Natural Language Generation and Discourse Context: Computing Distractor Sets from the Focus Stack

David DeVault
Department of Computer Science
Rutgers University
110 Frelinghuysen Road
Piscataway, NJ 08854
ddevault@cs.rutgers.edu
http://www.cs.rutgers.edu/∼ddevault

Charles Rich and **Candace L. Sidner**
Mitsubishi Electric Research Laboratories
201 Broadway
Cambridge, MA 02139
rich, sidner@merl.com
http://www.merl.com/projects/collagen

Abstract

In human-human conversation, people use linguistic expressions that are flexibly tailored to context as a matter of course, and they expect their conversational partners to do likewise. Towards our goal of helping computers achieve natural linguistic interactions with human users, we have developed an application-independent dialogue system integrating the COLLAGEN collaboration manager with the SPUD natural language generation algorithm. In this paper, we show how this integration supports investigation into strategies for distractor set based generation in dialogue. We discuss, as a case study, our experience in using the integrated system to generate contextually appropriate noun phrases in an email and scheduling application. While we have achieved natural, context-sensitive output in certain cases, our work has also revealed several issues facing systems wishing to exploit distractor sets to fine-tune their linguistic output. We discuss our approach to handling these issues and suggest areas where further research is required.

Introduction

While approaches to natural language generation that ignore context are straightforward to implement, they often produce unsatisfactory linguistic output. In generating noun phrases, for example, a system that ignores context must make a global choice about which properties and relations to include in its output. Such a system may awkwardly evoke properties or relations in contexts in which the referent would be perfectly clear without them, as in (2a), where the italicized material is intuitively unnecessary in a reply to (1). Or it may produce ambiguous output in contexts where some categorically excluded property or relation would disambiguate the intended referent, as in (2b), where the global suppression of last names prevents the system from distinguishing two different Johns. While much attention has been paid to generating noun phrases (and indeed we focus on generating noun phrases in the case study described in this paper), related issues arise elsewhere, for example in generating verb phrases headed by verbs with optional arguments. The decision about whether to reply to (1) as in (2c) or as in (2d)

(1) User: Add a meeting with John at 3pm Tuesday.

(2a) System: The meeting *with John at 3pm Tuesday* conflicts with the meeting with Mary at 3pm Tuesday.

(2b) System: The meeting with John at 3pm Tuesday conflicts with the meeting with John at 3pm Tuesday.

(2c) System: I've added the meeting to your calendar.

(2d) System: I've added the meeting.

ought to depend on whether the optional PP "to your calendar" helps distinguish the system's action in the context. For example, if the user is in the process of making additions both to her calendar and to someone else's calendar, the optional PP may be necessary to make clear whose calendar is being modified.

There is a clear trade-off between saying too much and saying too little in human-human conversation, with the result that natural human-computer conversation depends on the computer being able to discern how much it is appropriate to say in a specific context. The prevailing algorithms for generating noun phrases formalize this trade-off using the notion of *distractor* sets; see e.g. (Dale & Reiter 1995; Reiter & Dale 2000; Horacek 1997; Gardent 2002; Stone *et al.* 2003). Generally speaking, these algorithms search for a set of properties, and in some cases relations, holding for a noun phrase's intended referent, but not holding for any of the entities in its distractor set. Research in this area typically *presupposes* the availability of distractor sets, and focuses on techniques for handling the complexities that arise when searching over large domains of properties, relations, or potential referents. Though it has been assumed that distractor sets encapsulate important gradations in salience that determine potential ambiguities in context, it has remained unclear precisely what principles ought to determine the exact contents of distractor sets (Jordan 1999; Krahmer & Theune 2002). One common suggestion (see,

e.g. (Horacek 1997; Reiter & Dale 2000)), which we pursue in this paper, is that the rich hierarchical model of attention in a discourse context that was introduced in (Grosz & Sidner 1986) may capture the relevant information for constructing distractor sets.

