A Dynamic Logic Formalisation of Inquiry-Oriented Dialogues*

Raquel Fernández
Department of Computer Science
King's College London
raquel@dcs.kcl.ac.uk

Abstract

In this paper we explore the possibility of using the paradigm of Dynamic Logic (DL) to formalise information states and update processes on information states. In particular, we present a formalisation of *Inquiry-Oriented Dialogues* in terms of Ginzburg's *dialogue gameboard*. From a more general point of view, we show that DL is specially well suited to develop rigorous formal foundations for an approach to dialogue dynamics based on information state updates.

1 Introduction

In communication modelling, one can identify two main traditions: on the one hand, classical Artificial Intelligence approaches, inspired by ideas originated in analytical philosophy, are built on general models of rational agency, emphasising the role of mental attitudes such as knowledge, belief, desire, intention and so on. In order to formalise these notions, these approaches (most typically the BDI[1] framework; see e.g. (Cohen and Levesque, 1990; Grosz and Sidner, 1990; Sadek, 1991)) strongly rely on detailed logical theories of rational action based on modal logics with a possible worlds semantics. On the other hand, one can distinguish a parallel line of research, following the work of philosophers like (Lewis, 1979) and (Stalnaker, 1979), which, instead of focusing on the intentional attitudes of the interacting agents, highlights the public and conventional aspects of communication. Under this perspective, a dialogue can be seen as a *conversational scoreboard* that keeps track of the state of the conversation.

A particular development within this latter tradition, which has received much attention in recent dialogue modelling, is the use of *information states* to characterise the state of each participant's information as the conversation proceeds. The information state approach to dialogue, as developed for instance in the TRINDI project (e.g. (Bohlin et al., 1999; Traum et al., 1999), assumes that some aspects of dialogue management are best captured in terms of the relevant information that is available to each dialogue participant at each state of the conversation, along with a full account of the possible update mechanisms that change this information. Thus, unlike classical, intentional accounts built on the basis of axiomatic theories of rational agency, information state approaches tend to avoid the use of logical frameworks and concentrate on dialogue-specific notions such as common ground, discourse obligations and questions under discussion.

In this paper we explore the possibility of using a modal logic paradigm, namely Dynamic Logic (Harel et al., 2000), familiar from BDI approaches, to formalise not motivational attitudes, but information states and update processes on information states. In particular, we present a dynamic logic formalisation of *Inquiry-Oriented Dialogues* (IODs), in the

*Published in *Proceedings of the 6th CLUK Colloquium*, pages 17-24, Edinburgh, UK, 2003.

[1] Beliefs, Desires and Intentions.

sense of (Larsson, 2002), built on Ginzburg's *dialogue gameboard* (DGB), as introduced in (Ginzburg, 1996; Ginzburg, ms). From a more general point of view, we show that Dynamic Logic is specially well suited to develop rigorous formal foundations for an approach to dialogue dynamics based on information state updates.

1.1 Overview

The structure of the paper is as follows: First, we introduce the basic notions of First-Order Dynamic Logic (DL), describing its syntax and semantics. After briefly characterising the structure of the DGB in Section 3, our formalisation is presented in Section 4. We define the formal language and its semantic interpretation, and discuss how the different components of the DGB have been modelled. In section 5, we formalise conversational interaction as DL programs and show how to specify the correctness of a program that intends to characterise the class of IODs. Finally, in section 6, we present our conclusions and indicate some directions for future research.

2 Dynamic Logic: Basic Notions

The formalisation we present in this paper is based on the first-order version of Dynamic Logic (DL) as it is introduced in (Harel et al., 2000) and (Goldblatt, 1992). In short, DL is a multi-modal logic with a possible worlds semantics, which distinguishes between expressions of two sorts: *formulae* and *programs*. The language of DL is that of first-order logic together with a set of modal operators: for each program α there is a box $[\,\alpha\,]$ and a diamond $<\alpha>$ operator. The set of possible worlds (or states) in the model is the set of all possible assignments to the variables in the language. Atomic programs change the values assigned to particular variables. They can be combined to form complex programs by means of a repertoire of program constructs, such as *sequence* **;** , *choice* $\cup$, *iteration* * and *test* ?.

