DIUM – An Incremental Dialogue Manager That Can Produce Self-Corrections

Okko Buß
University of Potsdam
Germany
okko@ling.uni-potsdam.de

David Schlangen
Bielefeld University
Germany
david.schlangen@uni-bielefeld.de

Abstract

Incremental processing offers the potential for dialogue systems to produce more natural, spontaneous conversational behaviour. This processing strategy comes at a price, though, which is that processing results may have to be revised when more input becomes available. We distinguish between two possible consequences of such revisions: a) If no observable system reaction has been produced yet, revision is just a matter of properly keeping internal state, and can be handled along the lines of the IU model of (Schlangen and Skantze, 2009). b) If however an observable reaction has been produced, revocation itself becomes a dialogue move, and as such must be handled by the dialogue manager. In this paper, we describe a dialogue manager that is capable of doing so, and provide a first discussion of how to handle such self-corrections when producing output. This dialogue manager makes a connection between utterance-level incrementality and dialogue-level incrementality by using concepts from the IU model also internally; we discuss some of the implications of this approach.

1 Introduction

As much recent work has shown, incremental (or *online*) processing of user input or generation of system output helps spoken dialogue systems to produce behaviour that is perceived as more natural than and preferable to that produced by systems that are bound by a turn-based processing mode (Aist et al., 2006; Skantze and Schlangen, 2009; Buß et al., 2010; Skantze and Hjalmarsson, 2010).

However, incremental processing adds another dimension of uncertainty to the dialogue management task, which is that hypotheses can be *unstable* and get revised in the light of later information. This has been studied mostly in the context of speech recognition, where word hypotheses may change as more of the utterance is heard (see e.g. (Baumann et al., 2009; Selfridge et al., 2011) — for example, a hypothesis of "four" may turn into one of "fourty" — but it can in principle occur in all modules that are made to work with incomplete input (see e.g. (DeVault et al., 2009; Atterer and Schlangen, 2009; Schlangen et al., 2009; DeVault et al., 2011) for Natural Language Understanding, (Skantze and Hjalmarsson, 2010) for Natural Language Generation).

In their abstract model of incremental processing (henceforth, the *IU model*, where *IU* stands for *Incremental Unit*), Schlangen & Skantze (2009, 2011) describe in general terms methods for dealing with such revisions,[1] and note that additional work may be needed if hypotheses are revoked on which observable system behaviour has already been based. Previous work on dialogue management for incremental systems—to our knowledge, this comprises only (Buß and Schlangen, 2010; Buß et al., 2010)– does not yet carry out this additional work. In this paper, we aim to rectify this, and provide a dialogue management model that can deal in a principled way with this situation.

The rest of this paper is structured as follows: in Section 2 we describe in more detail the problem mentioned above and list possible strategies for deal-

[1] See (Schlangen et al., 2010) for concrete instantiations of this model in middlewares for dialogue systems.

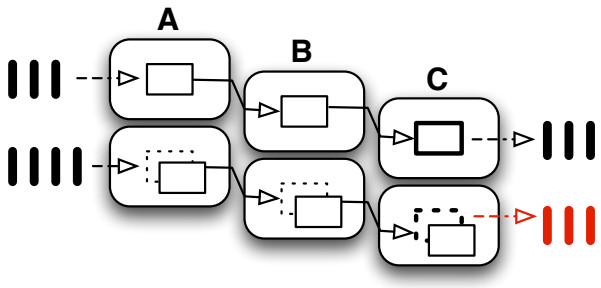

Figure 1: Schematic illustration of the problem (as described in text below)

ing with it. In Section 3 we develop our variant of one of those and illustrate how it offers a principled solution. We briefly describe our test implementation in Section 4. In Section 5 we discuss some further directions beyond addressing this particular problem are for our approach. We finish with a conclusion and outlook in Section 6.

