Utterances as Update Instructions

Matthew Purver and **Raquel Fernández**
Department of Computer Science
King's College, London
Strand, London WC2R 2LS, UK
matthew.purver@kcl.ac.uk, raquel@dcs.kcl.ac.uk

Abstract

We present an approach to utterance representation which views utterances and their sub-constituents as instructions for contextual update: programs in a dynamic logic defined with respect to the dialogue gameboard of (Ginzburg, 1996; Fernández, 2003b). This approach allows utterance processing protocols to be represented within the grammar rather than postulated as separate dialogue processes: it also allows a view of incremental grounding and clarification which reflects the insights of dynamic semantics and allows us to treat salience and information structure in a coherent manner.

1 Introduction

The Information State (IS) approach to dialogue modelling has received much attention in recent years (see e.g. (Cooper et al., 1999; Larsson et al., 2000; Lemon et al., 2001)). This approach allows modelling of the information available to each participant at each stage of the dialogue, with updates to this information being defined in terms of update protocols postulated as part of one's general dialogue capability.

In this paper we discuss an alternative view of IS update processes: that updates can be defined as part of the grammatically conveyed content of utterances. We present an approach to utterance representation which views utterances and their sub-constituents as instructions for contextual update: programs in a dynamic logic defined with respect to the dialogue gameboard of (Ginzburg, 1996; Fernández, 2003b). We argue that this approach has several advantages: transferring the burden of definition from general protocols to utterance content allows us to simplify the protocols, transparently express the update rules as part of our linguistic/grammatical competence, and reflect the insight of Gricean pragmatics that utterance content includes the speaker's intended effect on the hearer. It also provides us with a framework within which each sub-utterance can give its own effect on the context, allowing us to reflect the basic insights of dynamic semantics and define a simple approach to grounding and clarification.

The rest of the paper is structured as follows: section 2 describes some theoretical background underlying our approach. Our proposal is then presented in section 3, and applied to concrete phenomena such as grounding and the given/new distinction. In section 4, we sketch the HPSG-based grammatical framework we assume. Section 5 and section 6 show how our approach can be extended to integrate update rules and clarification questions, respectively. We present our conclusions and directions for further work in section 7.

2 Background

2.1 The Dialogue Gameboard

We adopt the theory of dialogue context developed in the KOS framework (Ginzburg, 1996; Ginzburg, ms). In KOS, conversational interaction involves updates by each dialogue participant of her own *dialogue gamebord* (DGB), a data structure characterised by the following components: a set of FACTS, which the dialogue participants take as common ground; a partially ordered set of questions under discussion QUD; and the LATEST-MOVE made in the dialogue.

2.2 Grounding and Clarification

Following Ginzburg (1996)'s work, (Ginzburg and Cooper, 2001; Ginzburg and Cooper, forthcoming) present an analysis of clarification questions which motivates a highly contextually dependent representation of utterances. Utterance types are viewed as lambda-abstracts over a set of contextual parameters, i.e. as functions from context to content. In HPSG terms, this is achieved by introducing a new C-PARAMS feature, which encodes the set of contextual parameters of an utterance, and is amalgamated over syntactic daughters by a C-PARAMS Amalgamation Principle. The grounding process for an utterance then involves finding values for these contextual parameters. Failure to do this results in the formation of a clarification question.

2.3 The Utterance Processing Protocol

To account for how utterances get integrated into a dialogue participant's IS, Ginzburg and Cooper (2001) formulate a set of instructions – the Utterance Processing Protocol (UPP) – for a dialogue participant to update her IS, leading either to grounding or clarification.

```
1. Add U to PENDING
2. Attempt to ground U by instantiating
   each P in C-PARAMS
3. If successful:
      3(a). remove U from PENDING;
      3(b). add content(U) to LATEST-
         MOVE;
      3(c). If content(U) = assert(P):
            push whether(P) onto QUD
      3(d). If content(U) = ask(Q):
            push Q onto QUD
4. Else: form clarification question C(P
   ) and add ask(C(P)) to AGENDA
```

Listing 1: Utterance Processing Protocol

Listing 1 shows a simple version: utterances are first added to PENDING for the grounding process, which consists of attempting to identify the referents of the elements of C-PARAMS in context. Failure leads to the formation of a clarification question relevant to that parameter. Success leads to removal from PENDING and addition to LATEST-MOVE. In the case of assertions, a question relevant to the asserted proposition is also added to QUD; in the case of ask moves, the asked question is added to QUD; other moves such as orders may have their own specific actions specified (although in the case of seemingly "empty" moves such as greetings, they may not).