Originally, DL was conceived as a formal system to reason about programs, formalising correctness specifications and proving rigorously that those specifications are met by a particular program. From a more general perspective, however, it can be viewed as a formal system to reason about transformations on states. In this sense, it is particularly well suited to provide a fine characterisation of the dynamic processes that take place in dialogue as updates on the information states of the dialogue participants.

In the remainder of this section, we formally introduce the syntax and the semantics of DL.

2.1 Syntax

The language of first-order DL is built upon First-Order Logic. It is generated by some first-order vocabulary Σ made up of a set of predicate symbols, a set of function symbols, a set of constants and a set of variables. In addition to the propositional connectives and the universal and existential quantifier symbols, the language also includes two modal operators [ ] and <>, a set Π of programs α and the program constructs **;**, $\cup$, * and ?.

Formulae and Programs. Atomic formulae φ are atomic, first-order formulae of the vocabulary Σ, including $\top$ and $\bot$. The set Φ of well-formed formulae A is then defined as follows:

$$A ::= \varphi \mid \neg A \mid A_1 \wedge A_2 \mid A_1 \vee A_2 \mid A_1 \rightarrow A_2 \mid \forall x A \mid \exists x A \mid [\,\alpha\,]\ A \mid <\alpha> A$$

In the basic version of DL, atomic programs π are simple assignments ($x := t$), where x is an individual variable and t is a first-order term. The set Π of programs α is defined as follows, where φ is any quantifier-free first-order formulae:

$$\alpha ::= \pi \mid \alpha_1; \alpha_2 \mid \alpha_1 \cup \alpha_2 \mid \alpha^* \mid \varphi?$$

2.2 Semantics

As usual in modal logic, the language is interpreted in a possible worlds semantical structure. A model is a structure $\mathcal{M} = \{\mathcal{A}, S, R, V\}$ where

- $\mathcal{A} = \{D, I\}$ is a first-order structure;
- S is a non-empty set of states;

$\mathcal{M} \models_s \varphi$	iff	$\mathcal{A} \models \varphi[v]$, for atomic formulae φ
$\mathcal{M} \models \top$		$\top$ is always true
$\mathcal{M} \not\models \bot$		$\bot$ is never true
$\mathcal{M} \models_s (t_1 = t_2)$	iff	$v_s(t_1)$ equals $v_s(t_2)$, for terms t_1 and t_2
$\mathcal{M} \models_s \neg A$	iff	$\mathcal{M} \not\models_s A$
$\mathcal{M} \models_s (A_1 \wedge A_2)$	iff	$\mathcal{M} \models_s A_1$ and $\mathcal{M} \models_s A_2$
$\mathcal{M} \models_s (A_1 \vee A_2)$	iff	$\mathcal{M} \models_s A_1$ or $\mathcal{M} \models_s A_2$
$\mathcal{M} \models_s (A_1 \rightarrow A_2)$	iff	$\mathcal{M} \not\models_s A_1$ or $\mathcal{M} \models_s A_2$
$\mathcal{M} \models_s \exists x A$	iff	there is an $a \in D$, s.t. $s(x\|a)s'$ and $\mathcal{M} \models_{s'} A$
$\mathcal{M} \models_s \forall x A$	iff	for all $a \in D$, if $s(x\|a)s'$ then $\mathcal{M} \models_{s'} A$
$\mathcal{M} \models_s <\alpha> A$	iff	there is an $s' \in S$, s.t. $sR_\alpha s'$ and $\mathcal{M} \models_{s'} A$
$\mathcal{M} \models_s [\ \alpha\ ] A$	iff	for all $s' \in S$, if $sR_\alpha s'$ then $\mathcal{M} \models_{s'} A$

Table 1: Definition of truth

• R is a function assigning to each program $\alpha \in \Pi$ a binary relation $R_\alpha \subseteq S \times S$;

• V is a function $V : S \rightarrow S^{\mathcal{A}}$ assigning to each $s \in S$ an $\mathcal{A}$-valuation $v_s : Var \rightarrow D$, i.e. a mapping from the set of variables to elements in the domain.

For $s, s' \in S$, we will write $s(x|a)s'$ to mean that $v_{s'}(x) = a$ and $v_{s'}(y) = v_s(y)$ whenever $y \neq x$.