2 The Problem, and Possible Solutions

Figure 1 gives a schematic illustration of the problem that we set out to tackle in this paper. In the beginning, a module **A** (which could be a module that processes user input or it could be a module that realises system intentions in an incremental fashion) decides that is can form an hypothesis about its input (the three bars on the left in the illustration), and passes this hypothesis on to later modules **B** and **C**. These in turn produce hypotheses, which they pass on, ultimately producing observable system behaviour (the three bars on the right in the illustration).

The fact that output has been produced now lends a special status to the hypothesis on which it was based; in the terms of the IU model (Schlangen and Skantze, 2011), this hypothesis is now *committed*. Unfortunately, however, while this further processing has been going on, additional input to **A** has become available which leads this module to substitute a new hypothesis for the old one. Now not only that new hypothesis has to be communicated to the later modules, but also the fact that the old hypothesis is not deemed viable anymore. If only internal state is concerned, as is the case with module **B**, that poses no problem: the hypothesis that was based on the now defunct input hypothesis is revoked as well, and the new input hypothesis is used to generate a new one. This, however, is not possible for modules that ultimately realise the system behaviour (here, module **C**), since their output may have been observed and thus become public knowledge already.

What is a dialogue system to do in such a case? In the following, we discuss several possible strategies.

2.1 Solution 1: Reducing Revisions

The first strategy for tackling the problem that one might think of is to try to attack it at its root, the instability of hypotheses: if revisions would never (or only very rarely) occur, situations in which system behaviour has to be 'taken back' would not (or rarely) occur.

Previous work has shown that hypothesis instability shows itself typically only with respect to the most recent input (Baumann et al., 2009). If sending out hypotheses is delayed until more potentially relevant material has been seen—i.e., some right context is allowed—some of this instability can be reduced. However, as shown in that paper, if done in such a way that all or even only most of the revisions are removed, a rather long delay (roughly 800ms) is needed, which would reduce responsiveness and hence the potential advantages of incremental processing dramatically. (For comparison, silence thresholds used for end-pointing in non-incremental systems are commonly set around 700ms, thus this solution would actually mean performing more poorly on end-pointing than non-incremental systems.) More importantly, this approach does not offer a principled way of dealing with the problem, as there is no theoretical limit on how much right context might be needed to disambiguate earlier hypotheses (cf. the well-known garden path sentences in parsing).

2.2 Solution 2: Ignoring the Problem

The second possible strategy addresses the other end of the processing chain, as it were, by simply treating output as revokable. The idea would be to let all modules change their internal state if revokes are requested. In our example, this would mean that module **C** removes its internal hypothesis which has turned out to be based on false assumptions, processes the revised input, and possibly produces the appropriate behaviour without further comment.

This is the strategy followed in the system described in (Buß and Schlangen, 2010), which pro-

duced mostly non-verbal signals such as highlighting areas on a computer screen. While 'revoking' such behaviour is easy to do and less noticeable than, for example, the system stepping back from a decision to interrupt the user mid-utterance, it still is a publicly observable action, and as such in danger of being interpreted (and possibly even overtly addressed) by the observant. If the system itself, having returned to an earlier state, has no record of having performed such an "undo" of actions, inconsistencies may arise later on when the user explicitly refers to the mistakenly realised action.

2.3 Solution 3: Explicitly Representing and possibly Acknowledging the Situation

The discussion of the downsides of these two strategies suggests a third strategy, namely to let the system a) represent to itself that it is in a conflicting state, and b) decide whether to publicly address it (e.g., by reverting the effects of its previous action and apologizing for the 'mistake'. This is indeed the strategy that we will detail in the next section.

3 Information States as Graphs of IUs

We will now describe the main ideas behind *DIUM*, the IU-based Dialogue Manager that we have devised for use in incremental dialogue systems, and show how it handles the problem. For concreteness, we will use a typical travel-information domain for our examples, and contrast our approach with a 'classical' information state update (ISU) approach (Larsson, 2002) and our own previous attempt at formulating an incremental ISU variant, *iQUD* (Buß et al., 2010).

