PRESUPPOSITION RESOLUTION WITH DISCOURSE INFORMATION STRUCTURES[1]

Paul C. Davis[2]

Abstract

An approach to resolving a number of presuppositional phenomena, including definite descriptions and pronominal anaphora, is described within the larger context of an architecture for query-based, task-oriented human/computer dialogue. The model of discourse context employed assumes that discourse structure is organized around a stack of questions under discussion, which plays an important role in narrowing the search space for referents and other presupposed information. The algorithms of individual presuppositional operators for maintaining discourse structures are presented and illustrated in several example dialogues in which human users interact with an agent in order to make hotel reservations. The overall architecture is compared to SDRT (Segmented Discourse Representation Theory), in terms of efficiency and ease of implementation.

1 Introduction

In the many theoretical treatments of discourse, a number of approaches have been used, some including such features as elaborate discourse structures, vast numbers of rhetorical relations, and plan inference engines. Clearly, as demonstrated

[1]This work is an extension of work presented in Kasper, Davis, and Roberts (1999). My thanks to Bob Kasper and Craige Roberts for their work on the project and earlier paper, and, along with Carl Pollard, for many helpful comments on this draft. Thanks also to Harry Bliss and Will Thompson of Motorola.

[2]Paul C. Davis is the recipient of a Motorola University Partnerships in Research Grant.

in the literature, many of these facets of the theories are necessary. However, in human/computer dialogue, when the domain goal is constrained to a single, overarching task to be completed (such as making a hotel reservation), a number of such theoretical prerequisites can be either simplified or factored out, without greatly reducing the coverage of the system. The goal of such a simplified approach is to make a dialogue system more computationally tractable and efficient (and may also make the system more modular and easier to implement).

In this paper, we begin by presenting a computational architecture for human/computer dialogue and demonstrating how it can be employed to solve a number of presupposition resolution problems in discourse. A highly structured discourse model, in conjunction with a treatment of referring expressions as presuppositional, enables us to develop a common strategy for resolving a number of reference resolution problems, such as pronominal anaphora and definite descriptions. This approach extends to a larger group of phenomena which we take to be presuppositional, including domain restriction, ellipsis, and lexically and syntactically triggered presuppositions. All of these constructions are presuppositional in a broad sense, in that their use assumes that certain information can be retrieved from the discourse context. Thus, we are deliberately adopting a broader sense of presupposition than has been conventionally assumed.[3] After the presentation of our approach, we describe its similarity to a more complex and fully-developed theory, SDRT (Segmented Discourse Representation Theory), and attempt to show how our simplified, modular architecture eases the processing task. The architecture consists of a number of modules. We will be focusing on two key components for the major part of the paper, however, the full system will be described in the final section, where it is shown in Figure 7. The two key components, the Question Under Discussion stack (QUD) and the Common Ground (CG), are central for the discussion in this paper. The QUD offers a means to represent the hierarchical nature of the discourse and provides a way of relating utterances to one another–i.e., for keeping track of which utterances are subquestions of earlier utterances, and which are answers–as well as keeping track of the most immediately salient discourse entities. Representing the hierarchical structure is important for certain problems in dialogue (see the example dialogues below) and the QUD can be useful in constraining the search space during the resolution process. The CG is a record of the informational content of the discourse, as well as what would be assumed to be everyday knowledge of the domain. Both the QUD and the CG may be accessed by what we will term presuppositional *operators* (such as the definite description operator) during the resolution process, and both data structures are necessarily dynamically updated as the discourse progresses. We believe this approach is well-suited to certain genres of task-oriented dialogue, in particular for mixed-initiative query systems, i.e., systems where either the human or the com-

[3]The idea that anaphora and presuppositions are closely related is not new (cf. van der Sandt (1992)). However, our treatment of presuppostions is integrated with our discourse model in a different way from van der Sandt, and we apply it to a broader range of phenomena (although van der Sandt (1999) does extend his approach to include domain restriction).

puter may pose questions to the other, toward the end of completing the dialogue task, which itself is generally constrained to one main goal (such as making a hotel reservation, or ordering a pizza).[4]

We will illustrate our approach with four example human-computer dialogues, shown below. SYS indicates the utterances spoken by the computer system. (Dialogue 1) (discussed in detail in §4.1) illustrates a case of pronominal anaphora resolution (*it* in (8)), in which recognizing the hierarchical structure of the discourse is crucial for identifying the antecedent, which was introduced many utterances earlier. The overall topic of the conversation is the question of where the user can find a hotel for June 15th in New York, and this super-question both facilitates and constrains the interpretation of *it* in (8). This example is similar to the well-known examples of long-distance anaphora in task-oriented dialogues described by Grosz (1981). Our approach is consistent with previous research that uses the intentional structure of discourse to determine a set of potential antecedents for pronominal anaphora. The following dialogues will illustrate how a broader range of reference and presuppositional constructions may also be addressed by using the discourse structure to guide the search for relevant information.