Now we are ready to define the *truth-relation* $\mathcal{M} \models_s A$ of a formula A at state s in model $\mathcal{M}$. As usual in first-order logic, we write $\mathcal{A} \models \varphi[v]$ to mean that φ is *true in* $\mathcal{A}$ *under valuation* v. For conciseness, we will omit the part dealing with the semantics of first-order terms. The formal definition of truth in a model is shown in Table 1.

From the relations $R_\alpha \subseteq S \times S$, we can inductively define accessibility relations for the compound programs. Table 2 shows the accessibility relations for basic atomic programs and compound programs for all states $s, s' \in S$.

$sR_{x:=t}s'$	iff	$s(x\|v_s(t))s'$
$sR_{\alpha;\beta}s'$	iff	$\exists s''$ s.t. $sR_\alpha s''$ and $s''R_\beta s'$
$sR_{\alpha\cup\beta}s'$	iff	either $sR_\alpha s'$ or $sR_\beta s'$
$sR_{\alpha*}s'$	iff	there are finitely many states $s_1 \ldots s_n$ s.t. $sR_\alpha s_1 \ldots s_n R_\alpha s'$
$sR_{\varphi?}s'$	iff	$s = s'$ and $\mathcal{M} \models_s \varphi$

Table 2: Accessibility relations

Stack Variables. Interesting variants of DL arise from allowing auxiliary data structures such as *stacks* and *arrays*. Following (Harel et al., 2000), we will consider a version of DL in which programs can manipulate some variables as last-in-first-out stacks. Formally, stacks are modelled as variables ranging over finite strings of elements in the domain. To manipulate these *stack variables*, two additional atomic programs $\mathtt{X}.\mathbf{pop}$ and $\mathtt{X}.\mathbf{push}(x)$ are included. Here $\mathtt{X}$ is some stack variable (i.e. a string of elements) and x is an element in $\mathtt{X}$. The accessibility relations for these two new atomic programs are shown in Table 3, where, for a string σ and an element a, $\mathbf{tail}(a \cdot \sigma) = \sigma$ and $\mathbf{head}(a \cdot \sigma) = a$.

$sR_{\mathtt{X}.\mathbf{push}(x)}s'$	iff	$s(\mathtt{X} \mid v_s(x) \cdot v_s(\mathtt{X}))s'$
$sR_{\mathtt{X}.\mathbf{pop}}s'$	iff	$s(\mathtt{X} \mid \mathbf{tail}(v_s(\mathtt{X}))s'$

Table 3: **push** and **pop** programs

3 The Dialogue Gameboard

The *dialogue gameboard* (DGB) plays a central role in the theory of context developed by Ginzburg. It can be seen as the context relative to which conventionalised interaction is assumed to take place. The DGB provides a structured characterisation of the information which the dialogue participants view as common in terms of three main components: a set of FACTS, which the dialogue participants

take as common ground, a partial ordered set of questions under discussion QUD, and the LATEST-MOVE made in the dialogue. Inspired by the notion of *dialogue game* (e.g. (Hamblin, 1970; Carlson, 1983)), Ginzburg assumes that each move made by a dialogue participant determines a restricted set of options for follow-up in the dialogue, constraining what can be said and how.

The framework has been used to provide an account of the kind of context that licenses elliptical responses in dialogue (Ginzburg, 1999; Ginzburg et al., 2001; Fernández and Ginzburg, 2002) and has also been the starting point of implemented dialogue systems such as GoDiS (Cooper et al., 2001) and IBiS (Larsson, 2002).

4 A DL Formalisation of the DGB

To model context in dialogue as it is understood in Ginzburg's DGB, we will consider a particular domain of interpretation which includes entities such as agents (the dialogue participants), questions, propositions and dialogue moves. For the sake of simplicity, in this paper we restrict ourselves to four dialogue move types, namely *ask, assert, clarification request* and *acknowledge*. The main strategy to reason about the effects of conversational interaction on the DGB, will be to represent its main components as variables ranging over different domains. In what follows, we introduce the details of our formalism.

4.1 Introducing the Formalism

Let $\mathcal{L}$ be a first-order DL language with equality made up of unary predicates Q, P, G, DP, binary relations **infl**(uences) and **ans**(wers), a ternary relation Utt, a function symbol **whether**, constants a, b, `ask`, `ass`, `clr` and `ack`, and an infinite set Var of variables x. Var includes a set $V_1 = \{\mathtt{LM}_a, \mathtt{LM}_b, \mathtt{UTT}, turn, Goal\}$ of special individual variables and a set $V_2 = \{\mathtt{FACTS}, \mathtt{QUD}_a, \mathtt{QUD}_b, \mathtt{PENDING}_a, \mathtt{PENDING}_b\}$ of stack variables.

