Information State Update: Semantics or Pragmatics?

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

Abstract

We argue for an approach which treats the compositional semantic content of an utterance as including its basic dialogue update effects – those which can be derived entirely from its semantic and syntactic properties. This allows us to capture the distinction between these integral *semantic* contextual effects and those *pragmatic* effects which can only be determined from the interaction between features of the utterance and the context itself.

1 Introduction

This paper presents an approach to dialogue update processes that captures the distinction between that part of an utterance's contextual import that can be derived entirely from its semantic and syntactic properties, and that which results from interaction between features of the utterance and the context – by treating the former as part of the utterance's compositional *semantic* content and only the latter as having to be specified separately as *pragmatic* processes. We then show that this does not prevent an utterance's representation from articulating constraints on context, and illustrate this for context-dependent phenomena such as givenness and ellipsis.

1.1 Background

We adopt the approach to utterance representation introduced in (Purver and Fernández, 2003), which views utterances and their sub-constituents as instructions for contextual update: programs in a dynamic logic defined with respect to the dialogue gameboard (DGB) of (Ginzburg, 1996). The DGB provides a structured view of context in dialogue by keeping track of the following components: a set of commonly accepted FACTS; a partially ordered set QUD of questions under discussion (QUDs); and the LATEST-MOVE (LM) made in the dialogue.

In (Fernández, 2003), the DGB is formalised using first-order Dynamic Logic (DL) as it is introduced in (Harel et al., 2000). In short, DL is a multi-modal logic with a possible worlds semantics, which distinguishes between *formulae* and *programs*. Programs are interpreted as relations between states that 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* ?. The different DGB components are modelled either as individual variables ranging over terms (e,g. LM, for the latest move), or as *stack* variables ranging over strings of terms (e,g. QUD, a stack of questions). Update operations are brought about by program executions that involve changes in variable assignments. The atomic programs 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 and x stands for the element to be pushed onto X. Such programs can then be assigned to utterances and their sub-components by a HPSG grammar which relates programs to

grammatical types. The approach allows us to reflect the basic insights of dynamic semantics (e.g. indefinite NPs can be assigned programs which introduce new referents) and define a process of grounding and clarification, as well as specify update effects of utterances familiar from Information State (IS)-based theories of dialogue.

2 Update Programs

The basic assumption underlying the IS approach to dialogue modelling is that the main aspects of dialogue management are best captured by (i) keeping track of the relevant information available to each dialogue participant at each state of the conversation, and (ii) providing a full account of the possible update mechanisms that change this information. The notion of IS *update* is key, usually being governed by a set of *update rules* triggered by the observation and performance of dialogue moves.

Our starting point is, in fact, a fairly straightforward extension of this view: as long as dialogue move types can be incorporated into the grammatical representation of utterances, their update effects can also be seen as part of the utterance's linguistically conveyed information. The integration of direct illocutionary force into the grammar has been argued for in (Ginzburg et al., 2001b). The authors present an HPSG grammar where each illocutionary type introduces a constraint on the type of its message argument (*ask-rel* types are associated with *questions*, *assert-rel* types with *propositions*, and so on), with these message types being determined by syntactic form. SDRT (Asher and Lascarides, 2003) also assumes a *uniform semantics* of declaratives, interrogatives and imperatives, where each clause type is linked to its illocutionary force by means of compositional semantics. Our approach goes one step further in that it views the immediate contextual effects of these various illocutionary types (which are usually seen as brought about by independent IS update rules or pragmatic inference) as compositionally linked to syntactic and semantic properties of utterances. In (Purver and Fernández, 2003) this is achieved by associating appropriate DL programs with particular clause types, as shown inAVM (1) and AVM (2) for interrogatives and declaratives:

(1)
$$\begin{bmatrix} \textit{interrogative} & \\ \textsc{cont} & \boxed{1}\left[\textit{question}\right] \\ \textsc{c-prog} & A;\, \texttt{QUD}.\textbf{push}(\boxed{1}) \\ \textsc{head-dtr} \mid \textsc{c-prog} & A \end{bmatrix}$$

