

Dialogue Moves for Natural Command Languages

J. Gabriel Amores, José F. Quesada
Grupo de Investigación JULIETTA
Universidad de Sevilla

Abstract This paper studies Natural Command Language dialogues from the Information State Update perspective and proposes the specification of Dialogue Moves for this type of dialogues. First, we describe the properties of Natural Command Languages and Natural Command Language Dialogues. The paper proposes a classification of Dialogue Moves for Natural Command Language Dialogues and applies these moves to a sample dialogue. Finally, we compare Natural Command Language Moves with other Move Coding Schemes.¹

1 Introduction

One of the main goals of the Siridus project is to extend the range of types of dialogue to which the Information State Update (ISU) view of dialogue is applicable. Roughly speaking, the term *information state* means the information stored internally by an agent, in this case a dialogue system. More precisely, the information state of a dialogue includes all the cumulative additions from previous actions in the dialogue, which, taking into account the structure and goals of the dialogue, motivates and determines the future actions in the dialogue. By its own nature and functionality, the information state of a dialogue is a dynamic structure, that evolves (is updated) from the observation and performance of *dialogue moves*. This paper concentrates on the study of Natural Command Language dialogues from the Information State Update perspective, and on the specification of Dialogue Moves for this type of dialogues.

¹This work has been funded by EU Fifth Framework Project SIRDUS (IST-1999-10516)

2 Natural Command Languages

Briefly speaking, a *Natural Command Language* is a command language expressed through the medium of natural language. A Natural Command Language (NCL) should satisfy the following properties:

1. Use natural language vocabulary as far as possible.
2. Reflect natural subcategorization frames (e.g. optional elements to be expressed via optional PPs, required items to be subcategorized for), especially as far as semantic selectional restrictions is concerned. In this sense, an argument semantic type might determine the precise meaning of the function.
3. They reflect natural syntactic order (e.g. in English Verb-Obj-Obj for imperatives —not all commands need be imperative though). However, from a structural point of view, NCLs tend to reflect patterns of spoken language, rather than written language. They may, for example, make use of topicalised patterns permit some use of context dependent expressions such as pronouns, ellipsis, etc.
4. Constitute a sub-language, with reasonably clear boundaries. It is important that users can learn the degree of expressivity of the language, use elements of it compositionally and yet not expect any arbitrary NL expression to be machine understandable.
5. New functionality in the system ought to be able to be accommodated within the language without altering language structural properties. Users ought to be able to introduce to some degree their own modes of expression and have the system understand them.

6. Be domain-independent. Ideally, issuing commands in a new domain should not necessarily mean learning a completely new language. Lexical items might vary (and extra ambiguity might be introduced) but again structural principles ought not to vary. Taking this idea to an extreme would basically reduce the NCL grammar to just one production:

Function -> FunctionWord
Parameters*.

However, this desideratum may conflict somewhat with the previous goal (4) “Naturally occurring sublanguages” may very well have domain dependent restrictions on syntax and semantics (e.g. in domain X always interpret K as P), even deviant rules, and may lead to inefficiency from a computational point of view.

7. Cover system outputs, as well as inputs. If system outputs can be expressed in the sublanguage, then this may encourage the user to use it and help him to learn it. Feedback messages will need to be sensitive to problems in the linguistic expression of commands; for example, potential ambiguities.

In conclusion, we take NCLs as the set of input and output natural language expressions which are acceptable in a given application domain. This domain is semantically defined by the functions (commands) known by the user and the system, and the natural language vocabulary which may be used to express those commands. In addition, it should contain metalinguistic patterns and expressions typical of human-like interaction.

3 Natural Command Language Dialogues

Natural Command Language Dialogues (NCLD) are artificially constructed models (including knowledge representation and reasoning) able to guide the interaction between the different parts involved in a dialogue based on a NCL.

A NCLD will at least allow the following kind of phenomena:

1. Multiple Commands NCL

NCLD must be able to manage different commands and dialogue operations. So, one of the main functions of NCLD models will be Command Detection.