Language $\mathcal{L}$ is interpreted over a first-order structure $\mathcal{A} = \{D, I\}$. The domain D of $\mathcal{A}$ is made up of a set of dialogue participants $DP^{\mathcal{D}} = \{\mathbf{a}, \mathbf{b}\}$, a finite set of questions $Q^{\mathcal{D}}$, a finite set of propositions $P^{\mathcal{D}}$, a set of dialogue moves $M = \{ask, ass, clr, ack\}$, a singleton set $\{1\}$, a binary relation *infl* on $Q^{\mathcal{D}}$, a binary relation *ans* between $P^{\mathcal{D}}$ and $Q^{\mathcal{D}}$, a set of utterances $Utt^{\mathcal{D}}$, that will be modelled as triples (a, m, r) of a dialogue participant, a dialogue move and either a propositions or a question; and a function *whether* such that for every proposition p, $whether(p) \in Q^{\mathcal{D}}$. I is a function such that,

$$\begin{array}{rclrcl} I(Q) & = & Q^{\mathcal{D}} & I(G) & = & \{1\} \\ I(P) & = & P^{\mathcal{D}} & I(\mathtt{ask}) & = & ask \\ I(DP) & = & DP^{\mathcal{D}} & I(\mathtt{ass}) & = & ass \\ I(a) & = & \mathbf{a} & I(\mathtt{clr}) & = & clr \\ I(b) & = & \mathbf{b} & I(\mathtt{ack}) & = & ack \end{array}$$

$$\begin{array}{rcl} I(Utt) & = & Utt^{\mathcal{D}} \subseteq DP^{\mathcal{D}} \times M^{\mathcal{D}} \times (P^{\mathcal{D}} \cup Q^{\mathcal{D}}) \\ I(\mathbf{infl}) & = & infl \\ I(\mathbf{ans}) & = & ans \\ I(\mathbf{whether}) & = & whether : P^{\mathcal{D}} \to Q^{\mathcal{D}} \end{array}$$

Recall that stack variables range over strings of elements in the domain. Let Q^*, P^*, Utt^* denote the set of all finite-length strings over $Q^{\mathcal{D}}$, $P^{\mathcal{D}}$ and $Utt^{\mathcal{D}}$, respectively. This will be used later to model the stack variables in V_2.

4.2 The DGB Components

As mention earlier, in DL transitions between states are changes in variable assignment. We therefore represent the dynamic aspects of the information state as variables ranging over different domains. In particular, we use the variable names `FACTS`, `QUD` and `LM` to represent the three different components of the DGB. We also include two additional variables `UTT` and `PENDING`. New utterances are assigned to `UTT` and, in case the addressee cannot ground their content, they are also assigned to `PENDING`. This allows to distinguish between two kinds of grounding: content grounding (the value of `UTT` is assigned to `LM`) and proposition grounding (a proposition is incorporated onto `FACTS`).

To model content grounding we use a unary predicate G and assume that $G(x)$ only holds when the addressee of a particular utterance can ground its content. That is, according to the formalisation introduce in Section 4.1, $G(x)$ will be true in all those states where

$v(x) = 1$. As an abbreviation, we will write G when $G(x)$ and $v(x)=1$, and $\neg G$ otherwise.

Turn-taking is modelled by means of a variable *turn*, ranging over the set of dialogue participants. $turn = i$ indicates that DP i is the turn-holder, that is, that i is meant to be the speaker of the next utterance. We also introduce a special variable *Goal*, ranging over propositions. The (fairly simple) idea is that an IOD is driven by the goal of acquiring some piece of information, namely the proposition assigned to *Goal*.