(2)
$$\begin{bmatrix} \textit{declarative} & \\ \textsc{cont} & \boxed{1}\left[\textit{proposition}\right] \\ \textsc{c-prog} & A;\, \texttt{QUD}.\textbf{push}(whether(\boxed{1})) \\ \textsc{head-dtr} \mid \textsc{c-prog} & A \end{bmatrix}$$

Introducing DL programs into the grammatical representation of utterance types allows us to reflect the part of their contextual import which is compositionally derivable. Just as an indefinite NP intrinsically introduces a new entity into the context, *ask* moves, and therefore questions, intrinsically introduce new QUDs (in our formalisation, push their content onto QUD). Similarly, moves which assert a proposition p push the question $whether(p)$ onto QUD. Note that this is not to deny that some questions and assertions might have further contextual effects, or even that QUD introduction might also be achievable by other, less obvious means. The point here is that an important part of the context change potential of *ask* and *assert* moves (namely the fact that they introduce particular QUDs) can be fully derived from their grammatical properties. As far as dialogue goes, it is therefore possible and, we think, desirable to consider such updates as part of the semantic contribution of interrogative and declarative utterances (just as much as the introduction of new referents is part of the semantic contribution of indefinites). This is precisely what our programs achieve.

The main issue to consider now is: can this approach be extended to all dialogue move types? In other words, is it possible to encode the main contextual updates brought about by dialogue moves into the grammatical representation of utterances, thus removing the need for independent update rules?

3 Semantic vs. Pragmatic Updates

Here we must distinguish two different kinds of updates: updates whose assignment can be determined purely by properties of the utterance itself, and those which should only be assigned to ut-

terances given certain information in the current state.

3.1 Semantic (Direct) Updates

The immediate update effects of direct moves such as *ask* and *assert* (as given above) can be determined by simple examination of the linguistic properties of utterances – they don't have to be inferred using pragmatic information. The same can be said for many other move types included in dialogue act taxonomies such as *greetings*, *closings* and *acknowledgements*. In fact, one could argue that the meaning of an acknowledgement can *only* be represented as a contextual update – in our approach, acknowledgements are associated with a program that pushes onto FACTS whatever proposition was previously under discussion:

(3)
$$\begin{bmatrix} \textit{ack-cl} & \\ \text{CONT} & \textit{acknowledge-rel} \\ \text{C-PROG} & \textbf{head}(\texttt{QUD}) = whether(p)?; \\ & \texttt{FACTS}.\textbf{push}(p); \texttt{QUD}.\textbf{pop} \end{bmatrix}$$

The complex program shown in AVM (3) requires for its success the existence of some question $whether(p)$ under discussion. If there is no such question, the program will not succeed, the utterance cannot be understood or grounded (and on our account, will cause a clarification question). This seems correct: if there is nothing to be integrated into the common ground, or if the current QUD is a wh-question, an acknowledgement will seem quite odd. Acknowledgements require suitable QUDs in order to be understood (just as understanding an *ask* move seems to require recognition of its intention to raise a new QUD). It is important to note that although an acknowledgement therefore imposes a restriction on the type of state to which it can apply (expressed as a *test* subprogram), there is no need for pragmatic information to determine what its update effects should be (what program to associate with it).

3.2 Pragmatic (Indirect) Updates

Most dialogue act taxonomies and implemented dialogue systems include other move types which are less directly associated with the linguistic or grammatical form of the utterance. Indirect speech acts such as requests or commands can take the form of questions (*"Can you close the door please?"*); questions can be rhetorical (*"Do I look like an idiot?"*).

