
\qquad \text{DH} = \\
\qquad \text{SCP} = \\
\qquad \text{OPT} = \\
\quad \text{DU4} = \\
\qquad \text{OBL} = \\
\qquad \text{DH} = \\
\quad \text{DU5} = \\
\qquad \text{DH} = \\
\qquad \text{SCP} = \\
\quad \text{DU6} = \\
\qquad \text{OBL} = \\
\qquad \text{DH} = \\
\qquad \text{SCP} = \\
\quad \text{DU7} = \\
\qquad \text{DH} = \\
\qquad \text{SCP} =
\end{array}
\left[\begin{array}{l}
\left[\begin{array}{l}
\left[\begin{array}{l}
\textbf{answer}(\text{B,CH6}) \\
\textbf{understandingAct}(\text{A,DU7}) \\
\textbf{obliged}(\text{A,B,}\textit{'Give B a route'})
\end{array}\right] \\
\left\langle\begin{array}{l}
\textbf{acknowledge}(\text{B,DU6}) \\
\text{CH6:}\textbf{check}(\text{A,B,}\textit{'SP= Great Malvern'}) \\
\textbf{acknowledge}(\text{B,DU5}) \\
\textbf{answer}(\text{B,A,}\textit{'Where does B want to start?'},\textit{'SP=Malvern'}) \\
\textbf{acknowledge}(\text{B,DU4}) \\
\text{QU4:}\textbf{info\_request}(\text{A,B,}\textit{'Where does B want to start?'}) \\
\textbf{acknowledge}(\text{A,DU3}) \\
\textbf{answer}(\text{B,A,I2,}\textit{'A can help B by providing a route'}) \\
\text{DI3:}\textbf{direct}(\text{B,A,}\textit{'Give B a route'}) \\
\textbf{acknowledge}(\text{B,DU2}) \\
\text{I2:}\textbf{info\_request}(\text{A,B,}\textit{'How can A help B?'}) \\
\textbf{greet}(\text{A,B})
\end{array}\right\rangle \\
\left\{\begin{array}{l}
\textbf{agree}(\text{B,CH6}) \rightarrow \textbf{scp}(\text{A,}\textit{'SP= Great Malvern'}) \\
\textbf{scp}(\text{B,}\textit{'SP=Malvern'}) \\
\textbf{scp}(\text{B,}\textit{'A can help B by providing a route'})
\end{array}\right\} \\
\left\{\textbf{address}(\text{A,DI3:}\textbf{direct}(\text{B,A,}\textit{'Give B a route'}))\right\}
\end{array}\right] \\
\left[\begin{array}{l}
\textbf{get}(\text{DEST}) \\
\textbf{get}(\text{ST}) \\
\textbf{get}(\text{ROUTETYPE}) \\
\textit{'give B route(sp,dest,st,routetype)'}
\end{array}\right] \\
\left[\text{DU7}\right] \\
\left[\begin{array}{l}
\left[\textbf{answer}(\text{B,I2:}\textbf{info\_request}(\text{A,B,}\textit{'How can A help B?'}))\right] \\
\left\langle\begin{array}{l}
\text{I2:}\textbf{info\_request}(\text{A,B,}\textit{'How can A help B?'}) \\
\textbf{greet}(\text{A,B})
\end{array}\right\rangle
\end{array}\right] \\
\left[\begin{array}{l}
\left[\begin{array}{l}
\textbf{accept}(\text{A,DI3}) \rightarrow \textbf{obliged}(\text{A,B,}\textit{'Give B a route'}) \\
\textbf{address}(\text{A,DI3:}\textbf{direct}(\text{B,A,}\textit{'Give B a route'}))
\end{array}\right] \\
\left\langle\begin{array}{l}
\textbf{answer}(\text{B,A,I2,}\textit{'A can help B by providing a route'}) \\
\text{DI3:}\textbf{direct}(\text{B,A,}\textit{'Give B a route'})
\end{array}\right\rangle \\
\left\{\textbf{scp}(\text{B,}\textit{'A can help B by providing a route'})\right\} \\
\left\{\textbf{address}(\text{A,DI3:}\textbf{direct}(\text{B,A,}\textit{'Give B a route'}))\right\}
\end{array}\right] \\
\left[\begin{array}{l}
\left[\textbf{answer}(\text{B,QU4})\right] \\
\left\langle\text{QU4:}\textbf{info\_request}(\text{A,B,}\textit{'Where does B want to start?'})\right\rangle
\end{array}\right] \\
\left[\begin{array}{l}
\left\langle\textbf{answer}(\text{B,A,}\textit{'Where does B want to start?'},\textit{'SP=Malvern'})\right\rangle \\
\left\{\textbf{scp}(\text{B,}\textit{'SP=Malvern'})\right\}
\end{array}\right] \\
\left[\begin{array}{l}
\left[\textbf{answer}(\text{B,CH6})\right] \\
\left\langle\text{CH6:}\textbf{check}(\text{A,B,}\textit{'SP= Great Malvern'})\right\rangle \\
\left\{\textbf{agree}(\text{B,CH6}) \rightarrow \textbf{scp}(\text{A,}\textit{'SP= Great Malvern'})\right\}
\end{array}\right] \\
\left[\begin{array}{l}
\left\langle\textbf{agree}(\text{B,A,CH6})\right\rangle \\
\left\{\textbf{scp}(\text{B,}\textit{'SP= Great Malvern'})\right\}
\end{array}\right]
\end{array}\right]
$$

(contd. on next page)

$$
\left[\begin{array}{lll}
\text{B} & = & \left[\begin{array}{lll}
\text{G} & = & \left[\begin{array}{lll}
\text{OBL} & = & \left[\begin{array}{l}