One of the assumptions behind the DGB is that a realistic characterisation of context must allow for asymmetries between the information available to the different dialogue participants at a given point in a conversation. Thus, although the DGB attempts to represent the publicly accessible information at each state of the dialogue, it does so in terms of the collection of individual information states of the participants. In the current formalisation, however, only QUD, LM and PENDING are relative to each dialogue participant, while FACTS and UTT are unique. This is an obvious choice for the case of UTT, which is just used to hold new contributions publicly uttered by any dialogue participant. In the case of FACTS, however, this is a simplification motivated by the fact that the current formalisation only attempts to model simplified situations where FACTS is assumed to be empty at the initial state, and only propositions that have been commonly agreed on can be integrated into it. Thus, there is no room for disagreements in this respect, and the set of FACTS is always the same for the two dialogue participants.

We model QUD and PENDING as stacks, in a way that is very much inspired by QUD's actual implementation in the GoDiS dialogue system (Cooper et al. 2001). Although we think of FACTS as a set,[2] for technical reasons that will become clear below, we also model FACTS as a stack. On the other hand, UTT and LM range over utterances, i.e. triples (i, m, r), where i is interpreted as the speaker of u, m is the dialogue move performed by u and r represents its content. Formally:

$$\begin{array}{rclrcl} v(\texttt{QUD}_a) & \in & Q^* & v(\texttt{LM}_a) & \in & Utt^{\mathcal{D}} \\ v(\texttt{QUD}_b) & \in & Q^* & v(\texttt{LM}_b) & \in & Utt^{\mathcal{D}} \\ v(\texttt{PENDING}_a) & \in & Utt^* & v(\texttt{UTT}) & \in & Utt^{\mathcal{D}} \\ v(\texttt{PENDING}_b) & \in & Utt^* & v(turn) & \in & DP^{\mathcal{D}} \\ v(\texttt{FACTS}) & \in & P^* & v(Goal) & \in & P^{\mathcal{D}} \end{array}$$

The main reason why FACTS is modelled as a stack variable is that we want to be able to check whether a particular element (typically, the proposition assigned to *Goal*) is in FACTS. Modelling FACTS as a variable ranging over strings of propositions allows us to use the **pop** program to check whether some element x belongs to FACTS or not: if x is in FACTS and we pop the stack repeatedly, x will show up at some point as the head of the stack. Thus, we will use the notation $x \in \texttt{FACTS}$ as an abbreviation for $< \texttt{FACTS}.\mathbf{pop}^* > \ \mathbf{head}(\texttt{FACTS}) = x$.

5 Representing Interaction in IOD

Our main aim here is to show that the formalisation outlined in previous sections can be used to characterise the internal structure of conversational interaction. In particular, we restrict our account to *Inquiry-Oriented Dialogues* (IODs), in the sense of (Larsson, 2002).[3] Fairly abstractly, IODs can be described as (possibly discontinuous) question-answer sequences, typically driven by some information-seeking goal.

To formalise these conversational exchanges, we define complex DL programs (those shown in Table 4 and Table 5). After having introduced some abbreviations, in the remainder of this section we explain the programs in detail. Finally, we show how to specify the correctness of a program that is meant to characterise the class of IODs.

Abbreviations To simplify notation, the following abbreviations are used, where for all p q, $P(p)$ and $Q(q)$:

[2] Arguably, there are reasons to postulate some kind of order within the set of facts. See (Ginzburg, 1997) for an account of the restrictions on which contextually presupposed facts can serve as antecedents for some anaphoric elements.

[3] More precisely, to *non-negotiative* IODs.

$\mathbf{Ask}_a(q)$	$\mathtt{UTT} := (a, ask, q)$; $\mathtt{LM}_a := \mathtt{UTT}$; $\mathtt{QUD}_a.\mathbf{push}(q)$; $\mathbf{UPP}_b$; $turn := b$	$\mathbf{Clr}_a(q)$	$\mathtt{UTT} := (a, clr, q)$; $\mathtt{LM}_a := \mathtt{UTT}$; $\mathtt{QUD}_a.\mathbf{push}(q)$; $\mathtt{LM}_b := \mathtt{LM}_a; \mathtt{QUD}_b.\mathbf{push}(q)$; $turn := b$
$\mathbf{Ass}_a(p)$	$\mathtt{UTT} := (a, ass, p)$; $\mathtt{LM}_a := \mathtt{UTT}$; $\mathtt{QUD}_a\mathbf{push}(\mathbf{whether}(p))$; $\mathbf{UPP}_b$; $turn := b$	$\mathbf{Ack}_a(p)$	$\mathtt{UTT} := (a, ack, r)$; $\mathtt{LM}_a := \mathtt{UTT}$; $\mathtt{LM}_b := \mathtt{LM}_a$; $\mathtt{FACTS}.\mathbf{push}(p)$; $\mathtt{QUD}_a.\mathbf{pop}; \mathtt{QUD}_b.\mathbf{pop}$