2. Context Dependency

Only at the dialogue level is it possible to understand anaphora, ellipsis and other context dependent constructions. From the dialogue system design perspective, the treatment of these discourse phenomena will imply the representation and storage of the whole dialogue history so far.

3. Man-Machine Interaction

Naturalness and flexibility of the flow of interactions (restricted to the linguistic limitations of the underlying NCL), relevance and adequacy of the outputs, consistency (order of arguments in commands), etc.

4. Interface with External Functional Components

NCL are aimed at the specification of commands belonging to a command language. The user's goal is to execute the command(s). Therefore, the dialogue level must not only understand the naturally expressed command, but also execute it. In fact, this implies the definition and use of another Command Language between the NCLD System and the external functional components in charge of the execution of the commands.

4 Dialogue Moves in NCL Dialogues

This section focuses on Natural Command Languages from the Information State Update approach, paying special attention to the specification and classification of Dialogue Moves for this type of dialogues.

Following the classification proposed in the Information State Theory of Dialogue Modelling [Traum *et al*, 1999], the following subsection shows a description of the informational components of the information state.

5 Informational Components

- **Modeling Internal vs External aspects of the Dialogue:** As a consequence of their own nature, Natural Command Language Dialogues involve

just two participants: the user and the machine. The first decision concerns whether we should model the participants' internal state, or more external aspects of the dialogue.

Since the goal of this type of dialogues is that the user have control over the execution of one or more commands by the machine, most dialogues exhibit a marked functional or operational tendency. In addition, the sequence of commands may sometimes lack some logical flow, or even be the consequence of obscure intentions during the dialogue.

For all these reasons, it seems reasonable to focus our model on the external aspects of dialogue. That is, it should be based more on what was said than what was in the minds of the participants when things were said.

- **Static and Dynamic Aspects:** One of the aspects which differentiates Natural Command Language Dialogues from Information Seeking Dialogues is the dynamic nature of the knowledge bases involved.

In an Information Seeking Dialogue, in which the system as a whole is viewed by the user as a repository of knowledge, knowledge bases have a clear static character from the point of view of the user. That is, the data stored in these resources may be updated, but not by the user during its interaction with the system.

On the contrary, one of the main features of Natural Command Language Dialogues is the presence of a command execution system which is capable of dynamically modifying the contents of external resources to the dialogue (during its interaction with the user), such as the knowledge bases associated to the domain.

5.1 Dialogue Moves

Following is a list of the dialogue moves which we have identified after the study of several scenarios belonging to Natural Command Language Dialogues, namely the *ATOS (Automatic Telephone Operator System)* dialogue corpus, some examples from the description of the *Natural Language Interface to Virtual Reality Systems* project,

and some other examples obtained from artificially modelled possible natural language interactions with a database manager system.

The table below summarizes the Dialogue Moves for Natural Command Languages classifying them into three groups:

	<i>Dialogue Moves in NCL</i>
Command-oriented	askCommand specifyCommand informExecution
Parameter-oriented	askParameter specifyParameter
Interaction-oriented	askConfirmation answerYN askContinuation askRepeat askHelp answerHelp errorRecovery greet quit

5.2 Command-Oriented Dialogue Moves

These dialogue moves deal with command specification tasks such as askCommand, specifyCommand and informExecution.

5.2.1 askCommand

This move is related to those situations in which the system requests the user to specify a command or function to be performed.

For instance:

- *Specify a function, please.*
- *do you want a route?*

There are two possible notations for this dialogue move, depending on whether the move suggests or not a possible command:

- **askCommand** *Specify a function, please.*
- **askCommand(travelRoute)** *do you want a route?*

5.2.2 specifyCommand

This move takes place when a specific command or function has been selected. Often, a command specification may include one or more of its parameters.