In the next section, we describe a new dialogue system we have developed integrating COLLAGEN, an architecture for building collaborative agents based on a theory of discourse structure which began with (Grosz & Sidner 1986), with SPUD, a sophisticated, distractor set based natural language generation algorithm. Then we discuss, as a case study, our experience in using the integrated system to generate contextually appropriate noun phrases in an email and scheduling application. In the final section, we discuss our approach to handling several issues we encountered in constructing appropriate distractor sets, and suggest areas where further research is required.

System description

COLLAGEN

COLLAGEN (for COLLaborative AGENt) (Rich, Sidner, & Lesh 2001) is a practical architecture for building dialogue systems and an implementation of a theory of collaborative discourse developed over a number of years (Grosz & Sidner 1986; 1990; Grosz & Kraus 1996; Lochbaum 1998).

In order to model the intentional structure in a human-computer conversation, COLLAGEN applications include a task model consisting of top-level goals for the interaction and a library of recipes which decompose these goals into subgoals, recursively. Each discourse purpose is represented as a shared goal to be achieved collaboratively by the human user and the COLLAGEN agent. Goals can be achieved by utterances, by direct action, or by carrying out recipes; in the latter case, hierarchical discourse structure emerges. COLLAGEN uses recipe-based plan-recognition to interpret user utterances as contributions to open goals. System responses are generated using the same recipe library and some simple planning heuristics.

Importantly for generation, COLLAGEN enters each entity that figures in a user or system utterance into a hierarchical analysis of the *attentional structure* of the discourse, implemented as a stack of focus spaces. The placement of an entity in the focus stack is determined by how the utterance containing that entity fits into the current overall intentional structure of the discourse. The introduction of new discourse goals results in new focus spaces being pushed onto the stack, and the completion of discourse goals results in old focus spaces being popped, as in (Grosz & Sidner 1986). Entities that have been featured in utterances serving more immediate open goals are present in focus spaces near the top of the stack, and are thus, according to the theory, comparatively more salient. The end result is that COLLAGEN applications have access to a fine-grained model of the comparative saliences of entities at each point in a conversation; i.e., they have access to precisely what is needed to drive models of generation presupposing distractor sets. Some examples of focus stacks generated by COLLAGEN are illustrated in Figure 3 and discussed below.

Previous research using the COLLAGEN framework has been focused on appropriately modeling the intentional structure of discourse rather than on exploiting the focus stack for context-sensitive generation. In the past, COLLAGEN agents have thus employed a context-independent, template-based generation scheme.

SPUD

SPUD (for Sentence Planning Using Description) (Stone *et al.* 2003) is a general purpose, sophisticated natural language generation algorithm which we see as a promising approach to achieving much of the flexibility and context-sensitivity human users expect in natural human-computer dialogue. SPUD is general purpose in the sense that it is designed to generate complete utterances—it subsumes the generation of definite noun phrases as a special case instead of treating them as the central concern, as in work such as (Dale & Reiter 1995). For example, while SPUD could be used to choose whether to include the italicized material in generating the first noun phrase of (2a), it could also be used to decide whether to include the optional PP in generating a complete sentential utterance such as (2c) or (2d). It is sophisticated in that it works in a rich space of candidate utterances in response to a rich problem specification. We provide only a high-level summary of the algorithm here, and refer the interested reader to (Stone *et al.* 2003) for details.

A SPUD generation problem consists of:

(3a) the syntactic category of the root node for the generated utterance,
(3b) a set of contributions to the discourse state to be made by the generated utterance,
(3c) the intended interpretation of any discourse anaphors introduced by the root node,
(3d) any syntactic feature values imposed from above on the root node,
(3e) a lexicalized tree adjoining grammar (LTAG) in which lexical entries are decorated with semantic constraints,
(3f) a knowledge interface that can answer semantic queries, and
(3g) a mechanism for retrieving context-sensitive distractor sets for arbitrary individuals.