2.4 Formalising the DGB with Dynamic Logic

In (Fernández, 2003a; Fernández, 2003b) the DGB is formalised using first-order Dynamic Logic (DL) as it is introduced in (Harel et al., 2000) and (Goldblatt, 1992). Thus, Fernández (2003b) uses the paradigm of DL familiar from AI approaches to communication modelling[1] to formalise not motivational attitudes but information states and update processes on information states.

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. Programs are interpreted as relations between states. 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* ?.

Given that in DL transitions between states are changes in variable assignment, the components of the DGB are modelled as variables ranging over different domains, while update operations are brought about by program executions that involve changes in variable assignments. See (Fernández,

[1] For some DL-inspired formalisations within the Beliefs, Desires and Intentions (BDI) tradition, see e.g. (Cohen and Levesque, 1990; Sadek, 1991; Moore, 1995).

2003a; Fernández, 2003b) for the details of the formalisation, as well as for a basic introduction to first-order DL.

3 Utterances as Programs

Our approach in this paper is to view utterances as DL programs: as such, an utterance U denotes a transition between states $s \stackrel{U}{\longrightarrow} s'$. As long the program contains all relevant instructions for updating the IS, the UPP no longer needs to be specified separately – instead, we merely specify that an IS s can integrate an utterance U iff $\mathcal{M} \models_s < U > \top$ (i.e. U suceeds for the current IS).

3.1 About the formalism

Following (Fernández, 2003a; Fernández, 2003b), we use the variable names `FACTS`, `QUD` and `LM` to represent the three different components of the DGB. To distinguish between the information that has been commonly agreed on during the dialogue (stored in the initially empty `FACTS`) and the more general context available to a dialogue participant, we also include an additional variable `BG` (background).

We distinguish between individual variables ranging over terms, and *stack* variables ranging over strings of terms. The atomic programs we use to manipulate these variables are simple assignments ($x := t$), where x is an individual variable and t is a term; and `X`.**push**(x) and `X`.**pop** programs, where `X` is a stack variable (i.e. a string of elements) and x stands for the element to be pushed onto `X`.

`LM` is an individual variable ranging over moves; `QUD` is a stack variable ranging over strings of questions. Although we think of `BG` and `FACTS` as sets, we also model them as stack variables ranging over strings of terms. This allows us to use the **pop** program to check whether some term t belongs to `FACTS/BG` or not: if t is in `FACTS/BG` and we pop the stack repeatedly, t will show up at some point as the head of the stack. Thus, we will use the notation $t \in$ `FACTS/BG` as an abbreviation for $<$ `FACTS/BG`.**pop*** $>$ **head**(`FACTS/BG`) $= t$.

3.2 Replacing the UPP

The most basic form an utterance program can take is:

$$\texttt{LM} := m$$

thus assigning a conversational move m to `LM`, as per part 3(b) of the original UPP (see listing 1 above). The effects of parts 3(c,d) of the UPP are achieved by more complex programs for questions and assertions, which are sequences of atomic programs, as follows:

$$\texttt{LM} := ask(q); \texttt{QUD}.\mathbf{push}(q)$$

$$\texttt{LM} := assert(p); \texttt{QUD}.\mathbf{push}(whether(p))$$

We assume that these programs are assigned to utterances by the sentence grammar (we currently use a HPSG grammar which relates program to sentence type – see section 4). We can visualise the effect of the above as a transition between ISs:

$$\begin{bmatrix} \texttt{LM} & m_0 \\ \texttt{QUD} & qud_0 \end{bmatrix} \xrightarrow{\quad U \quad} \begin{bmatrix} \texttt{LM} & ask(q) \\ \texttt{QUD} & q \circ qud_0 \end{bmatrix}$$

Figure 1: Utterance as IS transition

In this way, the meaning associated with a particular move encapsulates the speaker's intention (in the case of an *ask* move, to introduce the asked question to `QUD`).