Answers in the Grammar? A common example in dialogue systems is an *answer* move. Answers differ from assertions and questions in many respects: if we were to specify the contextual update effect of an answer by a program, it might be of the form QUD.**pop** – i.e. a program that downdates QUD by popping the maximal question under discussion, rather than one which adds a new question to the stack. The notion of answerhood employed by many dialogue systems involves assertion of a proposition that unifies with the propositional content of a QUD question (see e.g. (Traum, 2003)). This could be easily defined within the grammar as in (4):

(4)
$$\begin{bmatrix} \textit{answer} & \\ \text{CONTENT} & \boxed{1}[\textit{proposition}] \\ \text{C-PROG} & (\textbf{head}.(\texttt{QUD}) = \lambda\{\ldots\}.\boxed{1})?; \texttt{QUD}.\textbf{pop} \end{bmatrix}$$

The problem is of course that there will be no way of associating this program with an utterance based on its internal grammatical properties alone: to determine which update effects to associate with a declarative (those of an *assert* program as in (2) or an *answer* program as in (4)), we must take into account its relation to some relevant contextual information (precisely the maximal QUD). Given our formalism, this could be phrased as a single program using the *choice* operator:[1]

(5)
$$\begin{bmatrix} \textit{declarative} & \\ \text{CONTENT} & \boxed{1}[\textit{proposition}] \\ \text{C-PROG} & ((\textbf{head}.(\texttt{QUD}) = \lambda\{\ldots\}.\boxed{1})?; \texttt{QUD}.\textbf{pop}) \\ & \cup\, \texttt{QUD}.\textbf{push}(whether(\boxed{1})) \end{bmatrix}$$

However, we see several problems with such an approach. The first is that downdating QUD, which must be one of the update effects of an answer, does not need to be performed in order to *understand* it: one can understand an answer without accepting it, and indeed can discuss whether it is true – so making this downdate part of the semantic content seems inappropriate. A second is that the contextual effects expressed as the se-

[1] Another equally unattractive solution would be to see *all* declarative sentences as ambiguous between being answers and assertions, with two alternative analyses assigned by the grammar and with the decision between the two made later.

mantic content have now become *determined by* context, rather than just expressing *restrictions on* context as before. Even worse, the third is that this approach seems very difficult to scale up to more complex notions of answerhood: in particular, indirect answers could not be detected by the test of unification with the head QUD as above, but would require some further inference – thus the C-PROG program would have to involve such inference and presumably access to further contextual information. Semantic content, then, would not only be context-dependent but include (possibly unrestricted) access to pragmatic components.

Note that this is not the case for acknowledgements: the program in AVM (3) shows no contextually determined variation in its possible effects – the program simply imposes a restriction on the current state that has to be met for the update program to be executable: if the restriction is not met, the program will just fail.

Answers outside the Grammar A more reasonable approach therefore seems to be to take answers as having update effects at two levels: at the *direct* level, expressible as part of the grammatically assigned semantic content, the effect of an assertion as in AVM (2) (introducing $whether(p)$ to QUD); and then at the *indirect* level the further answering effect (popping the QUD stack). This indirect effect must be outside the realm of grammar, as its applicability will depend on the current IS – reasoning or update rules must decide whether p answers the current maximal QUD, and if so whether it is to be accepted.

The semantic content then no longer varies with context (although it can still express a restriction on context as before), and can be grammatically assigned as long as this basic program is not inconsistent with the possible later indirect updates. For answers, the basic effect is an assertion which is then used to license QUD downdate; for rhetorical questions, the basic effect would be to introduce a new QUD which is seen to be already answered (by domain/world knowledge or context) and thus immediately downdated; for indirect requests, again the basic effect would be to introduce a new QUD, which further inference would then presumably determine to be influenced by the indirectly requested task (see (Ludwig, 2001) for a similar approach to inferring requests from basic declaratives).

This distinction, between direct updates which stem from the utterance's internal properties on the one hand, and indirect updates which stem from its relation to context on the other, now allows us to draw a line between the kind of updates that can be thought of as part of an utterance's *semantic content*, and those that should be specified separately by means of *pragmatic* operations (e.g. update rules or inference). Note that this distinction does not correspond to the one drawn between forward and backward looking acts (Allen and Core, 1997) – acknowledgements and answers are both usually classified as backward-looking. In a typical system such as GoDiS (Larsson et al., 2000) the only move type which requires separate pragmatic processes (and which we would therefore classify as indirect) is *answer*.