Figure 2 shows prototypical information states (ISs) according to these approaches. From the left: the IS labeled *QUD* is the aforementioned 'classical' IS (example simplified from (Larsson, 2002)), which keeps record of a current 'issue', a 'plan' and latest user and system moves. Next, *iQUD*, adapted from (Buß et al., 2010), is an incrementalised version thereof which uses a compact representation of the plan (the first column). It also contains in the second column for each plan item output instructions such as relevant non-linguistic actions (`RNLA`) that are to be triggered once the appropriate information is collected; this is used to showcase how the dialogue manager (DM) can be made to react as soon as possible. The structure also records the grounding status of each relevant bit of information, in the third column.

Lastly, *DIUM* is the IS introduced here, consisting of a network of IUs. We will now discuss this formalisation in some more detail.

3.1 The DIUM Information State

The *DIUM* information state in Figure 2 represents the DM's initial state in our example travel information domain. Like the discourse plan in the other two approaches, it has to be hand-crafted by the system designer for the domain at hand.

The nodes in the graph can best be thought of as *discourse units* (roughly as in (Traum, 1994)), structuring which 'chunks' of information the user is expected to provide during the dialogue, prompted or unprompted. These units are *incremental units* (IUs) in the terms of the IU-model mentioned above, and will be called *DiscourseIU*s henceforth; but note that these units incrementally build up the dialogue, where the IUs more typically dealt with in previous work using the IU model are those building up an utterance. These nodes take on the role of the *findout* items on the *QUD* or the slots of the *iQUD* in the other models mentioned above.

In this example, *DiscourseIU*s are either terminal *slots* or inner *topic* nodes on a tree structure. Again following the IU-model, they are connected with one another by two types of same-level links (SLL, i. e. links that connect IUs from the same module; the other type being grounded-in links, GRIN, which link units across levels of processing.) The first are called *seq*, which indicate a sequential relationship (for example to encode order preference or expectations) and are depicted with dotted arrows in the figure. The second are *dom*, which indicate a dominance/hierarchical relationship, depicted with solid lines.[2] For example, the `topic:date` unit *dom*inates slots `day` and `month`, which in turn are connected by a *seq* link. The semantics of these links here is essentially: 'to collect a date, learn about a day and a month' and 'preferably, learn about the day before the month', respectively.[3]

[2] This taxonomy of relations is inspired by the dominance and satisfaction-precedence relations of (Grosz and Sidner, 1986).

[3] Arrangement of DM states or plans in tree-like graphical structures is of course rather popular and used in various ap-

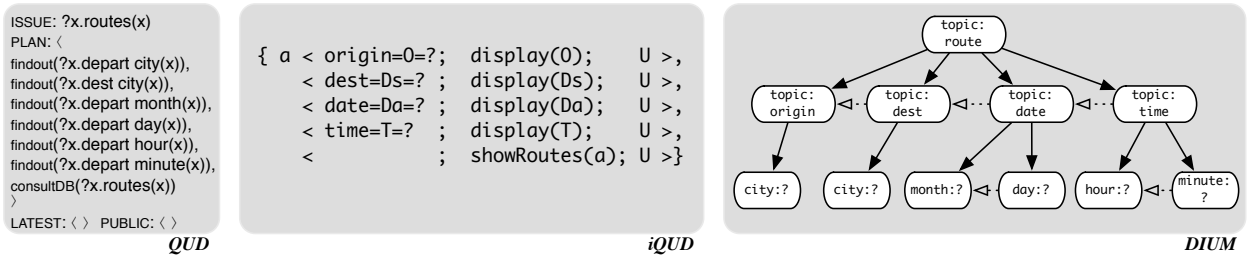

Figure 2: Three types of information states for modelling a travel timetable domain

In this sense, *DiscourseIU*s represent at the same time items that the system can ask for as well as underspecified *projections* of expected, future input. This lets them serve both main goals of any DM, which are to provide context for new input and to initiate the production of relevant system behaviour. The former is achieved by checking whether input IUs can ground *DiscourseIU*s, fulfilling expectations about how the dialogue will proceed. The latter similarly works by linking *DiscourseIU*s and other IUs, but this time by creating appropriate output IUs which are grounded in specific *DiscourseIU*s. How these processes work in detail will be discussed next.

