From Manual Text to Instructional Dialogue: an Information State Approach

Staffan Larsson

Department of linguistics, Göteborg University
Box 200-295, Humanisten, SE-405 30 Göteborg, Sweden
sl@ling.gu.se

Abstract

We present preliminary research on the relation between written manuals and instructional dialogue, and outline how a manual can be converted into a format which can be used as domain knowledge by a dialogue system, capable of generating both instructional dialogue and monologue. Starting from a short sample text from a manual, we use the TRINDI information state approach (Traum et al., 1999) to build an experimental dialogue system capable of instructing a user to perform the task. IMDiS, a small experimental implementation based on the GoDiS dialogue system (Bohlin et al., 1999), is presented.

1. Goal of the paper

In this paper, we will present preliminary research on the relation between written manuals and instructional dialogue. We outline how a manual can be converted into a format which can be used as domain knowledge by a dialogue system, capable of generating both instructional dialogue and monologue. Starting from a short sample text from a manual, we use the TRINDI information state approach (Traum et al., 1999) to build an experimental dialogue system capable of instructing a user to perform the task. IMDiS, a small experimental implementation based on the GoDiS dialogue system (Bohlin et al., 1999), is presented. We look at sample monologue and dialogue output and discuss the advantages provided by the dialogue mode in IMDiS. One of the main advantages is that the user can control the dialogue to make the system provide exactly the information needed. Finally, we discuss possible research issues.

We will make two basic assumptions: monologue is a special case of dialogue, and discourse structure corresponds to task structure. These assumptions are by no means original (see e.g. (Grosz and Sidner, 1986)); however, the preliminary work here attempts to combine these assumptions using the TRINDI information state approach to investigate the possibility of generating dialogue (as in built-in automatic assistant) or monologue (as in a traditional written manual) from a single database of domain task plans.

2. IMDiS

IMDiS (Instructional Monologue and Dialogue System) is an adaption of GoDiS to instructional dialogue, and like GoDiS it provides a simple but efficient grounding strategy and facilitates question and task accommodation (Bohlin et al., 1999). In addition, IMDiS can give instructions and the user can request more specific instructions by asking the system how to perform a given instruction. IMDiS can also be made to generate the original text by setting it in "monologue" mode that uses a slightly altered set of dialogue moves and information state update rules, but which still uses the same database and generation facilities as the dialogue mode.

IMDiS is implemented using the TRINDIKIT (Larsson et al., 1999), a toolkit for experimenting with information states and dialogue move engines and for building dialogue systems. We use the term *information state* to mean, roughly, the information stored internally by an agent, in this case a dialogue system. A *dialogue move engine* updates the information state on the basis of observed dialogue moves and selects appropriate moves to be performed. In this paper we use a formal representation of dialogue information states that has been developed in the TRINDI[1], SDS[2] and INDI[3] projects.

IMDiS has a type of information state similar to that of GoDiS, with the addition of a subfield SHARED.ACTIONS whose value is a stack of actions which the system has instructed the user to perform, but whose performance has not yet been confirmed by the user. The IMDiS information state is shown in Figure 1.

$$\begin{bmatrix} \text{PRIVATE} : \begin{bmatrix} \text{PLAN} : \text{StackSet(Action)} \\ \text{AGENDA} : \text{Stack(Action)} \\ \text{TMP} : \text{(same as SHARED)} \end{bmatrix} \\ \text{SHARED} : \begin{bmatrix} \text{BEL} : \text{Set(Prop)} \\ \text{QUD} : \text{StackSet(Question)} \\ \text{ACTIONS} : \text{Stack(Action)} \\ \text{LU} : \text{Utterance} \end{bmatrix} \end{bmatrix}$$

Figure 1: IMDiS information state type

The main division in the information state is between information which is private to the agent and that which is shared between the dialogue participants. The private part of the information state contains a PLAN field holding a dialogue plan, i.e. is a list of dialogue actions that the agent wishes to carry out. The plan can be changed during the course of the conversation. The AGENDA field, on the other hand, contains the short term goals or obligations that the agent has, i.e. what the agent is going to do next. We have included a field TMP that mirrors the shared fields.