Table 4: Moves

$$
\begin{array}{rcl}
\mathbf{Ask}(q) & = & \mathbf{Ask}_a(q) \cup \mathbf{Ask}_b(q) \\
\mathbf{Ass}(p) & = & \mathbf{Ass}_a(p) \cup \mathbf{Ass}_b(p) \\
\mathbf{Clr}(q) & = & \mathbf{Clr}_a(q) \cup \mathbf{Clr}_b(q) \\
\mathbf{Ack}(p) & = & \mathbf{Ack}_a(p) \cup \mathbf{Ack}_a(p)
\end{array}
$$

$$
\begin{array}{rcl}
\mathbf{AP} & = & \mathbf{AP}(p_1) \cup \ldots \cup \mathbf{AP}(p_n) \\
\mathbf{QP} & = & \mathbf{QP}(q_1) \cup \ldots \cup \mathbf{QP}(q_m) \\
\mathbf{UPP}_a & = & \mathbf{UPP}_a(q_1) \cup \ldots \cup \mathbf{UPP}_a(q_m)
\end{array}
$$

Also, we use $\mathbf{QP}^{\neq q}$ as the union of $\mathbf{QP}(q_i)$ for all q_i such that $Q(q_i)$ excluding q.

5.1 Moves and Protocols

The current formalisation models three different scenarios: asking and responding to a question, integrating a proposition into the commonly agreed facts, and asking for clarification when the content of an utterance has not been grounded. Following Ginzburg's work, we model these scenarios in the form of protocols: Querying Protocol (**QP**), Assertion Protocol (**AP**), and Utterance Processing Protocol (**UPP**). Protocols can be thought of as means to characterise the range of possible follow-ups in cooperative dialogue or, alternatively, as a representation of the obligations the dialogue participants are socially committed to (see (Traum and Allen, 1994; Kreutel and Matheson, 1999)). In the formalisation we present here, for instance, questions have to be answered (possibly after additional question-answer sub-dialogues) and propositions have to be acknowledged before being introduced into the commonly agreed facts.

As shown in Table 5, protocols are modelled as complex programs performed by conversational moves. Each conversational move is also represented by a compound program which defines the effects of a particular utterance as update operations on the DGB components. Table 4 shows the programs corresponding to our four dialogues moves for DP **a**.[4]

Following Ginzburg's account, when a dialogue participant a utters an utterance u, $\mathtt{LM}_a$ is updated with u. When the content of $\mathtt{LM}_a$ is a question q, q is pushed onto $\mathtt{QUD}_a$ ($\mathbf{Ask}(q)$). On the other hand, asserting a proposition p raises the question *whether* p for discussion. Thus, when the content of $\mathtt{LM}_a$ is a proposition p, $\mathbf{whether}(p)$ will be pushed onto $\mathtt{QUD}_a$ ($\mathbf{Ass}(p)$). At this stage, in $\mathbf{Ask}(q)$ and $\mathbf{Ass}(p)$ **UPP** is called. **UPP** checks whether the addressee of u can ground its content. If that is the case, then it updates her LM and QUD accordingly. Otherwise, if the addressee cannot ground the content of u, u will be put aside (i.e. pushed onto PENDING) and a clarification question will be posited.

For simplicity, we assume that clarification questions and acknowledgements are always understood, so in $\mathbf{Clr}(q)$ and $\mathbf{Ack}(p)$ the addressee's LM and QUD are updated accordingly, without need to call **UPP**. As formalised in $\mathbf{Ack}(p)$, uttering an acknowledgement has the effect of incorporating a proposition p that was under discussion into FACTS. Once p is pushed onto FACTS, $\mathbf{whether}(p)$ can be downdated from QUD.

The DL programs in Tables 4 and 5 can be

[4]The programs for DP **b** are defined accordingly, (i.e. substituting a for b, and the other way round). The same applies to $\mathbf{UPP}_b$.