3.3 Grounding

This approach also allows us to formulate programs which achieve the same effect as Ginzburg and Cooper (2001)'s application of the utterance abstract to the context – namely the identification of referents in the contextual background. The program associated with an expression which requires such a referent to exist must be a program which finds that referent in context (i.e. that succeeds only when the referent is present). Given our formalism, this will be a program ($t \in$ `BG`)? (where ϕ? is a program which checks that ϕ holds in the current state).

We therefore propose that referential sub-utterances (e.g. certain NPs) can contribute such

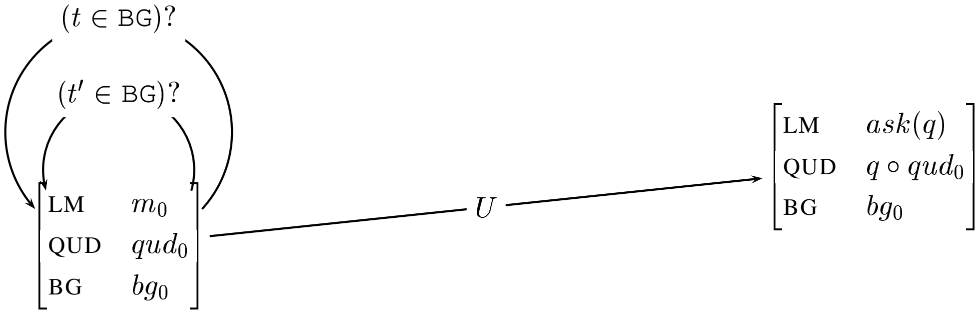

Figure 2: Grounding existing referents

programs to the utterance (again, the details of this must depend on the grammar – our HPSG approach uses a general amalgamation principle over syntactic daughters). A suitable representation for the sentence *"john snores"*, in which a referent for *john* must be found, might be the following complex program:

$$\begin{aligned}&(name(x, john) \in \texttt{BG})?;\\ &\qquad \texttt{LM} := assert(snore(x));\\ &\qquad \texttt{QUD}.push(whether(snore(x)))\end{aligned}$$

Here we take x to be a variable ranging over a finite set of people. In this case the test program $(name(x, john) \in \texttt{BG})?$ would succeed iff there is a *john* in BG and x happens to be assigned to *john* in the current world.[2] We are thus narrowing down the set of possible assignment functions to those that have this property.

In the same way, common nouns & verbs (which, following (Purver and Ginzburg, 2003) we take to refer to a predicate which must be identified) will require a predicate referent to be identified in context.

Thus, we take an utterance to be a program made up of a sequence of tests (that check whether the reference of particular expressions can be found in the context) followed by several atomic programs which update the relevant DGB components, as shown in figure 2. An utterance U can therefore be grounded iff $<U>\top$ holds.

3.4 The Given/New Distinction

This approach allows us to distinguish between *given* referents (e.g. definites) which must be *found* in context as above, and *new* referents (e.g. indefinites) which should be *added* to the context, by associating indefinites with a program which introduces a new referent (thus following the dynamic semantic tradition of (Heim, 1982; Kamp and Reyle, 1993; Groenendijk and Stokhof, 1991), and subsequent DRT-based dialogue theory such as (Poesio and Traum, 1998)).

"the dog": $(dog(x) \in \texttt{BG/FACTS})?$

"a dog": $\texttt{FACTS}.\textbf{push}(dog(x))$

Thus the program associated with a sub-utterance with a given referent tests for existence in the current state, while that for a new referent involves a state change introducing that referent.

This need not be restricted to definites and indefinites, but can be extended to the given/new distinction in general, including the information-structural focus/ground distinction. Following (Engdahl et al., 1999; Ginzburg, ms), we express this distinction by a condition on membership of QUD: that a particular focus/ground partition presupposes a corresponding question on QUD:

"JOHN snores": $(\lambda x.snore(x) \in \texttt{QUD})?$

"john SNORES": $(\lambda R.R(john) \in \texttt{QUD})?$

4 Grammar

We use an HPSG grammar similar to that of (Ginzburg and Sag, 2000), with the utterance programs assigned to a new feature C(ONTEXTUAL)-PROG(RAM); this replaces the C-INDICES feature of (Ginzburg and Sag, 2000), or the C-PARAMS feature of (Ginzburg and Cooper, 2001).