[1] TRINDI (Task Oriented Instructional Dialogue), EC Project LE4-8314, www.ling.gu.se/research/projects/trindi/

[2] SDS (Swedish Dialogue Systems), NUTEK/HSFR Language Technology Project F1472/1997, http://www.ida.liu.se/ nlplab/sds/

[3] INDI (Information Exchange in Dialogue), Riksbankens Jubileumsfond 1997-0134.

This field keeps track of shared information that has not yet been grounded, i.e. confirmed as having been understood by the other dialogue participant. The SHARED field is divided into four subfields. One subfield is a set of propositions which the agent assumes for the sake of the conversation. The second subfield is for a stack of questions under discussion (QUD). These are questions that have been raised and are currently under discussion in the dialogue. The ACTIONS field is a stack of (domain) actions which the user has been instructed to perform but has not yet performed.The LU field contains information about the latest utterance.

The dialogue version uses 9 move types, basically the 6 used in GoDiS (**Ask**, **Answer**, **Inform**, **Repeat**, **ReqRep**, **Greet**, **Quit**) plus instructions to check preconditions (**InstructCheck**), plain instructions (**InstructExec**), and confirmations (**Confirm**). Confirmations are integrated by assuming that the current topmost action in SHARED.ACTIONS has been performed, as seen in the update rule below.

RULE: **integrateUsrConfirm**
CLASS: **integrate**
PRE: { val#rec(shared.lu.speaker, usr)
assoc#rec(shared.lu.moves, confirm, false)
fst#rec(shared.actions, A) }
EFF: { set_assoc#rec(shared.lu.moves, confirm, true)
pop#rec(shared.actions)
add#rec(shared.bel, done(A)) }

Elliptical "how"-questions from the user are interpreted as applying to the currently topmost action in the SHARED.ACTIONS stack.

The monologue mode uses only 3 moves (**InstructExec**, **InstructCheck** and **Inform**). Since there is no user to confirm that actions have been performed, all actions are automatically confirmed using the update rule **autoConfirm**.

RULE: **autoConfirm**
CLASS: **integrate**
PRE: { fst#rec(shared.actions, A) }
EFF: { pop#rec(shared.actions)
add#rec(shared.bel, done(A)) }

3. Manuals and dialogues

The text below is taken from a user manual for the Homecentre, a low end Xerox MultiFunctional Device.

Reinstalling the print head
Caution: Make sure that the green carriage lock lever is STILL moved all the way forward before you reinstall the print head.
1. Line up the hole in the print head with the green post on the printer carriage.
Lower the print head down gently into position.
2. Gently push the green cartridge lock lever up until it snaps into place.
This secures the print head.
3. Close the top cover and reattach the scanner.
4. Press and release the yellow LED button.
The printer will prepare the cartridge for printing.
Note: If the carriage does not move from the center position after you press the cartridge change button, remove and reinstall the print head.

From this text, one can (re)construct a domain plan for reinstalling the print head. Such a plan may be represented as in Figure 2. Note that this is a conditional plan, i.e. it contains branching conditions.

From this plan, IMDiS generates two plans: a monologue plan and a dialogue plan. This is done using the "translation schema" in Figure 3.

The difference between the text plan and the dialogue plan is in the way that conditionals in the domain plan are interpreted. In the monologue plan, they correspond to simply informing the user of the conditional. In dialogue mode, however, the system raises the question whether the condition holds. When the system finds out if the condition holds, it will instruct the user to execute the appropriate guarded action.