(Dialogue 1)	1)	USER:	I'm looking for a hotel for June 15th in New York.
	2)	SYS:	What part of the city would you prefer?
	3)	USER:	Manhattan, near Central Park.
	4)	SYS:	How many nights?
	5)	USER:	Just 1.
	6)	SYS:	Will anyone be traveling with you?
	7)	USER:	No.
	8)	USER:	Oh, I want **it** to have a swimming pool too.

(Dialogue 2) (discussed in §4.2) shows a definite description, *the hotel* in (7), whose referent can only be uniquely determined with respect to the indefinite hotel description (*a hotel close to Madison Square Garden*) in the question under discussion (1):

(Dialogue 2)	1)	USER:	I want to make a reservation at a hotel close to Madison Square Garden.
	2)	SYS:	What dates will the reservation be for?
	3)	USER:	March 3rd and 4th.
	4)	SYS:	Would you like a single room?
	5)	USER:	Yes.
	6)	USER:	Also, I'll need a conference room on the 4th.
	7)	USER:	I'd prefer it if **the hotel** had one.

[4]It remains an open question whether this approach is a useful one for more open-ended systems/tasks.

(Dialogue 3) (see §4.3) involves a contextually determined domain restriction, with a quantificational determiner *every*, illustrating that domain restriction must be handled in a similar way for a broader class of expressions than those which are normally regarded as referring expressions or presupposition triggers.

(Dialogue 3)	1)	USER:	Does **the Holiday Inn** have any vacancies for
			a) Tuesday, 12/4 - Friday 12/7?
			b) Thursday, 12/6 - Saturday 12/8?
	2)	SYS:	Yes, several.
	3)	USER:	Do they have a breakfast buffet **every morning**?
	4)	SYS:	a) Yes, Monday through Friday.
			b) No. There's a breakfast buffet Monday through Friday, but none on Saturday.

Finally, in (Dialogue 4) (see §4.4) we give a glimpse into our larger research program, where an elliptical question (3) must be resolved with respect to the question under discussion, in addition to establishing the reference of the definite description *the Marriott*, where the context might contain more than one hotel with that name:

(Dialogue 4)	1)	USER:	Which hotels near the airport have vacancies?
	2)	SYS:	The Holiday Inn and Sheraton have vacancies.
	3)	USER:	How about **the Marriott?**
	4)	SYS:	No, the airport Marriott doesn't have any vacancies.

The remainder of the paper is organized as follows. In section 2 we discuss our assumptions about the structure of discourse and the related background literature. In section 3, we present the individual operators and algorithms which we have developed in a partially completed implementation of a natural language dialogue system where users interact with an automated hotel reservation booking system. In section 4, we discuss the use of the operators and discourse structures to resolve the reference and presupposition problems shown in the above dialogues. In section 5, we describe SDRT, and then compare the two approaches. In the final section, we present an overview of the complete system and our plans for future development.

2 Background: Discourse Structure

Our characterization of the structure of discourse is based on the general theoretical framework of Roberts (1996), where discourse is formally viewed as a game of intentional inquiry.[5] As in Grosz & Sidner (1986), discourse is organized by the

[5]The material in this section is largely unchanged from that in Kasper *et al.* (1999), and was originally written by Craige Roberts.

interlocutors' goals and intentions and by the plans, or strategies, which the interlocuters develop to achieve them. Following Stalnaker (1979), the primary goal of the language game is communal inquiry: Interlocutors attempt to share information about their world, and the repository of that shared information is the interlocutors' *common ground* (CG). The set of acceptable moves in the game are defined by the (conventional and conversational) rules of the game, and are classified on the basis of their relationship to the goals. Ignoring imperatives, there are two main types of moves (see also Carlson 1983): questions and assertions. If the interlocutors accept a question, this commits them to a common discourse goal of finding a satisfactory (asserted) answer: Like the commitment to a goal in Planning Theory, this strong commitment persists until the goal is satisfied or is shown to be unsatisfiable. An accepted question becomes the immediate topic of discussion, the *question under discussion*. An assertion is a move which proposes an addition of information to the CG.