Essentially, a SPUD problem dictates what kind of utterance is required (3a), what the utterance should accomplish (3b), whether the root node must refer to any particular individual (3c),[1] whether any syntactic constraints such as case or number bear on the utterance (3d), what linguistic resources are available (3e), what the relevant facts are (3f), and what the relative saliences of various individuals are (3g).

The linguistic resources of (3e) record the semantic effects of using individual lexical entries as sets of constraints defined over *discourse anaphor* variables. The knowledge

[1]For example, if SPUD is used to generate a definite noun phrase, the intended referent can be specified here. This is exactly what we do in the case study described later in this paper.

interface of (3f) is expected to provide exhaustive sets of solutions to these semantic constraints during generation. There are two types of constraints, and two associated types of knowledge interface queries.[2] *Presuppositions* record properties and relations that a speaker can presuppose as known by a hearer. For example the lexical entry for the noun *meeting* may have $\{meeting(X)\}$ as its presupposition, where X is a discourse anaphor corresponding to the referent of the noun phrase headed by the word *meeting*.[3] The knowledge interface solves presupposed constraints of the form $meeting(X)$ by providing interpretations for X as domain individuals that the hearer can be assumed to know are meetings. *Assertions* record the contributions to the discourse state made by particular lexical items. For example the lexical entry for the verb *sings* might assert $\{sings(X)\}$, where X is the subject of the verb phrase headed by the word *sings*. The knowledge interface solves assertion constraints of the form $sings(X)$ by providing interpretations for X such that X can be asserted to sing. Ordinarily a dialogue system will allow the assertion of some fact only if it is believed to be true, for example. Providing this knowledge interface in general constitutes the difficult problem of translating between the semantics of general linguistic words and specific domain predicates and individuals; see, e.g., (Stone 2003) for an approach consonant with the SPUD framework.

Given a problem specification as in (3a-g), SPUD construes generation as a search problem defined over a space of *communicative intents*. A communicative intent is a data structure recording all the information SPUD uses to gauge its progress, including a partially saturated syntactic tree structure, an intended interpretation for any discourse anaphors in the provisional utterance, the supported interpretations for the discourse anaphors, and which of the desired contributions the provisional utterance achieves. SPUD employs head-first greedy search, with search states expanded by syntactic operations that combine new lexical entries into the provisional utterance. Candidate next states are evaluated by a greedy search heuristic that includes preferences to achieve more desired contributions, resolve ambiguities, and saturate the syntactic tree structure, *inter alia*. If generation is successful, SPUD's product is a communicative intent containing a fully saturated syntactic tree structure and recording the intended (and, since generation was successful, distinguished) interpretations of all the discourse anaphors in the utterance.

An integrated dialogue system

The overall architecture of our integrated dialogue system is illustrated in Figure 1. The essential issue in exploiting SPUD in COLLAGEN dialogue systems is determining how to formulate appropriate SPUD generation problems, as in (3a-g), in particular dialogue situations. The appropriate values for (3a-d) vary according to whether SPUD is used to generate an NP or a complete sentence. For generating distinguishing noun phrases, (3a) is NP, (3b) is generally empty, (3c) is the intended referent, and (3d) depends on the context in which the NP is to be embedded. We intend to generate complete sentences with SPUD eventually, in which case (3a) is S, (3b) depends on the intention motivating the utterance, (3c) is empty unless sentences are treated as referential, and (3d) is probably empty.

Developing a SPUD grammar for (3e) is straightforward, if somewhat time intensive. In COLLAGEN applications, where a static mapping between linguistic and domain properties is possible, specifying a knowledge interface for (3f) is relatively straightforward. The only complexity is that information about whether properties and relations hold for various individuals tends to be distributed among a diffuse set of objects designed for the particular COLLAGEN application, so that providing a knowledge interface requires manual specification of the conversion between the constraint-based representation utilized for linguistic semantics and actual application object queries.