In short, here's how conditionals are treated by the system in dialogue mode: When the system has found out what the user's task is, it will load the appropriate dialogue plan into the PRIVATE.PLAN field of the information state. It will then execute the actions in the appropriate order by moving them to the agenda and generating appropriate utterances. When a conditional statement is topmost on the plan, IMDiS will check whether it has been established that the condition holds (by checking the SHARED.BEL field). Since the system has previously asked the user and the user has answered, either the condition or its negation will be in the set of established propositions. If the condition or its negation holds, the conditional will be popped off the plan and replaced by the first or second guarded action (respectively).

4. Monologue and dialogue

In the monologue mode in IMDiS, the control module does not call the input and interpretation modules. The text is output "move by move" as a sequence of utterances from the system[4].

```
S: Reinstalling the print head.
S: Make sure that the green carriage lock
lever is STILL moved all the way forward
before you install the print head.
S: Line up the hole in the print head with
the green post on the printer carriage
```

Compared to the monologue mode, the dialogue mode offers several advantages:

User attention and control The user can direct her attention to the machine and does not have to look at the manual. This means that the user does not have to keep track of the

[4]While perhaps not practically useful, the implementation of a monologue mode in IMDiS is primarilly intended to show how one can construe the claim that monologue is a special case of dialogue.

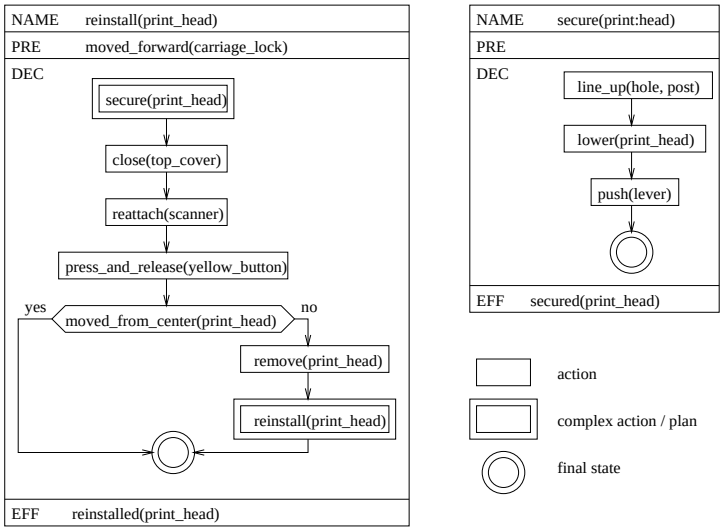

Figure 2: Domain plan

DOMAIN	MONOLOGUE	DIALOGUE
precondition P	**InstructCheck(**P**)**	**InstructCheck(**P**)**
action A	**InstructExec(**A**)**	**InstructExec(**A**)**
`if_then_else(`C`,`A`,`A'`)`	**Inform**(`if_then_else(`C`,`A`,`A'`)`)	`findout(`C`);` `if_then_else(`C`,` **InstructExec(**A**)**, **InstructExec(**A'**))**
effect E	**Inform(**E**)**	**Inform(**E**)**

Figure 3: Plan conversion table

current point in the manual. Also, the user decides when to move on to the next action, by confirming that the previous action is done.

Avoid irrelevant information When the action to be taken depends on a condition, the system does not give irrelevant information.

```
S: Has the carriage moved from the center
position?
U: yes
S: The print head is now installed
```

Control over level of detail If the user does not know how to perform a substep, she can ask the system for more detailed instructions.

```
S: secure the print head
U: how?
S: Line up the hole in the print head with
the green post on the printer carriage
U: right, ok
S: Lower the print head down gently into
position
U: ok
S: Gently push the green cartridge lock
lever up until it snaps into place
U: ok
S: The print head is now secured
```

On the other hand, if the user already knows how to perform a substep, the system moves on to the next step.

```
S: secure the print head
U: ok, and now?
S: Close the top cover
```

If the user manages to complete the whole action sequence without instructions, she can tell the system this and the system proceeds to the next relevant action.

```
S: secure the print head
U: how?
S: Line up the hole in the print head with
the green post on the printer carriage
U: right, ok
S: Lower the print head down gently into
position
U: ok right, I remember now (pause) the
print head is secured
S: Ok.  Close the top cover
```