By default, this C-PROG feature is built up by phrases, by linear combination of the programs associated with its syntactic daughters, using the sequence operator as shown in AVM [1].

$$[1]\quad \begin{bmatrix}\text{C-PROG} & A;\ldots;B\\ \text{DTRS} & \left\langle \begin{bmatrix}\text{C-PROG} & A\end{bmatrix},\ldots,\begin{bmatrix}\text{C-PROG} & B\end{bmatrix}\right\rangle\end{bmatrix}$$

The default program associated with a phrase is therefore a purely sequenced combination of

[2] In fact, for proper names and definites, it will not be enough to require that there is *a* known referent: we need there to be a *unique/most salient* referent. The statement of a suitable test program will depend on one's theory of definiteness, but we do not see this as affecting our general approach.

the programs contributed by its daughters, but this default is overwritten for particular phrase types which by their nature make their own contributions to the overall program. For example, the clause type *root-clause* is specified to add the sub-program which updates LM:

$$[2]\quad \begin{bmatrix} \textit{root-clause} & \\ \text{CONT} & \boxed{1}\big[\textit{illoc-rel}\big] \\ \text{C-PROG} & A; \texttt{LM} := \boxed{1} \\ \text{HEAD-DTR} \mid \text{C-PROG} & A \end{bmatrix}$$

while clauses of type *declarative* (which have propositions as their semantic content) and *interrogative* (questions) add the sub-program which updates QUD:

$$[3]\quad \begin{bmatrix} \textit{declarative} & \\ \text{CONT} & \boxed{1}\big[\textit{proposition}\big] \\ \text{C-PROG} & A; \texttt{QUD}.\mathbf{push}(whether(\boxed{1})) \\ \text{HEAD-DTR} \mid \text{C-PROG} & A \end{bmatrix}$$

$$[4]\quad \begin{bmatrix} \textit{interrogative} & \\ \text{CONT} & \boxed{1}\big[\textit{question}\big] \\ \text{C-PROG} & A; \texttt{QUD}.\mathbf{push}(\boxed{1}) \\ \text{HEAD-DTR} \mid \text{C-PROG} & A \end{bmatrix}$$

Phrases which contribute given or new referents are specified in an entirely parallel way, e.g. for a definite NP:

$$[5]\quad \begin{bmatrix} \textit{definite} & \\ \text{CONT} & \boxed{1}\big[\textit{parameter}\big] \\ \text{C-PROG} & A; B; (\boxed{1} \in \texttt{BG/FACTS})? \\ \text{DTRS} & \Big\langle \big[\text{C-PROG} \quad A\big], \big[\text{C-PROG} \quad B\big] \Big\rangle \end{bmatrix}$$

The interaction with information structure is expressed at the top *root-clause* level:

$$[6]\quad \begin{bmatrix} \textit{root-clause} & \\ \text{CONT} & \boxed{1}\big[\textit{illoc-rel}\big] \\ \text{CTXT} \mid \text{IS} & \begin{bmatrix} \text{FOCUS} & \boxed{2} \\ \text{GROUND} & \boxed{3} \end{bmatrix} \\ \text{C-PROG} & A; (\lambda\boxed{2}.\boxed{3} \in \texttt{QUD})?; \texttt{LM} := \boxed{1} \\ \text{HEAD-DTR} \mid \text{C-PROG} & A \end{bmatrix}$$

In figure 3 we show an example derivation of a sentence: the resulting C-PROG program contains instructions for grounding a referent, for establishing the required information structure, and applying the UPP.

5 Integrating Update Rules

Typical IS-based dialogue system behaviour is defined by means of *update rules*, either stated in terms of plans and agendas (Larsson et al., 2000; Larsson, 2002) or in terms of obligations (Poesio and Traum, 1997; Poesio and Traum, 1998). The rules can be quite general but must state the expected behaviour for particular types of move: e.g. one might specify that a move which asks a question causes an obligation (or plan) to respond to the question to arise, that a greeting leads to an obligation (or plan) for a reciprocal greeting, and so on.