For instance:

- *I want to place a call.*
- *flights to paris.*
- *I want to call 1 2 3 4 collect.*

The notation should include the command, and the set of instantiated parameters:

- **specifyCommand(phoneCall)** *I want to place a call.*
- **specifyCommand(fly,cityTo=paris)** *flights to paris.*
- **specifyCommand(collectCall, destination = 1234)** *I want to call 1 2 3 4 collect.*

5.2.3 informExecution

This corresponds to the actual execution of the command, which, of course, may change the state of the world and the information state of the participants. Once this dialogue move has finished, the participants know that the command has been executed. At a linguistic level, this move will be associated with utterances where the system acknowledges the execution of the task. In some situations, to improve naturalness, it may be an implicit move. In other cases, this move corresponds to the actual execution (as with a phone call).

- Example 1:
 - *Do you really want to call 1 2 3 4 collect?*
 - *yes, please.*
 - *CALL COLLECT EXECUTION*
- Example 2:
 - *Do you want the directory list ordered by name?*
 - *Yes, please.*
 - *DIRECTORY LIST.*

The notation should include the command plus all relevant parameters:

- **informExecution(collectCall, destination = 2134)** *CALL COLLECT EXECUTION*
- **informExecution(ls, directory = test, order = name)** *DIRECTORY LIST.*

5.3 Parameter–Oriented Dialogue Moves

As commands usually require specific parameters, several dialogue moves control command completeness (mainly, instantiation of mandatory parameters). Two dialogue moves (**askParameter** and **specifyParameter**) have been identified in this group.

5.3.1 askParameter

In the **askParameter** move, the system asks for the value of a specific parameter.

Examples:

- *What city do you want to go from?*
- *Do you want a return ticket?*
- *when do you go?*
- *Yes, who would you like to call?*

The notation should include the parameter (the command may be inferred from the dialogue context), and possibly the suggested value for the parameter if it was present (as in the last example):

- **askParameter(cityFrom)** *What city do you want to go from?*
- **askParameter(returnTicket)** *Do you want a return ticket?*
- **askParameter(when)** *when do you go?*
- **askParameter(destination)** *Yes, who would you like to call?*

5.3.2 specifyParameter

This move corresponds to the assignment of some value to one parameter.

For instance:

- *london.*
- *yes, as cheap as possible.*
- *I will start at nine am.*

In this case, the notation must include the parameter and its value, which, in some cases, may have a very complex structure.

- **specifyParameter(cityFrom = london)** *london.*
- **specifyParameter(returnTicket = yes yes, specifyParameter(price = cheap)** *as cheap as possible.*
- **specifyParameter(when = 9am)** *I will start at nine am.*

5.4 Interaction-oriented Dialogue Moves

Finally, a third group includes those dialogue moves related to some intrinsic characteristics of the dialogue such as the confirmation of the command before its execution, sub-help dialogues, error recovery strategies (of special interest with spoken language systems), etc.

5.4.1 askConfirmation

Once a command has been completed, some situations will require an explicit and/or implicit confirmation. In some applications, a confirmation may be required for each command and parameter, in order to acknowledge each piece of information.

Example:

- *Do you really want to call 1 2 3 4 collect?*

It is important to specify the explicit part of the command that is being confirmed:

- **askConfirmation**(collectCall, destination = 1234) *Do you really want to call 1 2 3 4 collect?*

5.4.2 answerYN

This move represents the classical yes/no reply.

- *Yes.*
- *That's wrong.*

The current logical value will be stored as part of the dialogue move:

- **answerYN**(yes) *Yes.*
- **answerYN**(no) *That's wrong.*

5.4.3 askContinuation

Given the cyclical nature of command execution in this type of dialogues, each time a command has been completed (a sort of conversational game) there will follow a move asking for the continuation of the dialogue. This move could explicitly suggest a next command given the state of affairs and the last executed commands. Thus, an **answerYN** affirmative reply would actually behave as a **specifyCommand** move.

Some examples are:

- *Do you want to continue?*

Whose representation would be:

- **askContinuation** *Do you want to continue?*

5.4.4 askRepeat

Any of the participants may request the other to repeat the last utterance, or even a specific parameter or command.