3.1.1 Integrating Input

Initially, *DiscourseIU*s are not connected to any input (much like the *QUD* plan items are unanswered or *iQUD* slots are unfilled). Incoming incremental input will trigger update rules whose effect includes the creation of new grounded-in links between relevant *DiscourseIU*s and input. Figure 3 provides an example showing a subset of the *DIUM* network. Here, the *DiscourseIU*s have become grounded in input-IUs representing the spoken user input "from hamburg", where the *WordIU*s represent word hypotheses and the *SemIU*s are representations of the content of those words.

Note that the *WordIU*s arrive incrementally so that *DiscourseIU* `topic:origin` may become grounded in "from" possibly before "Hamburg" was even spoken.

What the illustration does not show is *how* the update rules arrive at identifying relevant *DiscourseIU-SemIU* pairs. For this, the rules encode a search

proaches, see e. g. (Xu and Rudnicky, 2000; Stede and Schlangen, 2004; Ljunglöf, 2009; Bangalore and Stent, 2009). As we will discuss presently, it is the easy integration into the general IU model that makes this form of representation attractive here.

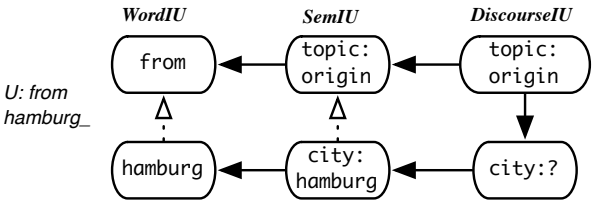

Figure 3: Integrating "from hamburg" incrementally.

over the $DIUM$ information state, starting from a 'focus' *DiscourseIU*, i. e. the most recent one to link with input. When new input arrives, a narrow search space is traversed, including only *DiscourseIU*s that are *dom*inated by this focus node in the tree structure. If a single matching pair is found here (a 'match' meaning that the two IUs unified, such as `city:hamburg` and `city:?`), the GRIN link is created as shown. If more than one pair was found, the system asks the user to clarify between them (as discussed below). If no matching pairs are found, the search is extended to cover the *entire* information state (not just the subgraph 'below' the focus node). If this second iteration still yields no matching pair(s), the system requests more information from the user. In this way recent input (here, "from") determines the appropriate context for the current input (here, "Hamburg", which without this focus would be ambiguous, as there there are two `city` nodes in this dialogue model where it could fit). At the same time this mechanism allows users to over-answer ("from Hamburg on the third of may") or switch topic ("from, uhm hold on, on the third of may") within a single utterance.

3.1.2 Producing Output

DM output is similarly produced by adding GRIN links, this time between *DiscourseIUs* and newly

created output IUs. At each DM step that integrates input, further update rules specify what kind of output may be appropriate. Figure 4 illustrates this. Here, after grounding the `topic:origin` *DiscourseIU* in input, a *DialogueActIU* representing an intention to enquire about the departure city is immediately added; it will be the role of a later component (which we call the action manager, AM) to decide on how (or even whether) to realise this intention.

It is important to note that adding such a *DialogueActIU* only signals an intention to act, a temporary projection. (This is in contrast to the `RNLA` instructions of the *iQUD* approach.) Whether such a kind of intention gets turned into action right away (or at all) depends on the overall system setup as well as what information the user provides next. In a multimodal system it might be turned into a "relevant non-linguistic action" `RNLA` immediately, such as an on-screen signal that a departure city is expected now. In the speech modality, it might only be realised if there is no overlap with user speech; in the example here, the user continues to talk and hence no opportunity is given for the system to produce a spoken utterance (nor is there need to do so). If however a hesitation had been detected, the system could have realised some form of request for more information; depending on further rules, perhaps gently as a continuer ("mhm?") or more explicitly ("from where?").