In our experience, the most substantive issue in achieving this integration lies in the construction of distractor sets for generation. While COLLAGEN associates discourse states with a focus stack (and therefore a partial order on object salience), this model has not been extensively explored for generation purposes, and how to exploit it to construct distractor sets that lead to good generation results is not well understood. As an initial strategy, we have adopted the simple strategy of searching down the focus stack, starting at the most salient focus space, accumulating all encountered individuals as distractors until we reach the first focus space containing the intended referent. Thus the distractor set for a target individual includes all individuals mentioned in utterances supporting goals at least as immediate as the goal that motivated the most recent mention of the target individual. The algorithm is given in Figure 2. Note that the algorithm identifies individuals not yet in COLLAGEN's focus stack as a special case. We return to the issue of first mentions of new individuals below.

A case study: an email and scheduling application

As a case study, we used our new integrated system to extend a preexisting COLLAGEN-based email and scheduling application (Gruen *et al.* 1999). Though we believe our system will eventually generate full sentential utterances, as a first step, we investigated the generation of referring noun phrases, which were then integrated into the output of COLLAGEN's template-based generation system to make full sentences.

The grammar we developed includes six simple lexical entries, *the*, *message*, *meeting*, *from*, *you*, and *it*. It also includes several quasi-lexical entries that allow us to generate references to individuals from several open ended classes, including dates, meeting topics, and email addresses, without actually developing a fully lexicalized grammar sufficient for covering these domains. We mapped presupposed

[2]Here we discuss presuppositions and assertions, but the full SPUD system additionally accommodates *pragmatic* constraints as well. See (Stone *et al.* 2003) for details.

[3]This example assumes the lexical entry can only figure in a definite noun phrase.

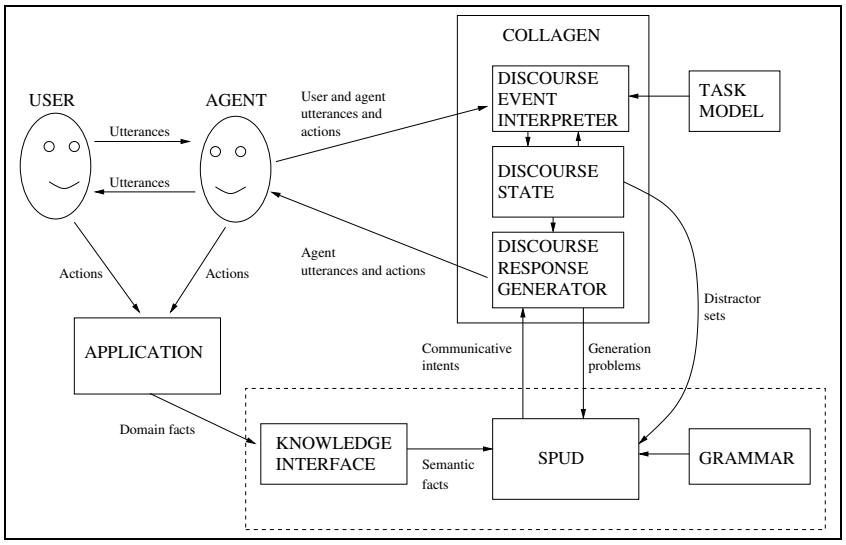

Figure 1: The architecture of our integrated system.

To compute a distractor set D_r for individual r:
Initialize D_r to $\emptyset$.
Search over focus spaces F, from most to least salient
Search over individuals x in F
If $x \neq r$ add x to D_r.
If $x \in F$ return D_r.
Fail (first mention).

Figure 2: A straightforward algorithm for deriving distractor sets from the current COLLAGEN attentional state.

constraint queries onto application object queries in a simple knowledge interface, and generated distractor sets according to the algorithm of Figure 2. We then posed SPUD distinguishing noun phrase generation problems, as described above.