- Repeat example:

- *Excuse me.*
- *I want to call 1 2 3 4 collect.*
- *Can you repeat the destination, please?*
- *1 2 3 4.*

The askRepeat sentences in the previous sub-dialogue will be specified as follows:

- Repeat example:

- **askRepeat** *Excuse me.*
- *I want to call 1 2 3 4 collect.*
- **askRepeat**(destination) *Can you repeat the destination, please?.*
- *1 2 3 4.*

The askRepeat sentences in the previous sub-dialogue will be specified as follows:

- Repeat example:

- **askRepeat** *Excuse me.*
- *I want to call 1 2 3 4 collect.*
- **askRepeat**(destination) *Can you repeat the destination, please?.*
- *1 2 3 4.*

5.4.5 askHelp

This move shows up with a petition for help, which may be very general (i.e. functionality of the whole system), a specific command, or a specific parameter (i.e. the range of possible values), etc.

The following sub-dialogue illustrates this situation:

- Help sub-dialogue example:

- *What are the available functions of the system?*
- *You can place a phone call, consult your directory or send a message.*

- *Well, how can I send a message?*
- *To send a message, first you have to indicate the addressee of the message*
- *That's better, but what types of addressees may I use?*
- *The destination of a message may me a telephone number or a person in your directory.*

The annotation we suggest for these petitions are:

- Help sub-dialogue example (with **askHelp** notations):
 - **askHelp** *What are the available functions of the system?*
 - *You can place a phone call, consult your directory or send a message.*
 - **askHelp(sendMessage)** *Well, how can I send a message?*
 - *To send a message, first you have to indicate the addressee of the message*
 - **askHelp(destination)** *That's better, but what types of addressees may I use?*
 - *The destination of a message may me a telephone number or a person in your directory.*

5.4.6 answerHelp

This dialogue move corresponds to the reply to an **askHelp** move. Therefore, there are different modes depending on the type of help requested.

Next, we show the previous dialogue sample using this new dialogue move:

- Help sub-dialogue example (with **askHelp** and **answerHelp** notations):
 - **askHelp** *What are the available functions of the system?*
 - **answerHelp** *You can place a phone call, consult your directory or send a message.*
 - **askHelp(sendMessage)** *Well, how can I send a message?*
 - **answerHelp(sendMessage)** *To send a message, first you have to indicate the addressee of the message*

- **askHelp(destination)** *That's better, but what types of addressees may I use?*
- **answerHelp(destination)** *The destination of a message may me a telephone number or a person in your directory.*

5.4.7 errorRecovery

During the dialogue interaction, due to inconsistencies in the information provided, in what was understood by the participants or in what was recognized by a speech recognition system, there may arise situations in which the continuation of the dialogue is impossible.

In these situations, an **errorRecovery** dialogue move generated by a participant warns the other participant about the inconsistency detected, and suggests a restart from the last consistent state, or he/she may show some reservations about the item which may be causing the error or inconsistency.

The following example illustrates this situation:

- *I'm sorry. I don't understand which function you would like to perform. Let's start from the beginning. Could you tell me which function you would like to perform?*

In this case, the recovery move has clearly identified the source of the error (the command was not recognized), and a repetition of the command is requested.

- **errorRecovery(command)** *I'm sorry. I don't understand which function you would like to perform. Let's start from the beginning. askCommand Could you tell me which function you would like to perform?*

5.4.8 greet

This move describes the usual greeting operation:

- **greet** *Welcome to the system!*

5.4.9 quit

Likewise, this move represents the usual closing operation:

- **quit** *thanks, bye!*
- **quit** *Thank you for your visit!*

6 *NCL Dialogue Moves for the Spanish Telephone Command Dialogue Corpus*

This section applies the model of dialogue moves proposed in the previous section to a sample of a dialogue belonging to the Spanish Telephone Command Dialogue (STCD) corpus.

The main features of this scenario have been discussed in document *D3.1 User Requirements on a Natural Command Language Dialogue System* [Torre, Amores & Quesada, 2000], which also includes a set of representative dialogues in this corpus as well as a detailed description of the expected functionality of the system and the supported commands.