3.3 Discourse & Turn-Taking Effects

So far we have assumed that the direct update effect of questions and assertions is to introduce a question which becomes topmost in QUD (q, in the case of asking a question q, and $whether(p)$ in the case of asserting a proposition p). In Ginzburg's account, this topmost position is taken to explicate why the last question posed takes conversational precedence (has to be addressed first) and why elliptical forms are licensed as responses to it. Several authors (Asher, 1998; Ginzburg, forthcoming), however, have pointed out that when multiple moves are performed by the same speaker within a single turn, the evolution of QUD seems to be somewhat different.

(6)
A : Where were you? Did you talk to anyone?
B : I was at home. I didn't talk to anyone.
B': I didn't talk to anyone, I was at home.

(7)
A : Who did you invite? Did you invite Jill?
B : Yes. Also Merle and Pat.

(8)
A : Who did you invite? And why?
B : Merle and Pat, because they are very undemanding folks.

Examples like the ones above have motivated a view according to which the way several queries

asked in sequence by the same speaker are integrated into QUD depends on the discourse relation that links them. Thus, the questions in example (6) (adapted from (Asher, 1998)) are taken to be in what has been called *coordinate structure*, with none of them taken precedence over the other one. The questions in (7), on the other hand, would be related by *query-elaboration*, which would account for the fact that apparently the second one takes precedence over the first one. Contrastingly, the questions in example (8) would be related by *query-extension*, which would explain why in this case the first question tends to be answered first.

At a first glance, one may think that three different QUD updating operations are needed to account for these examples: one that pushes the second question next to the maximal QUD, the standard push on top operation, and a "push-under" operation (or *QUD-FLIP*, as it is called by Ginzburg (forthcoming)) that would push the second question under the topmost element in QUD. If we were to specify these distinctions in our account, we would presumably have to do so by a program that first tests the kind of rhetorical relation that holds between the head of QUD and the current question, and then applies the right QUD.**push** program, as in AVM (9):

(9)
$$\begin{bmatrix} \textit{interrogative} & \\ \text{CONT} & \boxed{1}\left[\textit{question}\right] \\ \text{C-PROG} & \textit{q-elab}(\textbf{head}(\text{QUD}),\boxed{1})?;\text{QUD}.\textbf{push}(\boxed{1})\cup \\ & \textit{coor}(\textbf{head}(\text{QUD}),\boxed{1})?;\text{QUD}.\textbf{push-next}(\boxed{1})\cup \\ & \textit{q-ext}(\textbf{head}(\text{QUD}),\boxed{1})?;\text{QUD}.\textbf{push-under}(\boxed{1}) \end{bmatrix}$$

However, as with answers in the previous section, the test subprograms in AVM (9) not only express restrictions on the kind of state the program can be applied to (like the program for acknowledgements in AVM (3) above), but crucially they both require further pragmatic information, and use it to determine the program's effects. To decide on the kind of push program that has to be applied, we must first compute the rhetorical relation that holds between the current question and the maximal QUD, and this will involve using pragmatic reasoning. Thus, to use a grammar to assign the complex program in AVM (9) to interrogative clauses seems both problematic and rather pointless, given that its update effects are actually ambiguous between three different QUD updating operations, and such ambiguity is only resolved by pragmatic knowledge about rhetorical relations.

Instead, we think that the *semantic* update effects of questions and assertions are still best characterised by the simple programs proposed in AVMs (1) and (2). In fact, a closer look at the examples above reveals that the discourse relations that link different questions in the same turn do not play such a significant role in determining availability and licensing of elliptical forms:

(10) A : Who did you invite? Did you invite Jill?
B': (I invited) Merle, Pat, and Jill, yes.

(11) A : Who did you invite? And why?
B': I thought we'd need a guitar, so Merle.

As (10) and (11) show, regardless of the rhetorical relation that holds between the questions, both questions are still available: they can be answered by a fragment and it is up to the addressee to choose which one to answer first. We can therefore assume that the basic QUD update mechanism (and therefore our basic programs) do not require, and need not be affected by, computation of the rhetorical relation that links moves within a single turn. This is not to claim that discourse relations are not needed at any level: they may be required to establish the coherence of the dialogue at the pragmatic level, or indeed to decide which member of QUD to answer first. We do claim however that one can still specify some basic *semantic* contextual update potential brought about by questions and assertions as monotonically introducing QUDs.