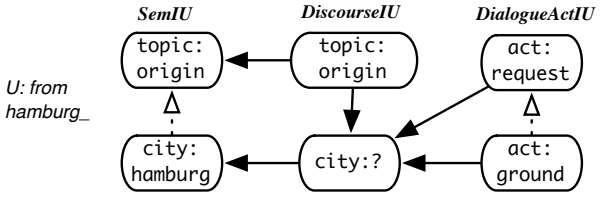

Figure 4: Producing incremental output.

The example shown here however continues with the `city` *DiscourseIU* being grounded in "Hamburg", and again a *DialogueActIU* is generated, this time to achieve grounding (mutual knowledge) of the system's understanding. Again, in a multi-modal system it can be left to a later module to decide on the best way to realise this dialogue act.

Note also that at this point the previous *DialogueActIU* has been made irrelevant by new input (`act:request` is answered with "hamburg"). If it hasn't lead to actual output, it no longer needs to be realised to the user and can be downdated by being revoked (in other words, the projected intention to act is retracted). Though this is not indicated in the illustration, an update rule to take care of this is triggered when grounding a *DiscourseIU* in new input. The rule then revokes any unrealised *DialogueActIU*s grounded in that *DiscourseIU*. For the DM this may be just good housekeeping. However for the system's behaviour this is critical to avoid that the AM overproduces output based on projections that were over-generated by the DM.

3.2 How DIUM Addresses the Problem

So far we've looked at how the *DIUM* IU network information state can be incrementally updated during two common DM activities: contextualising input and producing relevant output. Now we'll look at how the IU graph approach can be leveraged to implement the solution strategy identified in Section 2.3.

Let's revisit the problem and the solution for a moment. Input that triggers a decision to produce output is revoked. If the output was already made public by that point, it cannot simply also be revoked, and the DM must resolve the clash explicitly. For this, it needs to be able to do three things: (1) to *compute a new state* reflecting that input was revoked and (2) to *check its own output* to determine whether projected dialogue acts have indeed already been realised into observable output. In cases where such a clash arises, it needs to then (3) *initiate explicit repair*.

3.2.1 Handling Revokes and Checking Output

Requirements (1) and (2) turn out to be already addressed by basic mechanisms of the IU model. In that model, IU graphs can be modified using REVOKE edits, which in turn are communicated as 'edit messages' among modules in an incremental spoken dialogue system. In *DIUM*, we simply use reception of such an edit message as an additional type of update rule trigger. This trigger can then be used in update rules that compute the appropriate new IS. In Figure 5 this happens at step **2**. Here, revoked input `city:hamburg` triggers an update that causes the grounded-in links from any *DiscourseIU*s to the revoked input to be removed.

A similarly simple solution exists for monitoring DM output. The IU formalism specifies that