Roberts defines the structure of a discourse at a given point, its *Information Structure*, as a tuple which includes (among other things) the (totally) ordered set of moves in the discourse (M), CG,[6] and the stack of the questions currently under discussion at that point (QUD). The QUD is ordered by order of utterance and is updated in a stack-like fashion,[7] with questions popped when they are answered (or determined to be practically unanswerable). The ordered set of questions under discussion corresponds to the hierarchical intentional structure of the discourse. The QUD in this structure constitutes the set of *discourse goals* of the interlocutors; the discourse goals are only a subset of the set of common goals of the interlocutors, their *domain goals*, and the discourse goals are subordinate to the domain goals. Hence, the requirement that interlocutors stick to the question under discussion is just an instance of the more general commitment to plans; and in turn, in a fully integrated theory we would expect that domain goals and plans would influence interpretation as directly as the discourse goals represented by the questions under discussion.[8]

Any move in a discourse game is interpreted with respect to the Information Structure of the discourse at that point. There are two main aspects to the interpretation of any given move: its *presupposed content* and its *proffered content*, the latter including what is asserted in an assertion and the non-presupposed content of questions and commands. When an utterance presupposes a proposition p, then in order for the utterance to be felicitous in the context, p must be entailed by the CG (Stalnaker 1979). But in addition, any move in a discourse is interpreted by interlocutors under the Gricean meta-presupposition of Relevance, with Relevance

[6]Formally, in Roberts's (1996) framework, the CG is a function from M to sets of propositions, yielding for each move the common ground of the domain of discourse as it existed just before the utterance which the move represents occurred.

[7]However, all elements of the QUD list are accessible during the interpretation of an utterance. Only the top element is writable, but any entry is readable.

[8]Whether these domain goals need to always be computed will be discussed later in the paper (see sections 5 and 6).

formally defined in Roberts' framework, as follows:[9]

(1) A move m is **Relevant** to the question under discussion q iff (i) m is an assertion such that CG$\cup\{m\}$ entails a partial answer to q, or (ii) m is a question whose complete answer contextually entails[10] a partial answer to q.

(1(i)) tells us that the interpretation of an assertion will be constrained so as to yield a partial answer (possibly via contextual entailment) to the question under discussion. (1(ii)) tells us that the QUD in a felicitous Information Structure is constrained by Relevance so that each question on the QUD must address the (prior) question below it on the stack. Of course, (1) correctly predicts a variety of classical Gricean conversational implicatures, now characterizable as contextual entailments. But Roberts argues that Relevance is also crucial in *presupposition resolution*, broadly construed to include anaphora resolution, the interpretation of ellipsis, and domain restriction (Roberts 1995), as well as lexically and syntactically triggered presuppositions.

We will also assume the general approach to anaphora resolution argued for in Roberts (1999). The CG is augmented with a set of discourse referents familiar to the interlocutors, the *Domain* of the discourse context.[11] All definite NPs, including pronouns and demonstratives as well as definite descriptions using *the*, presuppose both *weak familiarity* and *informational uniqueness*. Weak familiarity (cf. the slightly different notion of familiarity in Heim 1982) is the theoretical realization of anaphoricity, and is licensed by existential entailments of the common ground, not requiring an explicit NP antecedent or even perceptual salience of the intended referent:

(2) **Weak Familiarity:** A discourse referent i is weakly familiar in a context C ($i \in$ Domain(C) and C encodes the information that i has properties $P_i, \ldots, P_k$) iff the Common Ground of C entails the existence of an entity with properties $P_i, \ldots, P_k$.

[9]This definition depends on defining partial and complete answers as is done in Roberts (1996), which is based on Groenendijk & Stokhof (1984)

[10]The notion of *contextual entailment* follows straightforwardly from Groenendijk & Stokhof's (1984) notion of *pragmatic implication*. That a contextually entails b simply means that the union of a with the context (in the present theory, this is the common ground) entails b.