QUDs introduced in the same turn must then have equal satus in the QUD stack. We regard this coordinate status as a consequence of the dynamics governing turn management. It is implicitly assumed that information about turn taking and turn change is part of the resources commonly shared by dialogue participants. To encode this information explicitly in the dialogue context, here we assume that QUD not only includes the questions under discussion themselves, but also information on turn change that acts as an additional structuring mechanism of the QUD order. Assuming that

turn change is recorded in QUD, the maximal elements of QUD are then those questions between the top and the turn change indicator.[2]

4 The Contextual Interface

As we have seen, *semantic* update programs such as acknowledgements can specify restrictions on the current state, without requiring state information to determine their form. How does this distinction apply for other contextually-dependent phenomena such as ellipsis?

4.1 Conditions on State

Some interaction between the utterance representation and the context is required not only by moves like acknowledgements, but by the treatment of givenness: given referents such as those associated with definite NPs and proper names contribute sub-programs which express restrictions on the type of state to which the utterance program can be successfully applied – to whit, that the state contain a suitable antecedent (seeAVM (12) and AVM (13)). The same is true for other givenness effects such as the focus/ground distinction: following (Engdahl et al., 1999; Ginzburg, forthcoming) a particular focus/-ground partition introduces a sub-program which must find a particular maximal QUD in the current state. This type of program, then, expresses a condition on the kind of state to which it can apply: in other words, the kind of context in which an utterance is licensed.

(12)
$$\begin{bmatrix} \textit{definite} & \\ \textsc{cont} & \boxed{1}[\textit{parameter}] \\ \textsc{c-prog} & (\boxed{1} \in \texttt{BG/FACTS})? \end{bmatrix}$$

(13)
$$\begin{bmatrix} \textit{root-clause} & \\ \textsc{info-struct} & \begin{bmatrix} \textsc{focus} & \boxed{2} \\ \textsc{ground} & \boxed{3} \end{bmatrix} \\ \textsc{c-prog} & (\mathbf{head}(\texttt{QUD}) = \lambda\boxed{2}.\boxed{3})? \end{bmatrix}$$

4.2 Fragments

Elliptical fragments can also be seen in this way: as being licensed only in certain types of context, and therefore as expressing conditions on the kind of state to which their programs can apply. Fragments, of course, specify their content only partly, requiring the presence of some information in context in order to resolve their fully specified sentential content. Ginzburg et al. (2001a) analyse this by use of two contextual features in their HPSG grammar, MAX-QUD and SAL-UTT: the content of a fragment is specified in terms of constraints on these, by identifying the propositional content of the elliptical utterance with that of MAX-QUD and the referential index of the fragment itself with that of SAL-UTT. Until resolution in context, this information is essentially underspecified. Schlangen (2003), on the other hand, regards the content of such an elliptical utterance as containing an unknown anaphoric propositional relation, which must be enriched using contextual inference.

Instead, we regard elliptical fragments as introducing sub-programs which must ensure that the required contextual information is present in the current state, and by finding it, fully instantiate the content. The grammatical approach can directly follow that of Ginzburg et al. (2001a): the content of a (declarative) elliptical fragment utterance is taken to be a proposition which must be associated with the current MAX-QUD question; the referential index of its head daughter must be identified with that of a SAL-UTT utterance which is also constrained to be syntactically parallel to it. This is expressed in the grammar via the type *decl-frag-cl* (see AVM (14)).[3]

(14)
$$\begin{bmatrix} \textit{decl-frag-cl} & \\ \textsc{content} & \boxed{1} \\ \textsc{head-dtr} & \begin{bmatrix} \textsc{cat} & \boxed{2} \\ \textsc{cont}\,|\,\textsc{index} & \boxed{3} \end{bmatrix} \\ \textsc{context} & \begin{bmatrix} \textsc{max-qud} & [\textsc{prop} \quad \boxed{1}] \\ \textsc{sal-utt} & \begin{bmatrix} \textsc{cat} & \boxed{2} \\ \textsc{cont}\,|\,\textsc{index} & \boxed{3} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Now, the only change that must be made is that top-level sentences (in our grammar, signs of type *root-cl*) must add sub-programs which require the specified contextual information to be found, as shown in AVM (15).[4] Note that the order of the

[2] A way of implementing this idea is to think of QUD as a stack of sets. See (Fernández and Endriss, ms) for a formalisation of this in the context of dialogue protocols.