\textbf{answer}(\text{B,CH6}) \\
\textbf{understandingAct}(\text{A,DU7}) \\
\textbf{obliged}(\text{A,B,}'\textit{Give B a route}')
\end{array}\right] \\
\text{DH} & = & \left\langle\begin{array}{l}
\textbf{acknowledge}(\text{B,DU6}) \\
\text{CH6:}\textbf{check}(\text{A,B,}'\textit{SP= Great Malvern}') \\
\textbf{acknowledge}(\text{B,DU5}) \\
\textbf{answer}(\text{B,A,}'\textit{Where does B want to start?}'\text{,}'\textit{SP=Malvern}') \\
\textbf{acknowledge}(\text{B,DU4}) \\
\text{QU4:}\textbf{info\_request}(\text{A,B,}'\textit{Where does B want to start?}') \\
\textbf{acknowledge}(\text{A,DU3}) \\
\textbf{answer}(\text{B,A,I2,}'\textit{A can help B by providing a route}') \\
\text{DI3:}\textbf{direct}(\text{B,A,}'\textit{Give B a route}') \\
\textbf{acknowledge}(\text{B,DU2}) \\
\text{I2:}\textbf{info\_request}(\text{A,B,}'\textit{How can A help B?}') \\
\textbf{greet}(\text{A,B})
\end{array}\right\rangle \\
\text{SCP} & = & \left\{\begin{array}{l}
\textbf{agree}(\text{B,CH6}) \rightarrow \textbf{scp}(\text{A,}'\textit{SP= Great Malvern}') \\
\textbf{scp}(\text{B,}'\textit{SP=Malvern}') \\
\textbf{scp}(\text{B,}'\textit{A can help B by providing a route}')
\end{array}\right\} \\
\text{OPT} & = & \left\{\ \textbf{address}(\text{A,DI3:}\textbf{direct}(\text{B,A,}'\textit{Give B a route}'))\ \right\}
\end{array}\right] \\
\text{INT} & = & \left[\begin{array}{l}
\textbf{agree}(\text{B,CH6}) \\
\textbf{understandingAct}(\text{B,DU6}) \\
'\textit{Get a route from malvern to edwinstowe}'
\end{array}\right] \\
\text{UDUS} & = & \left[\ \text{DU7}\ \right] \\
\text{DU2} & = & \left[\begin{array}{lll}
\text{OBL} & = & \left[\ \textbf{answer}(\text{B,I2:}\textbf{info\_request}(\text{A,B,}'\textit{How can A help B?}'))\ \right] \\
\text{DH} & = & \left\langle\begin{array}{l}
\text{I2:}\textbf{info\_request}(\text{A,B,}'\textit{How can A help B?}') \\
\textbf{greet}(\text{A,B})
\end{array}\right\rangle
\end{array}\right] \\
\text{DU3} & = & \left[\begin{array}{lll}
\text{OBL} & = & \left[\begin{array}{l}
\textbf{accept}(\text{A,DI3}) \rightarrow \textbf{obliged}(\text{A,B,}'\textit{Give B a route}') \\
\textbf{address}(\text{A,DI3:}\textbf{direct}(\text{B,A,}'\textit{Give B a route}'))
\end{array}\right] \\
\text{DH} & = & \left\langle\begin{array}{l}
\textbf{answer}(\text{B,A,I2,}'\textit{A can help B by providing a route}') \\
\text{DI3:}\textbf{direct}(\text{B,A,}'\textit{Give B a route}')
\end{array}\right\rangle \\
\text{SCP} & = & \left\{\ \textbf{scp}(\text{B,}'\textit{A can help B by providing a route}')\ \right\} \\
\text{OPT} & = & \left\{\ \textbf{address}(\text{A,DI3:}\textbf{direct}(\text{B,A,}'\textit{Give B a route}'))\ \right\}
\end{array}\right] \\
\text{DU4} & = & \left[\begin{array}{lll}
\text{OBL} & = & \left[\ \textbf{answer}(\text{B,QU4})\ \right] \\
\text{DH} & = & \left\langle\ \text{QU4:}\textbf{info\_request}(\text{A,B,}'\textit{Where does B want to start?}')\ \right\rangle
\end{array}\right] \\
\text{DU5} & = & \left[\begin{array}{lll}
\text{DH} & = & \left\langle\ \textbf{answer}(\text{B,A,}'\textit{Where does B want to start?}'\text{,}'\textit{SP=Malvern}')\ \right\rangle \\
\text{SCP} & = & \left\{\ \textbf{scp}(\text{B,}'\textit{SP=Malvern}')\ \right\}
\end{array}\right] \\
\text{DU6} & = & \left[\begin{array}{lll}
\text{OBL} & = & \left[\ \textbf{answer}(\text{B,CH6})\ \right] \\
\text{DH} & = & \left\langle\ \text{CH6:}\textbf{check}(\text{A,B,}'\textit{SP= Great Malvern}')\ \right\rangle \\
\text{SCP} & = & \left\{\ \textbf{agree}(\text{B,CH6}) \rightarrow \textbf{scp}(\text{A,}'\textit{SP= Great Malvern}')\ \right\}
\end{array}\right] \\
\text{DU7} & = & \left[\begin{array}{lll}
\text{DH} & = & \left\langle\begin{array}{l}
\textbf{agree}(\text{B,A,CH6}) \\
\textbf{agree}(\text{B,A,CH6})
\end{array}\right\rangle \\
\text{SCP} & = & \left\{\ \textbf{scp}(\text{B,}'\textit{SP= Great Malvern}')\ \right\}
\end{array}\right]
\end{array}\right]
\end{array}\right]
\end{array}\right]
$$