7 *Annotating the Spanish Telephone Command Dialogue Corpus*

This section contains the hand annotation of the dialogue number 10 shown in chapter 6 of [Torre, Amores & Quesada, 2000], using the dialogue move scheme described above.

- *S(1)*: **greet** Welcome to the system.
askCommand Which function would you like to perform?
- *U(1)*: **specifyCommand(phoneCall)** I want to place a call
SILENCE **specifyCommand(phoneCall)** place a call.
- *S(2)*: **askParameter(destination)** Who would you like to call?
- *U(2)*: **specifyParameter(destination = 9898)** Extension nine eight nine eight.
- *S(3)*: **informExecution(phoneCall,destination = 9898)** I'll put you through to extension nine eight nine eight.
- *U(3)*:
- *S(4)*: **askContinuation** Would you like to perform any other function?
- *U(4)*: **specifyCommand(phoneCall)** I want to place a call.
- *S(5)*: **askParameter(destination)** Who would you like to place a call?
- *U(5)*: **answerYN(no)** No, **specifyCommand(collectcall)** I want to place a collect call.
- *S(6)*: **askParameter(destination)** Who would you like to call?
- *U(6)*: **answerYN(no)** No, **specifyCommand(sendMessage)** I want to send a message.
- *S(7)*: **askParameter(destination)** If you want to place a collect call, you must provide a telephone number or a name.
- *U(7)*: **specifyCommand(sendMessage)** I want to send a message.
- *S(8)*: **errorRecovery** I'm sorry. I don't understand which function you would like to perform. Let's start from the beginning. **askCommand** Could you tell me which function you would like to perform?
- *U(8)*: **specifyCommand(sendMessage)** I want to send a message.
- *S(9)*: **askParameter(destination)** Who would you like to send a message to?
- *U(9)*: **specifyParameter(destination = JuanMedinaRobles)** To Juan Medina Robles.
- *S(10)*: **answerHelp(sendMessage)** Please, record your message after the tone. Pi.
- *S(11)*: **askConfirmation(sendMessage,destination = JuanMedinaRobles)** Sending the message to Juan Medina Robles.
- *U(10)*: **answerYN(no)** No, **specifyParameter(destination = JuliaMedinaRobles)** Julia Medina Robles.
- *S(12)*: **askConfirmation(destination = JuliaMedinaRobles)** Julia Medina Robles?
- *U(11)*: **answerYN(yes)** Yes.
- *S(13)*: **informExecution(sendMessage,destination = JuliaMedinaRobles)** Sending the message to Julia Medina Robles.
- *U(12)*:
- *S(14)*: **askContinuation** Would you like to perform any other function?
- *U(13)*: **answerYN(no)** No.
- *S(15)*: **quit** Thank you for using the system.

8 Comparing Natural Command Language Moves with Other Move Coding Schemes

This section compares the taxonomy of NCL Moves with other move coding schemes. We will try to point out the similarities and differences between our coding scheme and that of other projects and scenarios such as TRAINS, DRI, MapTask, LINLIN2, GBG-IM both at the level of scheme layers and move taxonomies.

8.1 Scheme Layers

As regards scheme layers, we have classified our moves into three groups: Command-oriented, Parameter-oriented and Interaction-oriented.

Table 2.1 in [Cooper *et al*, 1999] *Rough Impression of relations between scheme layers* may be reused to some extent in order to accommodate our scheme layers as follows (Table 1).

8.2 Move Taxonomies

Similarly, Tables 2.2 and 2.3 in [Cooper *et al*, 1999] have guided us in comparing NCL moves with previous coding schemes, as shown in Tables 2 and 3 below.

9 Conclusion

This paper has proposed a taxonomy of Dialogue Moves for Natural Command Language Dialogues. The classification has been successfully applied to the Spanish Telephone Command Dialogue Corpus.

As part of the Siridus project (<http://www.cam.sri.com/siridus/>) we plan to apply this taxonomy as part of the implementation of a Dialogue System for the Automatic Telephone Operator Scenario, and to evaluate the taxonomy (its advantages and problems) in relation to the other referenced taxonomies.