We describe our results in the context of a typical interaction with the email and scheduling application. The interaction consists of 55 dialogue events, including utterances and actions by both the user and the agent. Figure 3 illustrates some of the results. For example, in utterance 31, our integrated system refers to `msg2` as "the message" rather than "the message from you" because `msg2` is the most salient individual in the current discourse state (and hence has no distractors). Indeed, SPUD would have generated "it" in this case, but we explicitly forbid a pronoun here since pronouns integrate poorly with the template "Please fill in the cc field of <SPUD-output>", which we used in integrating SPUD's output with COLLAGEN's broader template-based generation system. Note that if SPUD had been used to generate a referring noun phrase for `msg1` at this point, "the message from John" would have been generated, rather than "the message", because `msg1` has `msg2`, which is also a message, as a distractor. In utterance 52, our integrated system refers to `mtg1` as "it" instead of awkwardly reiterating the meeting date and topic as recently specified by the user in utterance 48. In fact, the intervening utterances, which have been left out of the figure, all concern `mtg1` as well; the verbose output of COLLAGEN's template-based generation system underscores the awkward redundancy that can occur when context does not inform generation choices.

Overall, in the interaction described here, our integrated system uttered noun phrases to refer to individuals already present in the discourse state 10 times. In 7 of these 10 times, the system produced shorter noun phrases, including four uses of the pronoun *it*. The overall economy was a savings of 2 words in 3 cases, 3 words in 1 case, 8 words in 1 case, and 9 words in 2 cases. In the remaining 3 cases, the output of COLLAGEN's template-based generation system was already as economical as contextually possible.

To assess the improvement quantitatively would require a time-intensive exploration of a large space of possible user behavior and resulting interactions, which is methodologically prohibitive. Nevertheless, we have found the results presented here to be representative of results that occur in other interactions with this application. While we discuss some difficulties we encountered in the next section, we have generally found that our integrated system produces an appropriate gradient of unambiguous noun phrases. We take our experience to show that generic applications can exploit this architecture to generate hierarchical salience gradients that can be used to better tune linguistic output to context. As such it constitutes a promising venue for further research into context-sensitive generation for dialogue systems.

Discussion and further work

One issue brought out by our case study concerns how generated utterances place the individuals they feature into a discourse's evolving attentional structure, where they can serve as candidates for subsequent reference and as distractors in subsequent generation. A naive approach to focus stack update, which we initially took, is to simply read the newly salient individuals directly off of the domain representations that were used to drive generation. For example, in our email and scheduling application, upon generating and uttering a noun phrase designed to refer to `msg1`, we would simply add `msg1` to the top level focus space. However, this approach is inadequate because, in general, additional individuals can be evoked in an utterance as a consequence of how the generated utterance "presents" the intended referent to the user. A simple example is the gen-

Utterance #	Focus stack	Intended referent	Distractors	Utterance
24	msg1(John,Debra)	msg1	n/a (user)	USER: "Please forward the message from John."
25	msg1(John,Debra)	msg2.to	n/a (first mention)	COLLAGEN: "Who is *the recipient*?" SPUD: n/a (first mention)
31	msg2(user,Steve Kranz) msg1(John,Debra)	msg2	{}	COLLAGEN: "Please fill in the cc field of *the message from you*." SPUD: "Please fill in the cc field of *the message*."
48	msg1(John,Debra)	mtg1	n/a (user)	USER: "Let's schedule the meeting at May 5, 1999 10:00 AM with Brian."
52	mtg1(with Brian,May 5...) msg1(John,Debra)	mtg1	{}	COLLAGEN: "I'm going to add *the meeting at May 5, 1999 10:00 AM with Brian* to the calendar." SPUD: "I'm going to add *it* to the calendar."