[11]In the implementation, this *Domain* is implicit in the CG, in the sense that for all discourse referents there is an *instance* in the knowledge base, where an *instance* is simply a database object. We use the terms CG and knowledge base interchangibly in the paper, but it is important to realize that the latter is the implementation of former, and further, that there is more than one knowledge base in the system, i.e., one representing the CG and another representing the knowledge that the computer system has about hotels, and the like (e.g., how many rooms are vacant in a given hotel– such knowledge proves crucial when accommodation is necessary). These knowledge base distinctions are made where relevant elsewhere in the paper.

Informational uniqueness only requires that the discourse referent which satisfies the definite's familiarity presupposition be unique among the discourse referents in the local context in satisfying the definite's descriptive content. In other words, a referent need not necessarily be unique in the entire CG, but rather be unique in the current unit of discourse structure. The constraints of weak familiarity and informational uniqueness suffice to characterize the presuppositional content of definite descriptions:

(3) **Presuppositions of Definite Descriptions** (informal): Given a context C, use of a definite description NP_i presupposes that there is a discourse referent weakly familiar in C which is the unique weakly familiar discourse referent which satisfies the (possibly contextually restricted) descriptive content of NP_i.

Unlike Russell's (1905) theory, this does not generally entail semantic uniqueness, although in certain special contexts it will yield the same effect via pragmatic means. Definite descriptions may have their descriptive content contextually enriched in the same way that domain restriction works for operators generally, i.e., via Relevance to the question under discussion. This will be illustrated in our discussion of (Dialogue 4) (in §4.2). Many apparent counter-examples to the presupposition of uniqueness for definite descriptions are explained by appeal to this principled contextual enrichment, as discussed at length in Roberts (1999). Pronouns carry the additional presupposition of maximal salience:

(4) **Presuppositions of Pronouns** (informal): Given a context C, use of a pronoun Pro_i presupposes that there is a discourse referent i in C which is the unique weakly familiar discourse referent that is both maximally salient[12] and satisfies the descriptive content suggested by the person, number, and gender of Pro_i.

This amounts to an additional, conventional restriction on the search space for pronominal antecedents, implemented along the general lines suggested by Grosz & Sidner, and explains the differential distribution of pronouns and definite descriptions. We will discuss how maximal salience is implemented in terms of the QUD stack in §4. These presuppositional constraints result in a straightforward theory of anaphoric reference which explains a broad range of data and can be extended to a treatment of demonstrative NPs as definites, as well.

[12]The notion of the relative salience of discourse referents is discussed at length in Roberts (1998); for the purposes of this paper we assume the existence of an algorithm for ordering referents in terms of their salience. Informally, salience can be thought of as a measure of how closely related the referent is to the current discourse, i.e., how relevant it is, and to what degree it is in the *attention* (in the sense of Grosz & Sidner (1986)) of the participants.

3 The Resolution Process

In Figure 1 we show a simplified version of the main algorithm of the overall dialgoue system, and in Figures 2, 3 (shown in §4.1), 4 (§4.2), 5 (§4.4), and 6 (§4.4), we show simplified, pseudo-coded versions of the algorithms for some of the individual operators. The format of the individual operator terms, *OP(VAR, RESTR, NS)*, shown in Figures 2-6, where OP is the operator, VAR the variable, RESTR the restriction, and NS the nuclear scope, is in the familiar generalized-quantifier style, but may generate some confusion. First, our use of *restriction* and *nuclear scope* is not the same as that sometimes used when speaking of universal and existential quantifiers. When we refer to nuclear scope, we are not refering to the delimiting of the scope of a variable (i.e., we are not refering to the range in which a variable may be legally referred to–the range in which it is bound). Rather, the nuclear scope refers to part of the content of the utterance (i.e., the proferred content of that part of the utterance that the operator term represents). The restriction also refers to part of the content of the utterance (the pressuposed part). Informally, the restriction can be thought of as restricting what it is that is being talked about, while the nuclear scope is what is said about that restricted entity. So for the sentence *The man kissed Mary*, we might have a term such as: $def[x, man(x), kissed(x, M)]$, where the restriction is that x is a Man, and the nuclear scope is that x kissed Mary (for this discussion we ignore the familiarity and uniqueness presuppositions). We carry this format to our treatment of all operators. Thus, for pronouns (see Figure 3), which might traditionally be thought of as variables themselves (and therefore, under this traditional view, it would make no sense for them to have restrictions or nuclear scopes), the restriction again refers to what kind of entity it is, and the nuclear scope refers to what is said about the entity (and does not in any way delimit the scope of the variable), so similar to the definite description case, for *He kissed Mary*, our representation might be $pro[x, male(x) \land singular(x) \land third\_person(x), kissed(x, M)]$ (again, ignoring the familiarity and uniqueness presuppositions). Thus, we have a format for all of the operators, presuppositional and nonpresuppositional, which gives us a uniform way of delineating what the presupposed content is and what the proferred content is, which is very important for the resolution process.