References

- [Alexandersson *et al*, 1998] Alexandersson, J., B. Buschbeck-Wolf, T. Fujinami, M. Kipp, S. Koch, E. Maier, N. Reithinger, B. Schmitz & M. Siegel. (1998) *Dialogue Acts in VERBMOBIL-2*. DFKI Saarbrücken and TU Berlin, Report 226, July 1998.
- [Allen & Core, 1997] Allen, J. and Core, M. (1997). DAMSL: Dialog Act Markup in Several Layers. Draft contribution for the Discourse Resource Initiative.
- [Albesano *et al*, 1997] Albesano, D., Baggia, P., Gemello, R., Gerbino, E., and Rulenti, C. (1997) A robust system for human-machine dialogue in a telephony-based application. *Journal of Speech Technology*, 2(2), 99–110.
- [Allwood *et al*, 1994] Allwood, J., Nivre, J., and Ahlson, E. (1994) Semantics and Spoken Language: Manual for Coding Interaction Management. Report from the HSFR project *Semantik och talspråk*.
- [Cooper, 1998] Cooper, R. (1998). 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 & Larsson, 1999] Cooper, R., and Larsson, S. (1999). Dialogue moves and information states. In *Proceedings of the Third IWCS*, Tilburg.
- [Cooper *et al*, 1999] Cooper, R., Larsson, S., Matheson, C., Poesio, M., and Traum, D. (1999). Coding Instructional Dialogue for Information States. Deliverable D1.1, Trindi Project.
- [Kamp & Reyle, 1993] Kamp, H. & Reyle, U. (1993) *From Discourse to Logic*. Dordrecht: Kluwer.
- [Poesio *et al*, 1999] Poesio, M., R. Cooper, S. Larsson, D. Traum & C. Matheson. (1999) Annotating Conversations for Information State Updates. In *Proceedings of the Amsterdam '99 Workshop on the Semantics and Pragmatics of Dialogue*. Amsterdam University, May 1999.
- [Torre, Amores & Quesada, 2000] Torre, D., J. G. Amores and J. F. Quesada. (2000) *User Requirements on a Natural Command Language Dialogue System*. Deliverable 3.1. Siridus project.
- [Traum & Hinkelman, 1992] Traum, D., and Hinkelman, E. A. (1992). Conversation acts in task-oriented spoken dialogue. *Computational Intelligence*, 8(3). Special Issue on Non-literal Language.
- [Traum *et al*, 1999] Traum, D., Bos, J., Cooper, R., Larsson, S., Lewin, I., and Matheson, C. (1999). A model of dialogue moves and information state revision. Deliverable D2.1, Trindi Project.

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

Table 1: Rough impression of relations between scheme layers

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

Table 2: Rough impression of relations between move taxonomies, pt. 1

LINLIN2	MapTask	DRI	TRAINS	GBG-IM	NCL	
—	<i>(Response moves)</i>	<i>Understanding</i>	<i>Grounding</i>	<i>Feedback function</i>	—	
	—	—	ReqAck	Elicit FB		
	(Acknowledge)	—	ReqRepair	+Accept-com-act		
	—	—	—	-Accept-com-act		
	(Acknowledge)	<i>Signal-understanding</i> Acknowledge Repeat-rephrase Completion	Ack	+Understanding		
	—	Signal-Non-Understanding	—	-Understanding		askRepeat
	Acknowledge	—	—	+Perception		
	—	(Signal-non-und.)	—	-Perception		
	—	—	—	+Contact		
	—	(Signal-non-und.)	—	-Contact		
	—	Correct-Misspeaking	Repair	—		errorRecovery
—	—	—	<i>Turn-taking</i>	<i>Turn management</i>	—	
			take-turn	Turn acceptance		
			keep-turn	Turn holding		
			release-turn	Turn closing		
			assign-turn	—		
<i>Discourse management</i>	—	<i>Conventional</i>	—	—	—	
Opening		Opening				
Continuation		—				
Closing		Closing				
					greet	
					askContinuation	
					quit	

Table 3: Rough impression of relations between move taxonomies, pt. 2