In section 3.2 we have seen that part of the speaker's intentions associated with particular move types (e.g. in the case of an *ask* move, to make the question "under discussion") can be specified within the grammar. We could actually go one step further and allow moves to introduce other update effects usually brought about by update rules, allowing us to replicate the rules of (Poesio and Traum, 1998) or pragmatic interpretations of (Stone, 2003).

Assuming that dialogue move types are integrated into the grammatical analysis of utterances (Ginzburg et al., 2001), the need for domain independent rules can be removed by specifying suitable programs as being directly associated with a particular move type. For example, an update rule such as the following (extracted from (Matheson et al., 2000)), would be replaced by the complex DL program below:

act	ID:2, **info_request**(DP,Q)
effect	push(OBL, **address**(**o**(DP),ID))

$$\texttt{LM} := info\_request(DP, Q);$$
$$\texttt{OBL}.\mathbf{push}(address(\mathbf{o}(DP), Q))$$

where OBL is a stack of obligations and $\mathbf{o}(DP)$ refers to *the other dialogue participant*.

A clear motivation for integrating certain contextual updates as part of the grammatical analysis of utterances is the existence of moves whose meaning can only be represented as an update of the dialogue context. An example of such moves are acknowledgements. Acknowledgements as 'okay' or 'uh-huh' are grounding acts with no further descriptive content associated. Thus, given

$$\begin{bmatrix} \textit{root-clause} & \\ \text{PHON} & \langle \text{JOHN, snores} \rangle \\ \text{CONT} & \boxed{1}\begin{bmatrix} \textit{assert-rel} & \\ \text{MSG-ARG} & \boxed{3} \end{bmatrix} \\ \text{CTXT} \mid \text{IS} & \boxed{2}\begin{bmatrix} \text{FOCUS} & \boxed{7} \\ \text{GROUND} & \boxed{4} \end{bmatrix} \\ \text{C-PROG} & \left[\boxed{5}; (\lambda\boxed{7}.\boxed{4} \in \texttt{QUD})?; \texttt{LM} := \boxed{1}\right] \end{bmatrix}$$

$$\begin{bmatrix} \textit{declarative} & \\ \text{PHON} & \langle \text{JOHN, snores} \rangle \\ \text{CONT} & \boxed{3}\begin{bmatrix} \textit{proposition} & \\ \text{SOA} & \boxed{4}snore(x) \end{bmatrix} \\ \text{CTXT} \mid \text{IS} & \boxed{2} \\ \text{C-PROG} & \boxed{5}\left[\boxed{6}; \texttt{QUD}.\textbf{push}(whether(\boxed{3}))\right] \end{bmatrix}$$

$$\begin{bmatrix} \text{PHON} & \langle \text{JOHN} \rangle \\ \text{CONT} & \boxed{7}\begin{bmatrix} \text{INDEX} & x \\ \text{RESTR} & name(x, john) \end{bmatrix} \\ \text{CTXT} \mid \text{IS} & \begin{bmatrix} \text{FOCUS} & \boxed{7} \end{bmatrix} \\ \text{C-PROG} & \boxed{6}\left[(\boxed{7} \in \texttt{BG/FACTS})?\right] \end{bmatrix} \quad \begin{bmatrix} \text{PHON} & \langle \text{snores} \rangle \\ \text{CONT} & \boxed{4} \\ \text{CTXT} \mid \text{IS} & \begin{bmatrix} \text{GROUND} & \boxed{4} \end{bmatrix} \end{bmatrix}$$

Figure 3: Example derivation: *"JOHN snores"*

our formalism, acknowledgements will be assigned the following program by the grammar:

$$\texttt{LM} := ack; \texttt{FACTS}.\textbf{push}(\textbf{head}(\texttt{QUD})); \texttt{QUD}.\textbf{pop}$$

Although one may argue that this approach contributes to a more unmodular theory of dialogue, we think that one of the advantages of the framework we present here is that it offers a means of specifying the contextual effect of utterances in a general fashion, from sub-utterances to dialogue acts.