Together, the algorithms for these operators drive the presupposition resolution process. Of central importance in this process is the maintenance of the QUD stack. Each entry on the stack is represented by a Question Data Log (QDL), an ordered triple which contains the utterance's[13] logical form (ULF), its Contextually Understood Logical Form (CULF), and a set of current discourse referents (CDRS). QDLs represent information about units of discourse structure which roughly correspond to the discourse segments developed by Grosz and Sidner.

Process_utterance (Figure 1) is the top-level function invoked for each discourse utterance.[14] The utterance is parsed to yield a logical form representing its context-

[13]An utterance is a full sentence or a fragment (e.g., "Yes."), and is not, in general, an entire turn.

[14]Of course, many of these steps may be eliminated when the system is the speaker. For example,

independent meaning (ULF). This ULF is further processed by *determine_CULF*, the goal of which is to produce a refined logical form (CULF) and a set of discourse referents (CDRS) by resolving presuppositions with respect to the current context. Presuppositions are represented in the logical form by certain operators, including *def*, *pronoun*, λ (for wh-questions), and *WH_Ellipsis*. The terms introduced by these operators, as well as other generalized quantifier terms, are processed by evaluation algorithms[15] for each operator (we call each of the evaluation algorithms *resolve_term*), each of which (again, see Figures 2-6) encodes individual pressuppostional requirements. The operators evaluate themselves relative to the discourse context. We believe this object-oriented methodology is suitable for implementing and testing different theoretical approaches, yet provides a common framework for development.

The set of presuppositional operators shown covers the examples that we will discuss, but is not intended to be exhaustive (an algorithm for non-presuppositional operators, such as indefinites, is shown in Figure 2). After *resolve_term* has processed a presuppositional term, the variable that it binds will appear on the CDRS list, and will either be identified with a set of referents from the common ground or be unanchored (indicated by '?'). This set of referents is a set of possible alternative referents, and may be required to be a singleton (e.g., in the case of the uniqueness presuppositions of pronouns and definite descriptions, see Figures 3 and 4), or may have more than one element, as in the case of a wh-question (where, for example, there can be many possible hotels given a question such as *Which hotels ... ?*). Once the CULF and CDRS are determined, the discourse structures, including the CG and QUD, are updated, depending on the type of conversational move (i.e., assertion or question). After the dialogue model has been updated, the CULF[16] is sent to the back-end application (e.g., to query or update its database), and the system may generate utterances as needed.

the system may pass a logical form directly to the dialogue system, rather than requiring parsing. Alternatively, the ouput of the generator may be fed directly back to the dialogue system for parsing. In this sense a system utterance may be treated no differently from a user utterance, should this behavior be desired. While such a strategy might seem somewhat perverse, it might be used in generation, for example, where different alternative system utterances could be generated and then reparsed, in an attempt to see which are easier to deal with (e.g., which lead to less potential ambiguities), before actually generating the sentences to the user.

[15]We generalize from the more familiar notion of a function to an operator which will have specialized implementation for arguments of different types, which is the general strategy in object-oriented program design. Thus, each individual operator has its own evaluation (or perhaps, more aptly, *resolution*) algorithm, called *resolve_term*. The appropriate version of *resolve_term* is indicated by the type of the operator.