Question accommodation The user can provide answers to questions which the system has not yet asked, provided the system is able to accommodate the question. For

```
$S> Press and release the yellow LED button

:  private =   agenda =   < instruct_exec(press_and_release(yellow_LED_button)) >
:              plan =     [ raise(move_from_center_position(carriage_head)),
                            if_then(not move_from_center_position(carriage_head),
                                    instruct_exec(remove_and_reinstall(print_head))),
                            inform(reinstalled(print_head)),
                            inform(next(prepare_cartridge_for_printing)) ]
:              tmp = (*surpressed*)
:  shared =    bel =      { done(reattach(scanner)),
                            done(close(top_cover)),
                            done(secure(print_head)),
                            done(check(moved_forward(carriage_lock))),
                            task(instruct_exec(reinstall(print_head))) }
:              qud =      < >
:              actions =  < press_and_release(yellow_LED_button) >
:              lu =        (*surpressed*)
```

Figure 4: Sample IMDiS information state, after uttering "Press and release the yellow LED button"

example, the user does not have to wait for the system to ask what task the user wants to perform.

```
S: Hello and welcome to the IMDiS homecen-
tre assistant
U: i want to reinstall the print head
S: Make sure that the green carriage lock
lever is still moved all the way forward
before you install the print head.
```

Grounding If the users does not hear or understand a system utterance, she can ask the system to repeat it.

```
S: Has the carriage moved from the center
position?
U: what ?
S: Has the carriage moved from the center
position?
```

5. Research issues

In building the experimental IMDiS, we have made several simplifications. For example, the problem of NL generation has been side-stepped by using canned text for output. Around 90% of the lexicon is used in both dialogue and monologue mode, while the rest is specific to one mode. It is a research issue to what extent canned text can be used, and how much "real" generation is necessary. Although this is experimental work, it does not seem implausible that useful systems could be constructed fairly easily to the extent that system output can be provided as canned text and that user input is limited in its lexical scope. On a domain level, what needs to be done is to construct domain plans and connect them to the corresponding text output. We make no claims here that this process is easily automated; rather, the idea is that instead of writing a manual (which will, in a sense, encapsulate both domain knowledge and its linguistic realisation), the author constructs the plans and output manually (possibly using a specialised authoring tool).

Also, IMDiS is not capable of referent disambiguation dialogue of the kind common in e.g. the MapTask corpus (Anderson et al., 1991). This type of dialogue would be needed for the system to be able to explain e.g. which component is being referred to and where it is to be found.

So far, we have only explored the extremes of the monologue-dialogue opposition. There are interesting intermediate levels of interactivity, such as dynamically generated text where the content depends on what has previously been related to the user. This is another area of possible future research, where it is likely that higher demands will be put on dynamic language generation.

Although this is not strictly relevant to the monologue-dialogue discussion, we would also like to compare IMDiS to previous instructional dialogue systems such as that described in (Smith and Hipp, 1994).

6. References

A. H. Anderson, M. Bader, E.G. Bard, E. Boyle, G. Doherty, S. Garrod, S. Isard, J. Kowtko, J. McAllister, J. Miller, C. Sotillo, H. Thompson, and R. Weinert. 1991. The HCRC Map Task corpus. *Language and Speech*, 34(4):351–366.

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

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

S. Larsson, P. Bohlin, J. Bos, and D. Traum. 1999. Coding instructional dialogue for information states. deliverable D2.2, TRINDI.

R. W. Smith and D. R. Hipp. 1994. *Spoken Natural Language Dialog Systems*. Oxford University Press.

D. Traum, J. Bos, R. Cooper, S. Larsson, I. Lewin, C. Matheson, and M. Poesio. 1999. Coding instructional dialogue for information states. deliverable D2.1, TRINDI.