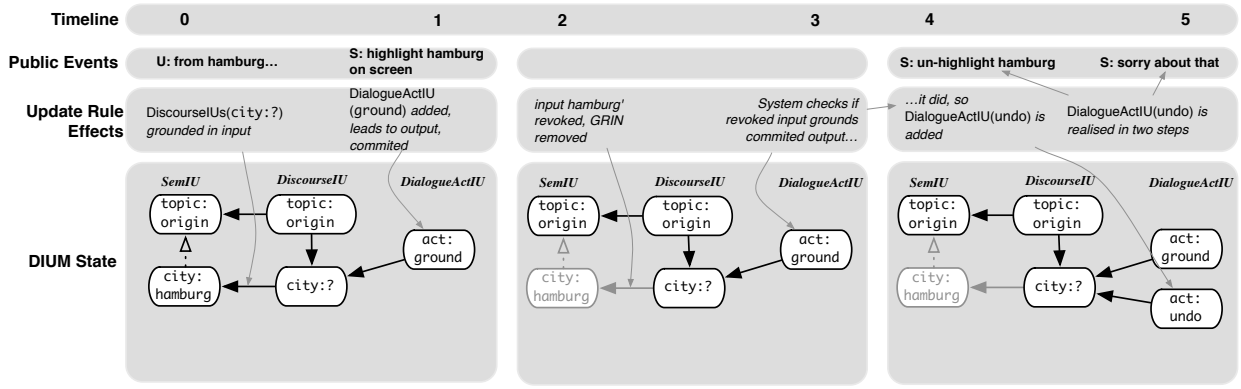

Figure 5: Trace of example: *revoke* clashes with *commit*, leading to an UNDO, which is realised as an apology.

the commit-status of IUs is recorded. We assume that if a *DialogueActIU* has been realised, it will be committed by the realiser, as shown in step **1** for `act:ground`. This status of the IU in turn is accessible to the DM, and can form a trigger in an update rule. After input to the DM module is revoked, the update rules check whether there are output IUs, and if so, what their commit status is. If a clash is detected (an IU needs to be revoked, but is committed), appropriate steps can be taken; this is what happens at step **3** in Figure 5.

3.2.2 Undo Dialogue Acts

The reaction to the detected clash takes the form of adding to the output of the DM a dialogue act (intention) of a special type, UNDO. This is shown in step **4** in the example. Here, this act is in turn realised in two steps: Visual output is updated immediately (the highlighting is removed), whereas at a later moment (as the system decided that it held the turn, not indicated in detail here), additionally an apology is issued (step **5**).

This is only one strategy for realising such UNDOs, though. Determining the best strategy for doing so is an empirical question that we have not turned to yet and leave for future work; here we wanted to lay the groundwork needed to explore this question. Strategies to test in an implemented system might come from studies on human repair strategies. For example, speakers tend to self-repair as soon as possible (Levelt, 1983) and different types of repair are associated with different costs, determined by modality as well as who initiates it (Clark and Brennan, 1991). In dialogue systems, (Skantze and Hjalmarsson, 2010) also offer some ideas for how a system might incrementally produce overt and covert self-repairs (however of spoken output only).

4 Implementation

We have implemented the DIUM approach in a small but fully functional example system, using the InproTK framework (Schlangen et al., 2010). Using DIUM and otherwise comparable components, the implemented system achieves the same coverage of phenomena relevant to incremental processing as the *iQUD* system, namely being able to react to user hesitations by producing continuer feedback utterances, and showing RNLAs. Additionally, *DIUM* is able to handle revoke-commit-clashes in the way described above; this adds occasional self-corrections of the type described above to the conversational flow.

While the initial domain in which we tested *DIUM* was the travel domain described here, we have also realised the puzzle domain described in (Buß and Schlangen, 2010) in this new approach. It proved to be straightforward to encode the expected dialogue shapes in the *DiscourseIU* graphs used by *DIUM*. Moreover, only very few changes to the *DIUM*-rule set were necessary; in combination, this shows that a certain domain-independence is given by the DIUM approach.

5 Further Directions

In this paper, we have focussed on how the approach to dialogue management followed in *DIUM* helps tackle the revoke-commit-problem. We are currently exploring further possible advantages that the graph-

based representation format affords, of which we will discuss a few now.

5.1 Dialogue History & Grounding

For one, *DIUM* does not require its own housekeeping for dialogue history. Since IUs encode start and end times, the information state *is*, in fact, a very finely granular dialogue history. Knowledge of user and system actions (and their timing) becomes a matter of querying the network. A special data structure keeping track of recent moves is thus redundant.