[16]Again, this is a simplification. The application actually receives an operator free CULF, where instances from the CDRS have been substituted for variables. The reason for this is obvious, the application relies on the dialogue system to take care of the resolution process, and has no use for the presuppositional operators.

```
process_utterance (U)
%%% 1. Determine contextually interpreted meaning.
ULF = parse(U)[17]
(CULF, CDRS) = determine_CULF(ULF)

%%% 2. Update discourse structures.
If presuppositional operators remain,
   indicate non-acceptance of move (resulting in a prompt for clarification)
If U is an assertion:
   assert CULF to CG,[18]
   update QDL of QUD[top] (i.e., merge CDRS into CDRS of QDL)
If U is a question:
   push new QDL <ULF, CULF, CDRS> onto QUD

%%% 3. Signal back-end application.
Perform SYSTEM action (e.g., query or update database)
Perform SYSTEM dialogue move if necessary (e.g., generate a response)

-----------------------------------------------------------------------------
determine_CULF (U)
  if atomic_formula(U)  % contains no operators
     return (U, {})
  else (U must contain an operator)
     return resolve_term(U)
```

Figure 1: Dialogue system driver algorithm & determine_CULF

The algorithms presented here have been implemented in Common Lisp and, more recently, in C++, using the Loom knowledge representation framework (MacGregor (1991)) to maintain the common ground and background knowledge of the hotel application domain. Several components, for example, the *match&substitute* and *add_domain_restriction* functions, have not yet been implemented in a fully general way, and currently handle only simplified cases. The example dialogues discussed in the next section demonstrate how the resolution procedure works.

[17]The current formulation, of first parsing, then determining the contextually understood meaning, rules out interactions that might be desirable. Here we make a trade-off, opting for more straightforward software engineering over the possible benefits that could be derived from using information as soon as it is available (e.g., using contextual information to help in the parsing step). Of course, in taking this approach, we are not making any claims about the way humans do processing.

[18]Note that this is a simplification of what actually occurs in the implementation, where what gets asserted to the CG is not a logical form containing operators, but rather an assertion where knowledge-base instances have been appropriately substituted for variables, and operators have been removed, according to the resolution algorithms.

```
resolve_term(Term)
Let OP(VAR, RESTR, NS) = Term
%%% this representation uses destructuring by pattern matching, as may be
%%% familiar from Prolog, where OP is the top-level operator of Term, VAR
%%% the top-level variable, RESTR the top-level restriction, and NS the top-
%%% level nuclear scope.  This shorthand is used in figures which follow.

%%% Process embedded formulas inside-out
(CULF_R,  CDRS_R)  = determine_CULF(RESTR)
(CULF_NS, CDRS_NS) = determine_CULF(NS)

if OP is a non-presuppositional operator:
   DomRestr = add_domain_restriction(VAR, CULF_R, QUD)
   return (OP[VAR,DomRestr,CULF_NS], CDRS_R ∪ CDRS_NS)
```

Figure 2: Generic resolution algorithm for non-presuppositional operators

4 Discussion of the Example Dialogues

In this section, we discuss the dialogues given in the introduction (repeated here), and highlight how the presupposition resolution operators and algorithms can be used to resolve pronouns, definites, and quantifiers in general (i.e., reference related presuppositions, under our view) as well as other presuppositional phenomena, such as elliptical questions.[19] We illustrate the crucial changes which take place to the QUD data structures, allowing effective resolution of referents and presuppositions.

While the Utterance LF (ULF) describes only the literal content of an utterance, the CULF, along with the CDRS, can be thought of as a record of what the utterance really means, in the context in which it is said. For example, the following (ULF, CULF, CDRS) triple illustrates the QDL structure that results from question (2) of (Dialogue 2) (*What dates will the reservation be for?*):

(QDL 1) $\langle\lambda[x, date(x), def[y, reservation(y), for\_time(y,x)]],$
$\lambda[x, date(x), def[y, reservation(y) \wedge$
$\exists[z, hotel(z) \wedge near(z, MSG), at\_loc(y,z)],\ for\_time(y,x)]],$[20]
$\{(\text{x:date ?})(\text{y:reservation ?})\}\rangle$

[19]The careful reader will note that these dialogues contain additional reference resolution problems, such as *one*-anaphora (Dialogue 2) and a nonplural antecedent for *they* (Dialogue 3), etc., not discussed here for brevity.