Figure 3: Examples of noun phrase generation using two versions of our email and scheduling application. The italicized noun phrases in utterances 25, 31, and 52 were generated by the original template-based generation algorithm (marked COLLAGEN:) and by our integrated system (marked SPUD:). The user utterances, numbered 24 and 48, are included here to provide context for the generated utterances. msg2.to in utterance 25 is the unknown recipient of the forward of msg1 requested by the user in utterance 24. msg2, the actual forward, has entered into the focus stack explicitly by utterance 31. Some detail has been omitted for clarity.

eration of "the message from John" to refer to msg1, which makes not only msg1 but also the domain individual representing John more salient.

A more interesting example from our application arises in utterance 31 of Figure 3. Our application represents the action of filling in the cc field of an email as an action fillCCField taking a single message argument. When an action like fillCCField(msg2) is presented linguistically, for example as in

(4) Please fill in the cc field of the message,

the linguistic expression of the single domain action fillCCField is split between the verb "fill" and a contribution "the cc field of $[_{NP}$ _]" to its noun phrase complement. Notice that a human user, upon hearing (4), will model *two* new individuals as more salient and available for subsequent reference: both the message (msg2) and its cc field (msg2.cc). However the naive focus stack update strategy might counsel that only msg2 should be added to the top focus space. The consequence for generation is that a system employing this naive approach to focus stack update may compute subsequent distractor sets that fail to capture ambiguities the user will perceive. For example, suppose the user fills in the cc field of msg2 in response to a system utterance of (4), and then asks the system, "What next?" The system, attempting to express that (the body of) message msg2 needs to be completed next, and believing that msg2 has no distractors, might utter

(5) Finish it.

Yet the user would very likely perceive (5) as ambiguous between an instruction to finish the message body and an instruction to finish the somehow still unfinished cc field.

The right general moral here seems to be that focus state update must result from a model of which domain individuals *users* (hearers) take as having been rendered more salient by an utterance, rather than from a simple record of what domain individuals the dialogue system intended to express. Developing such a model can be difficult in systems that integrate template-based output with context-sensitive output. In our application, we could have assumed that users take all noun phrases as bringing their referents into salience, and avoided this problem. This would have been difficult however since the output of COLLAGEN's template-based generation, into which we were embedding SPUD's generated descriptions of action arguments, lacks any explicit syntactic information. This strategy would also necessitate the manual specification of the referents of all the extra noun phrases. Instead, we accepted that our focus state would incorrectly model ambiguities from time to time, and attempted to mitigate the consequences by adopting a heuristic of preferring more specific linguistic output in certain situations. In particular, we explicitly ruled out pronouns in problematic situations (which were identified manually). The use of other noun phrases, with their generally more restrictive semantics, cuts down on the number of possible unanticipated ambiguities. For example, while "it" might be unexpectedly ambiguous between msg2 and msg2.cc after an utterance of (4), "the message" is less likely to be. This is clearly not a complete solution, since "the message" might itself be ambiguous due to some other unexpected ambiguity.

A second issue that arose in our case study concerns first mentions of new individuals. It is often suggested (see (Reiter & Dale 2000), for example), that the generation of referring expressions in dialogue can be understood as falling fundamentally into two types. In *initial references* or *first mentions*, the intended referent has not yet appeared in the discourse, and what is said is assumed to depend entirely on application-specific goals. Properties and relations are included in first mentions if they are expected to be relevant in subsequent discourse. In *subsequent references*, on the other hand, what is said depends essentially on what *has* to be said in order to distinguish the intended referent; this is the problem that distractor set based generation is geared toward solving.

However, in our case study, we often encountered circumstances, such as that in utterance 25 of Figure 3, where intuitively what is desired is an *initial* reference, but where there are *no* application-specific goals apart from successful reference, and where, as is the typical case with subsequent reference, there is a preference for only saying what *has* to be said in order to distinguish the intended referent.[4] In utterance 25, for example, "the recipient" seems appropriate because the forwarding action requested in utterance 24 is so prominent. Yet we can easily imagine other discourse scenarios where an initial reference at a different level of detail would be appropriate. For example, if the user had said "Please forward the message from John and the file from Robert," the italicized NP of (6a) would intuitively be an appropriate initial reference. If the user had said "Please forward the message from John and the message from Robert," the more elaborate initial reference of (6b) might be necessary.