$\mathbf{AP}(p)$	$\mathbf{Ass}(p)$; $\mathbf{Ack}(p)$
$\mathbf{QP}(q)$	$\mathbf{Ask}(q)$; $(\mathbf{AP} \cup (\mathbf{QP}^{\neq q}; \mathbf{AP}))$; $\mathtt{QUD}_a.\mathbf{pop}; \mathtt{QUD}_b.\mathbf{pop}$
$\mathbf{UPP}_a(q)$	$(G?; \mathtt{LM}_a := \mathtt{LM}_b;$ $\mathtt{QUD}_a.\mathbf{push}(\mathbf{head}(\mathtt{QUD}_b)) \cup$ $(\neg G?; \mathtt{PENDING}_a.\mathbf{push}(\mathtt{UTT});$ $turn := a;$ $\mathbf{Clr}(q); \mathbf{AP};$ $\mathtt{QUD}_a.\mathbf{push}(\mathbf{head}(\mathtt{PENDING}_a));$ $\mathtt{PENDING}_a.\mathbf{pop})$

Table 5: Protocols

seen as a means to express the rules underlying the structure of cooperative conversational interaction in terms of update operations on the DGB. However, we may also want additional properties, not derivable form the programs, to hold. For instance, according to $\mathbf{QP}(q)$, the appropriate follow-ups after an $\mathbf{Ask}(q)$ move are either an assertion (given that $\mathbf{Ass}(p)$ is the first move within $\mathbf{AP}(p)$) or a question (given that $\mathbf{Ask}(q)$ is the first move within $\mathbf{QP}^{\neq q}$). We may want to postulate that in cooperative dialogue the assertion should be an answer to the question, and that in case we reply to a question with another question, the second one should influence the first one.[5] Once the programs are defined, we can use them to formally write down these properties as formulas that we postulate to be valid.

$$\forall q\ (\mathbf{head}(\mathtt{QUD}_a) = \mathbf{head}(\mathtt{QUD}_b) = q \quad \rightarrow \\ \forall p\ [\mathbf{AP}(p)]\ \mathbf{ans}(p, q))$$

$$\forall q'\ (\mathbf{head}(\mathtt{QUD}_a) = \mathbf{head}(\mathtt{QUD}_b) = q' \quad \rightarrow \\ \forall q\ [\mathbf{QP}(q)]\ \mathbf{infl}(q, q'))$$

The following formula provides a fairly simple formalisation of turn-taking:

$$\forall ij\ (DP(i) \wedge DP(j) \wedge (i \neq j) \wedge (turn = i) \quad \rightarrow \\ \forall mr\ [\mathtt{UTT} := (j, m, r)]\ \bot)$$

[5] As is well known, the notion of *answerhood* and the *influences* relation are two rather complex concepts. This paper is not meant to provide any insights about these notions.

5.2 Correctness Specification

A sequence of states that can occur from the execution of a program α starting from a particular input state is called a *trace*. If we think of a dialogue as a sequence of states, then the set of all traces of a given program corresponds to the class of dialogues characterised by that program. In this sense, we can say that $\mathbf{QP}^*$ characterises the class of IODs. In this section we give a correctness specification for $\mathbf{QP}^*$.

Typically, a specification of correctness consists of a *precondition* φ and a *postcondition* ψ. We say that a program is *partially correct* with respect to a correctness specification φ/ψ if, whenever the program is started in a state satisfying φ then, if it halts, it does so in a state satisfying ψ. A program is *totally correct* with respect to a specification φ/ψ if (i) it is partially correct with respect to that specification, and (ii) it halts whenever it is started in a state satisfying φ.

We call a state *initial* if all stacks are empty. Let *Init* be a proposition which is true at all those states $s \in S$ such that valuation v_s assigns the empty string to all stack variables. Then, the following two formulas specify partial correctness (1) and termination (2) of $\mathbf{QP}^*$.

$$Init \ \rightarrow \ [\,\mathbf{QP}^*]\ \ Goal \in \mathtt{FACTS} \qquad (1)$$

$$Init \ \rightarrow \ <\mathbf{QP}^*>\ \top \qquad (2)$$

The conjunction of (1) and (2) specifies total correctness of $\mathbf{QP}^*$.