[3] Similar specifications can be given for short interrogatives, sluices, bare adjuncts and so on following (Fernández et al., 2004) directly.

[4] This *root-cl* specification also includes a sub-program to

program is important: the contextual information must be identified in the initial state, before it is changed by the utterance program (which may of course update QUD), and of course before the LM state variable can be set to the fully specified move (the overall utterance content).

(15)
$$\begin{bmatrix} \textit{root-clause} & \\ \text{CONTENT} & \boxed{1}\left[\textit{illoc-rel}\right] \\ \text{CONTEXT} & \begin{bmatrix} \text{MAX-QUD} & \boxed{2} \\ \text{SAL-UTT} & \boxed{3} \end{bmatrix} \\ \text{C-PROG} & (\textbf{head}(\texttt{QUD}) = \boxed{2})?; \\ & (\textbf{head}(\texttt{UTT}) = \boxed{3})?; A; \texttt{LM} := \boxed{1} \\ \text{HEAD-DTR} \mid \text{C-PROG} & A \end{bmatrix}$$

This seems to make the status of this contextual information clearer than in either Ginzburg et al. (2001a) or Schlangen (2003)'s approach. In the former, the utterance is left underspecified by the grammar, and we must assume separately specified pragmatic routines (update rules?) to fill it in; in the latter, this underspecification is replaced by anaphora essentially unaccompanied by information about possible antecedents, which must be identified by pragmatic inference. In our approach, not only is the method of content specification fully defined by the grammar as a program, the source of the antecedents (particular state variables) is also made clear.

Note the similarity between this program and that introduced by information structure in AVM (12). Both programs express a constraint on the current maximal QUD, and therefore restrict their utterances' use to suitable contexts.[5]

4.3 Setting Up State Conditions

Note that not only does the program for the fragment specify the state variables where antecedents must be found, the program for the previous utterance will have specified how the values of these state variables were updated. As already shown in (1) above, interrogatives introduce questions to QUD – this will automatically provide a suitable head value of QUD for an elliptical answer which follows it. In fact, the program for wh-interrogatives also pushes a salient utterance (the wh-word corresponding to the question's abstracted parameter) onto the UTT stack, thus providing a state which will fulfill both the requirements of an elliptical fragment:

(16)
$$\begin{bmatrix} \textit{interrogative} & \\ \text{CONT} & \boxed{1}\left[\lambda\boxed{3}.p\right] \\ \text{C-PROG} & A; \texttt{QUD}.\textbf{push}(\boxed{1}); \texttt{UTT}.\textbf{push}(\boxed{2}) \\ \text{HEAD-DTR} \mid \text{C-PROG} & A \\ \text{CONSTITS} & \left\{\ldots\boxed{2}\left[\text{CONT} \quad \boxed{3}\right]\ldots\right\} \end{bmatrix}$$

Similarly, declarative utterances (as we have already seen in (2) above) introduce $whether(p)$ QUDs; indefinites also introduce programs to push themselves onto UTT for later resolution of sluices.