We are also currently exploring to what extent properties of the IU network can be used to account for 'grounding' of input and output (in the sense of (Clark and Brennan, 1991; Clark, 1996)). Where spoken dialogue systems usually use symbolic representations of different grounding 'statuses' attached to input and output, e. g. (Skantze, 2007; Buß et al., 2010) or, alternatively, keep input and output representations in a special location within the IS to represent this status, e. g. (Larsson, 2002), in the DIUM approach grounding status may be reduced to configurations of an IU network. For example (using an informal taxonomy), *DIUM*'s 'private' beliefs about user input can be thought of as *DiscourseIU*s grounded in *SemIU*s, but with no observable reaction realised by *DialogueActIU*s, and beliefs that are made public are *DiscourseIU*s that ground committed *DialogueActIU*s while being grounded in one or more committed *SemIU*s. How far this reduction can be carried will be explored in future work.

5.2 DIUM & Discourse Theories

We also see exciting possibilities for forging connections to dialogue and discourse theories such as RST (Mann and Thompson, 1987) and SDRT (Asher and Lascarides, 2003), where discourse structure is represented through relations between smaller units, and where discourse meaning is jointly constituted by the contributions of the units and those of the relations.[4] While in the current implementations we have used pre-authored graphs of expected contributions, we are exploring a more dynamic construction process that combines a notion of *coherence* (as in SDRT) for use in integration of new material, with a notion of projection of next system moves that must be coherent and move the dialogue further towards a goal.

[4]It is an interesting historical coincidence that dialogue management has benn more influenced by theories (such as KOS (Ginzburg, 1995; Ginzburg, forth)) that in contrast chose a feature structure- or record-based approach rather than a graph-based one.

6 Conclusions

In this paper, we have introduced an approach to the representation of dialogue knowledge for dialogue management and operations on such representations, that lends itself to being used in incrementally working dialogue systems. In particular, the approach affords a straightfoward handling of what we have called the revoke-commit problem, where a system for internal reasons decides to 'take back' some of its actions. We have introduced this approach with examples from a simple information-seeking domain, for which we have realised an implementation that can produce certain naturalistic interactive phenomena (backchannels, continuers). In future work, we will explore the additional directions mentioned in the previous section, such as investigating the potential of the approach for handling grounding phenomena in a fine-grained way, and for connecting the underlying dialogue representations with theoretically better motivated approaches.

References

Gregory Aist, James Allen, Ellen Campana, Lucian Galescu, Carlos A. G. Gallo, Scott C. Stoness, Mary Swift, and Michael Tanenhaus. 2006. Software Architectures for Incremental Understanding of Human Speech. In *Proceedings of the 6th Interspeech*.

Nicholas Asher and Alex Lascarides. 2003. *Logics of Conversation. Studies in Natural Language Processing*. Cambridge University Press.

Michaela Atterer and David Schlangen. 2009. RUBISC-a Robust Unification-Based Incremental Semantic Chunker. In *EACL 2009 Workshop on Semantic Representation of Spoken Language*, Athens, Greece.

Srinivas Bangalore and Amanda J Stent. 2009. Incremental Parsing Models for Dialog Task Structure. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 94–102, Athens, Greece.

Timo Baumann, Michaela Atterer, and David Schlangen. 2009. Assessing and Improving the Performance of Speech Recognition for Incremental Systems. In *Proceedings of Human Language Technologies: The 2009*

Annual Conference of the North American Chapter of the Association for Computational Linguistics, Boulder, Colorado.

Okko Buß and David Schlangen. 2010. Modelling Sub-Utterance Phenomena in Spoken Dialogue Systems. In *Proceedings of SemDial*, Poznań, Poland.

Okko Buß, Timo Baumann, and David Schlangen. 2010. Collaborating on Utterances with a Spoken Dialogue System Using an ISU-based Approach to Incremental Dialogue Management. In *Proceedings of SigDial 2010*, Tokyo, Japan.