6 Conclusions and Future Work

In this paper we have explored the possibility of using DL to formalise IODs in terms the of Ginzburg's DGB. More specifically, we have put forward a model where the components of the DGB are represented by variables ranging over different domains, while update operations are brought about by program executions that involve changes in variable assignments. We have also given a correctness specification for a program which is intended to characterise the class of IODs.

Although this is still very much work in progress, we believe that the formalisation, even restricted to a particular kind of dialogues, shows that DL is an expressive and precise tool to formalise approaches to dialogue modelling based on information state updates.

There are many issues that remain still open, perhaps the most straightforward being how to use the formalisation presented here for instance to prove desirable properties of actual dialogue systems. Work along these lines has been done by Ljunglöf. In particular, (Ljunglöf, 2000) presents a calculus for reasoning mathematically about the rule-based engines developed within the TRINDI project. We expect to show in our future research that some version of DL can also be successfully used to provide precise specifications of dialogue systems based on information state approaches.

Acknowledgements We wish to thank two anonymous reviewers for CLUK6, Jonathan Ginzburg and, especially, Ulle Endriss for very helpful suggestions and discussion.

References

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

L. Carlson. 1983. *Dialogue Games*. Synthese Language Library. D. Reidel.

P. Cohen and H. Levesque. 1990. Rational interaction as the basis for communication. In P. Cohen, J. Morgana, and M. Pollack, editors, *Intentions in Communication*. MIT Press.

R. Cooper, S. Larsson, J. Hieronymus, S. Ericsson, E.Engdahl, and P. Ljunglof. 2001. Godis and questions under discussion. In *The TRINDI Book*.

R. Fernández and J. Ginzburg. 2002. Non-Sentential Utterances: A Corpus Study. *Traitement automatique des languages*, 43(2):13–42.

J. Ginzburg, H. Gregory, and S. Lappin. 2001. SHARDS: Fragment resolution in dialogue. In *Proceedings of the Fourth International Workshop on Computational Semantics*.

J. Ginzburg. 1996. Interrogatives: Questions, facts, and dialogue. In S. Lappin, editor, *Handbook of Contemporary Semantic Theory*. Blackwell, Oxford.

J. Ginzburg. 1997. Structural mismatch in dialogue. In *Proceedings of MunDial 97*. Universitaet Muenchen.

J. Ginzburg. 1999. Ellipsis resolution with syntactic presuppositions. In H. Bunt and R. Muskens, editors, *Computing Meaning: Current Issues in Computational Semantics*. Kluwer.

J. Ginzburg. ms. A semantics for interaction in dialogue. Forthcoming for CSLI Publications. Draft chapters available from: http://www.dcs.kcl.ac.uk/staff/ginzburg.

R. Goldblatt. 1992. *Logics of Time and Computation*. Lecture Notes. CSLI Publications.

B. Grosz and C. Sidner. 1990. Plans for discourse. In P. Cohen, J. Morgana, and M. Pollack, editors, *Intentions in Communication*. MIT Press.

C. L. Hamblin. 1970. *Fallacies*. Methuen, London.

D. Harel, D. Kozen, and J. Tiuryn. 2000. *Dynamic Logic*. Foundations of Computing Series. The MIT Press.

J. Kreutel and C. Matheson. 1999. Modelling questions and assertions in dialogue using obligations. In *Proceedings of Amstelog 99, the 3rd Workshop on the Sematics and Pragmatics of Dialogue*, Amsterdam.

S. Larsson. 2002. *Issue based Dialogue Management*. Ph.D. thesis, Gothenburg University.

D. Lewis. 1979. Score keeping in a language game. *Journal of Philosophical Logic*, 8:339–359.

P. Ljunglöf. 2000. Formalizing the dialogue move engine. In *Proceedings of the Götalog Workshop*.

M. D. Sadek. 1991. Dialogue acts as rational plans. In *Proceedings of the ESCA/ETR workshop on multi-modal dialogue*.

R. Stalnaker. 1979. Assertion. *Syntax and Semantics*, 9. Academic Press.

D. Traum and J. Allen. 1994. Discourse obligations in dialogue processing. In *Proceedings of the 32nd annual meeting of the ACL*.

D. Traum, J. Bos, R. Cooper, S. Larsson, I. Lewin, C. Matheson, and M. Poesio. 1999. A model of dialogue moves and information state revision. In *The TRINDI Book*.