4.4 Ordering Sub-Programs

The specification of AVM (15) is designed to ensure that sub-programs are executed in a certain order: firstly, checks on state variables, before any utterance programs have had any effect; secondly, the sub-programs projected by individual phrases (and inherited by the sentence from its daughters); and thirdly the top-level effects of the utterance – updating QUD, UTT and LM. The ordering of the daughter sub-programs themselves will also be important to account for e.g. intrasentential anaphora and presupposition projection. Anaphoric definites and pronouns must be able to identify variables introduced by preceding indefinites as their referents, so we must ensure that the indefinite programs which introduce them are executed before the definite programs which attempt to find them. In English at least, this requires sub-programs to be put together in linear order, and this is simply expressed:

(17)
$$\begin{bmatrix} \text{C-PROG} & A; \ldots; B \\ \text{DTRS} & \left\langle\left[\text{C-PROG} \quad A\right], \ldots, \left[\text{C-PROG} \quad B\right]\right\rangle \end{bmatrix}$$

5 Summary

Update effects which are specified entirely by, and are inseparable from, utterances themselves can be represented as part of their grammatically assigned content, even when this content is contextually dependent. It is only when the context determines the form of these effects (the type of pro-

set the latest-move LM variable to the value of the utterance's content, a move – see (Purver and Fernández, 2003) for details.

[5] Of course, the same *could* be said for answers, if (as discussed and rejected above) they were to be represented as testing for a suitable QUD and popping it from the QUD stack.

gram which represents them), as with answers, that we need these effects to be determined by pragmatic processes. This is, of course, not to deny that these pragmatic processes govern dialogue to a large extent: merely to say that the dividing line between semantics and pragmatics can be drawn in a different place. This approach is currently being implemented in a HPSG grammar and a prototype IS-based dialogue system.

6 Acknowledgements

We would like to thank two anonymous Catalog reviewers for several helpful comments that have significantly contributed to the final version of this paper. The authors are supported by ESRC grants RES-000-23-0065 and RES-000-22-0355 respectively.

References

James Allen and Mark Core. 1997. Draft of DAMSL: Dialog act markup in several layers.

Nicholas Asher and Alex Lascarides. 2003. *Logics of Conversation*. Cambridge University Press.

Nicholas Asher. 1998. Varieties of discourse structure in dialogue. In J. Hulstijn and A. Nijholt, editors, *Proceedings of the 2nd Workshop on Formal Semantics and Pragmatics of Dialogue (Twendial)*, Enschede, May.

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 and Ulle Endriss. ms. Abstract models for dialogue protocols. Under review.

Raquel Fernández, Jonathan Ginzburg, Howard Gregory, and Shalom Lappin. 2004. SHARDS: Fragment resolution in dialogue. In H. Bunt and R. Muskens, editors, *Computing Meaning*, volume 3. Kluwer Academic Publishers. To appear.

Raquel Fernández. 2003. 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, Howard Gregory, and Shalom Lappin. 2001a. SHARDS: Fragment resolution in dialogue. In H. Bunt, I. van der Sluis, and E. Thijsse, editors, *Proceedings of the 4th International Workshop on Computational Semantics (IWCS-4)*, pages 156–172. ITK, Tilburg University, Tilburg.

Jonathan Ginzburg, Ivan Sag, and Matthew Purver. 2001b. Integrating conversational move types in the grammar of conversation. In P. Kühnlein, H. Rieser, and H. Zeevat, editors, *Proceedings of the 5th Workshop on Formal Semantics and Pragmatics of Dialogue (BI-DIALOG)*, 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. forthcoming. *A Semantics for Interaction in Dialogue*. CSLI Publications. Draft chapters available from: `http://www.dcs.kcl.ac.uk/staff/ginzburg`.

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

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*.

Bernd Ludwig. 2001. Dialogue understanding in dynamic domains. In P. Kühnlein, H. Rieser, and H. Zeevat, editors, *Proceedings of the 5th Workshop on Formal Semantics and Pragmatics of Dialogue*, pages 287–297. BI-DIALOG.

Matthew Purver and Raquel Fernández. 2003. Utterances as update instructions. In *Proceedings of the 7th Workshop on the Semantics and Pragmatics of Dialogue (DiaBruck)*, pages 115–122, Saarbrücken, September.

David Schlangen. 2003. *A Coherence-Based Approach to the Interpretation of Non-Sentential Utterances in Dialogue*. Ph.D. thesis, University of Edinburgh.

David Traum. 2003. Semantics and pragmatics of questions and answers for dialogue agents. In *Proceedings of the International Workshop on Computational Semantics*, pages 380–394, January.