6 Clarification

The approach to grounding in section 3.3 is simplistic in that it does not consider the possibility of clarification of a referent that cannot be identified: the program U will simply fail in such cases. We can take this possibility into account by modifying the programs associated with sub-utterance referents. Such programs, rather than being simple instructions to find a referent, now become instructions to find a referent *or* determine it via clarification if one cannot be found. The program associated with *"john"* will therefore take the form:

$$\textbf{while}(\neg(name(x, john) \in \texttt{BG/FACTS}), \textbf{clarify}(S))$$

where S is the *sign*[3] corresponding to the sub-utterance. Here, $\textbf{while}(\phi, \alpha)$ abbreviates $(\phi?; \alpha)^*; \neg\phi?$, and $\textbf{clarify}(t)$ is a program which governs generation of a clarification question relevant to a term t, together with interpretation of any answer given. Its form therefore depends on one's overall system philosophy; for a plan-based approach, it might take the form:

$$\texttt{AGENDA}.\textbf{push}(ask(CQ(t))); \alpha_T$$

where $CQ(t)$ is a clarification question relevant to the sign t, and α_T is the program which governs the generation & production of the next turn together with the interpretation of its response.

Note that this sub-program will only succeed once the required information is present in

[3] As (Ginzburg and Cooper, 2001) point out, clarification questions must take into account many properties (including phonology, syntax etc.): we therefore assume all sign information is required.

FACTS, allowing the overall utterance program to continue (eventually e.g. updating QUD). Note also that this may result in LM now being set to the initial move (rather than the latest move in any clarification sequence) – but this is the behaviour we desire, as it is this move that is now being responded to.[4]

We can further modify this to take account of the possibility of *accommodation* of the referent (adding it to the common ground without clarification):

$$\textbf{while}(\neg(name(x, john) \in \texttt{BG/FACTS}),$$
$$\textbf{clarify}(S) \cup \texttt{FACTS}.\textbf{push}(name(x, john)))$$

Note that this treatment need not be confined to programs associated with sub-utterances that require given referents to be identified (e.g. names and definites), but can be used in general to account for clarification caused by the failure of any program (including those for indefinites, focus/-ground partition, and the overall move made) in a uniform way.

7 Summary

7.1 Conclusions

- This approach reduces the amount of our dialogue competence which is specified in processing protocols / update rules, and instead specifies it as part of the grammar. Utterance processing now merely consists of applying the state changes specified by the utterance itself.
- The resulting representation expresses the Gricean intuition that interpretation depends upon recognising the speaker's intention: utterance meaning includes the intended effect on the hearer (e.g. that they add the move to their IS, and make the desired question "under discussion").
- The approach also allows sub-utterances to specify their effect on the context, allowing us to express the given/new distinction neatly, and define a process for grounding & clarification.

[4]If one has a different view of the role of LM, a program can be constructed that avoids this.

- The representation therefore allows all immediate contextual effects of an utterance to be specified on the same level: from the update effect of NPs familiar from dynamic semantics, to the update effect of utterances familiar from IS-based theories of dialogue.

7.2 Further Work

The work presented in this paper is an initial proposal for the representation of utterances within a dialogue grammar. In order to investigate it further, we plan:

- A thorough study of dialogue update rules across systems/approaches to determine the level of domain-dependence and thus the extent to which specification in the grammar is desirable;
- Extension to other dialogue phenomena such as revision;
- Extension of our HPSG grammar fragment and implementation within a prototype dialogue system.

7.3 Acknowledgements

The authors would like to thank Jonathan Ginzburg, Ulle Endriss and the anonymous DiaBruck reviewers for several helpful comments. They are supported by EPSRC grant GR/R04942/01 and ESRC grant RES-000-23-0065, respectively.

References

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

Robin Cooper, Staffan Larsson, Massimo Poesio, David Traum, and Colin Matheson. 1999. Coding instructional dialogue for information states. In *Task Oriented Instructional Dialogue (TRINDI): Deliverable 1.1*. University of Gothenburg.

Elisabet Engdahl, Staffan Larsson, and Stina Ericsson. 1999. Focus-ground articulation and parallelism in a dynamic model of dialogue. In *Task Oriented Instructional Dialogue (TRINDI): Deliverable 4.1*. University of Gothenburg.

Raquel Fernández. 2003a. A dynamic logic formalisation of inquiry-oriented dialogues. In *Proceedings of the 6th CLUK Colloquium*, pages 17–24, Edinburgh. CLUK.