(6a) Who is *the recipient of the message*?
(6b) Who is *the recipient of the message from John*?

Thus, there appears to be a "purely referential" type of initial reference in which a contextually appropriate initial reference is desired, but in which application-specific goals do not motivate a particular detailed choice for the actual description. In our email and scheduling application, such contextually appropriate initial reference is desirable but not possible. Since individuals do not enter into COLLAGEN's focus stack until they have been mentioned, distractor sets are not available in cases of initial reference. Consequently, in our system, COLLAGEN's template-based generation mechanism, rather than SPUD, handles initial reference.

The intuitive naturalness of such contextually appropriate initial reference in discourse suggests a difficult problem: how can we generate noun phrases to an appropriate level of detail when distractor sets cannot be computed from the referent's current position in the focus stack? A general solution seems to depend on a deeper model of interpretation ambiguities than is available in COLLAGEN's focus stack, one in which an individual is salient not only according to its place in the intentional structure of the preceding discourse, but also according to its place in the strategy a hearer will employ in interpreting references to new individuals. Formalizing a generation strategy for contextually appropriate initial reference remains an interesting problem for future work.

[4] Of course, *some* application-specific goal motivates any given *entire* utterance — otherwise, presumably, there would be no point in saying anything. For utterance 25, the utterance-level goal is to propose that the user identify the recipient of the forward. But such a broad goal may not motivate any particular choice for an embedded initial reference.

References

Dale, R., and Reiter, E. 1995. Computational interpretations of the Gricean maxims in the generation of referring expressions. *Cognitive Science* 19(2):233–263.

Gardent, C. 2002. Generating minimal definite descriptions. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, 96–103.

Grosz, B. J., and Kraus, S. 1996. Collaborative plans for complex group action. *Artificial Intelligence* 86(2):269–357.

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

Grosz, B. J., and Sidner, C. L. 1990. Plans for discourse. In Cohen, P. R.; Morgan, J.; and Pollack, M. E., eds., *Intentions in Communication*. MIT. 417–444.

Gruen, D.; Sidner, C.; Boettner, C.; and Rich, C. 1999. A collaborative assistant for email. In *Proceedings of Human Factors in Computing Systems, Extended Abstracts*, 196–197.

Horacek, H. 1997. An algorithm for generating referential descriptions with flexible interfaces. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL)*, 206–213.

Jordan, P. W. 1999. An empirical study of the communicative goals impacting nominal expressions. In *Proceedings of the ESSLLI workshop on The Generation of Nominal Expressions*.

Krahmer, E., and Theune, M. 2002. Efficient context-sensitive generation of referring expressions. In van Deemter, K., and Kibble, R., eds., *Information Sharing: Reference and Presupposition in Language Generation and Interpretation*. CSLI Publications. 223–264.

Lochbaum, K. E. 1998. A collaborative planning model of intentional structure. *Computational Linguistics* 24(4):525–572.

Reiter, E., and Dale, R. 2000. *Building Natural Language Generation Systems*. Cambridge University Press.

Rich, C.; Sidner, C. L.; and Lesh, N. 2001. COLLAGEN: Applying collaborative discourse theory to human-computer interaction. *Artificial Intelligence Magazine* 22(4):15–25.

Stone, M.; Doran, C.; Webber, B.; Bleam, T.; and Palmer, M. 2003. Microplanning with communicative intentions: the SPUD system. *Computational Intelligence* 19(4):314–381.

Stone, M. 2003. Knowledge representation for language engineering. In Farghaly, A., ed., *Handbook for Language Engineers*. CSLI Publications. 299–366.