Herbert H. Clark and Susan E. Brennan. 1991. Grounding in communication. In L. B. Resnick, J. Levine, and S. D. Behrend, editors, *Perspectives on Socially Shared Cognition*, pages 127–149. American Psychological Association Books, Washington D.C., USA.

Herbert H. Clark. 1996. *Using Language*. Cambridge University Press, Cambridge.

David DeVault, Kenji Sagae, and David Traum. 2009. Can I Finish? Learning When to Respond to Incremental Interpretation Results in Interactive Dialogue. In *Proceedings of the SIGdial 2009*, London, UK.

David DeVault, Kenji Sagae, and David Traum. 2011. Incremental interpretation and prediction of utterance meaning for interactive dialogue. *Dialogue & Discourse*, 1. Special Issue on Incremental Processing in Dialogue.

Jonathan Ginzburg. 1995. Resolving questions I. *Linguistics and Philosophy*, 18:459–527.

Jonathan Ginzburg. forth. *The interactive Stance: Meaning for Conversation*. CSLI Publications.

Barbara J. Grosz and Candace L. Sidner. 1986. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(3):175–204.

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

Willem J.M. Levelt. 1983. Monitoring and self-repair in speech. *Cognition*, 14:41–104.

Peter Ljunglöf. 2009. Dialogue Management as Interactive Tree Building. In *Proceedings of DiaHolmia, 13th Workshop on the Semantics and Pragmatics of Dialogue*, Stockholm, Sweden.

William C. Mann and Sandra A. Thompson. 1987. Rhetorical structure theory: A theory of text organization. In Livia Polanyi, editor, *The Structure of Discourse*. Ablex Publishing Corporation, Norwood, N.J.

David Schlangen and Gabriel Skantze. 2009. A General, Abstract Model of Incremental Dialogue Processing. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, Athens, Greece.

David Schlangen and Gabriel Skantze. 2011. A general, abstract model of incremental dialogue processing. *Dialogue & Discourse*, 2(1). Special Issue on Incremental Processing in Dialogue.

David Schlangen, Timo Baumann, and Michaela Atterer. 2009. Incremental reference resolution: The task, metrics for evaluation, and a bayesian filtering model that is sensitive to disfluencies. In *Proceedings of SigDial 2009*, London, UK.

David Schlangen, Timo Baumann, Hendrik Buschmeier, Okko Buß, Stefan Kopp, Gabriel Skantze, and Ramin Yaghoubzadeh. 2010. Middleware for Incremental Processing in Conversational Agents. In *Proceedings of SigDial 2010*, Tokyo, Japan, September.

Ethan Selfridge, Iker Arizmendi, Peter Heeman, and Jason Williams. 2011. Stability and accuracy in incremental speech recognition. In *Proceedings of the SIGDIAL 2011 Conference*, pages 110–119, Portland, Oregon, June. Association for Computational Linguistics.

Gabriel Skantze and Anna Hjalmarsson. 2010. Towards Incremental Speech Generation in Dialogue Systems. In *Proceedings of SigDial 2010*, Tokyo, Japan.

Gabriel Skantze and David Schlangen. 2009. Incremental dialogue processing in a micro-domain. In *Proceedings of EACL 2009*.

Gabriel Skantze. 2007. *Error Handling in Spoken Dialogue Systems*. Ph.D. thesis, KTH.

Manfred Stede and David Schlangen. 2004. Information-Seeking Chat: Dialogue Management by Topic Structure. In *Proceedings of SemDial 2004*, Barcelona, Spain, July.

David R. Traum. 1994. *A Computational Theory of Grounding in Natural Language Conversation*. Ph.D. thesis, Computer Science, University of Rochester, Rochester, USA, December.

Wei Xu and Alexander I. Rudnicky. 2000. Task-based Dialog Management Using an Agenda. In *Proceedings of ANLP/NAACL 2000 Workshop on Conversational systems*, pages 42–47, Seattle, Washington.