Raquel Fernández. 2003b. A dynamic logic formalisation of the dialogue gameboard. In *Proceedings of the Student Research Workshop, EACL 2003*, pages 17–24, Budapest. Association for Computational Linguistics.

Jonathan Ginzburg and Robin Cooper. 2001. Resolving ellipsis in clarification. In *Proceedings of the 39th Meeting of the ACL*, pages 236–243. Association for Computational Linguistics, July.

Jonathan Ginzburg and Robin Cooper. forthcoming. Clarification, ellipsis, and the nature of contextual updates. *Linguistics and Philosophy*.

Jonathan Ginzburg and Ivan Sag. 2000. *Interrogative Investigations: the Form, Meaning and Use of English Interrogatives*. Number 123 in CSLI Lecture Notes. CSLI Publications.

Jonathan Ginzburg, Ivan A. Sag, and Matthew Purver. 2001. Integrating conversational move types in the grammar of conversation. In P. Kühnlein, H. Rieser, and H. Zeevat, editors, *Proceedings of the Fifth Workshop on Formal Semantics and Pragmatics of Dialogue (BI-DIALOG 2001)*, pages 45–56.

Jonathan Ginzburg. 1996. Interrogatives: Questions, facts and dialogue. In S. Lappin, editor, *The Handbook of Contemporary Semantic Theory*, pages 385–422. Blackwell.

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

Robert Goldblatt. 1992. *Logics of Time and Computation*. Number 7 in Lecture Notes. CSLI Publications.

Jeroen Groenendijk and Martin Stokhof. 1991. Dynamic predicate logic. *Linguistics and Philosophy*, 14(1):39–100.

David Harel, Dexter Kozen, and Jerzy Tiuryn. 2000. *Dynamic Logic*. Foundations of Computing Series. The MIT Press.

Irene Heim. 1982. *The Semantics of Definite and Indefinite Noun Phrases*. Ph.D. thesis, University of Massachusetts at Amherst.

Hans Kamp and Uwe Reyle. 1993. *From Discourse To Logic*. Kluwer Academic Publishers.

Staffan Larsson, Peter Ljunglöf, Robin Cooper, Elisabet Engdahl, and Stina Ericsson. 2000. GoDiS - an accommodating dialogue system. In *Proceedings of ANLP/NAACL-2000 Workshop on Conversational Systems*.

Staffan Larsson. 2002. *Issue-based Dialogue Management*. Ph.D. thesis, Göteborg University.

Oliver Lemon, Anne Bracy, Alexander Gruenstein, and Stanley Peters. 2001. Information states in a multimodal dialogue system for human-robot conversation. In P. Kühnlein, H. Rieser, and H. Zeevat, editors, *Proceedings of the Fifth Workshop on Formal Semantics and Pragmatics of Dialogue*, pages 57–67. BI-DIALOG.

Colin Matheson, Massimo Poesio, and David Traum. 2000. Modeling grounding and discourse obligations using update rules. In *Proceedings of the First Annual Meeting of the North American Chapter of the ACL*, Seattle, April.

Robert C. Moore. 1995. *Logic and Representation*. Lecture Notes. CSLI Publications.

Massimo Poesio and David Traum. 1997. Conversational actions and discourse situations. *Computational Intelligence*, 13(3).

Massimo Poesio and David Traum. 1998. Towards an axiomatization of dialogue acts. In J. Hulstijn and A. Nijholt, editors, *Proceedings of the Twente Workshop on the Formal Semantics and Pragmatics of Dialogues*, pages 207–222, Enschede, May.

Matthew Purver and Jonathan Ginzburg. 2003. Clarifying nominal semantics in HPSG. In *Proceedings of the 10th International Conference on Head-Driven Phrase Structure Grammar (HPSG-03)*, East Lansing, Michigan, July. Michigan State University.

M. D. Sadek. 1991. Dialogue acts as rational plans. In *Proceedings of the ESCA/ETR Workshop on Multimodal Dialogue*.

Matthew Stone. 2003. Linguistic representation and gricean inference. In *IWCS-5: Proceedings of the Fifth International Workshop on Computational Semantics*, pages 5–21, University of Tilburg, January.