[20]This CULF corresponds to what would arise given one of the possible parses for sentence 1 of (Dialogue 2) (note that there is no conjunct corresponding to the user wanting to *make y*; there are a number of subtle issues in instances such as this which still need to be worked out in this research, in terms of exactly what information is available, e.g., as mentioned earlier, *add_domain_restriction*

Each discourse referent in the set of CDRS is shown in the form *(variable:type instance)*, where *variable* is a logical variable from the CULF, and *instance* is an object in the model (i.e., it is from the knowledge base which represents the common ground–thus the CDRS can be thought of as a list of variables and their bindings). One fact to keep in mind when viewing the dialogues is that questions always produce a new QDL on top of the QUD stack, and therefore a new CULF and CDRS, while answers may update the CDRS of the QDL on top of the QUD stack (i.e., answers may introduce new entities, but these will be added to the CDRS of the relevant question), but answers never produce a new QDL. Another important point is that in the CDRS, there are discourse referents only for what is actually said in the given utterance–thus there is no referent for *hotel* in the CDRS in the example above. This is consistent with the idea that we need contextually enriched information for presupposition resolution, but do not want to create additional entities which could be referred to, for example, in pronoun resolution.[21]

4.1 Pronominal Anaphora: (Dialogue 1)

We will focus on the resolution of the pronoun *it* in the final utterance (8). We claim that at any time there is a set of accessible entities in the discourse, and when a pronoun is used in a discourse felicitously (i.e., as constrained by Relevance), there needs to be a unique maximally salient discourse referent for the pronoun belonging to this set of accessible entities. Under our approach, the set of accessible entities is represented by the union of the CDRSs of all entries on the QUD stack (again, entities mentioned in non-questions are also (potentially) available, since they are included in the CDRSs of the relevant questions on the QUD). Salience is a partial ordering on this set determined primarily by two factors. First, the members of the CDRS of each entry on the QUD stack are more salient than those for all entries below it on the stack. Second, the relative salience of discourse referents within the CDRS of a single QDL is determined by local constraints, such as those given by centering theory (cf. Grosz, et.al. (1995)), or the theory of focusing developed by Suri and McCoy (1994). Our overall approach could be adapted to use any theory of local coherence to determine a partial ordering over the CDRS within a discourse segment corresponding to a single QDL, but it is similar to Suri and McCoy's approach in allowing the CDRSs

still needs to be refined). Different earlier parses would thus lead to different contexts, affecting CULFs for utterances which follow–but for any given system/user interaction we will necessarily only choose one parse.

[21]In other words, we create an entity for *hotel* when that word is actually used, and that entity may or may not be available later to be referred back to. However, when we add conjuncts, for example, to the restriction for y in the example regarding the hotel, z, we don't want to make the mistake of reintroducing a referent that is already present in the discourse, since this could have negative effects later on, for example in the determination of salience, and the like. Again, we are not claiming that these entities are not available for reference, rather, the point is they were already made available (otherwise, we would never have been able to retrieve them in the first place) in the appropriate place in the discourse where they were used.

```
resolve_term(Term)
Let OP(VAR, RESTR, NS) = Term

%%% Process embedded formulas inside-out
(CULF_R,  CDRS_R)  = determine_CULF(RESTR)
(CULF_NS, CDRS_NS) = determine_CULF(NS)

RANKED_REFERENTS = rank_accessible_referents(QUD, CULF_R)
REF_SET = maximal_elements(RANKED_REFERENTS)
If singleton(REF_SET),
    %%% assume REF_SET = {INST}, substitute INST for VAR in CULF_NS
    return(CULF_NS[VAR->INST], {(VAR REF_SET)} ∪ CDRS_NS)
else report no salient referents or failure of uniqueness presupposition
```

Figure 3: Resolution algorithm for the pronoun operator

of prior questions to be stacked. Further explanation of how centering constraints can be integrated with our approach is given by Roberts (1998). In our implementation of pronoun resolution (see Figure 3), the function *rank_accessible_referents* gives the partial ordering of the accessible entities from the QUD, filtering out all entities that are incompatible with the agreement features of the pronoun,[22] which are represented in the restriction component of a pronoun term.

(Dialogue 1)	1)	USER:	I'm looking for a hotel for June 15th in New York.
	2)	SYS:	What part of the city would you prefer?
	3)	USER:	Manhattan, near Central Park.
	4)	SYS:	How many nights?
	5)	USER:	Just 1.
	6)	SYS:	Will anyone be traveling with you?
	7)	USER:	No.
	8)	USER:	Oh, I want **it** to have a swimming pool too.

In processing this dialogue, the system treats sentence (1) as a question (requests and statements of need and desire should be coerced to questions), and produces (CDRS 1), which is the set of discourse referents mentioned in sentence (1).

(CDRS 1) {(x:person user)(y:hotel ?)(z:date D1)(w:city NYC)}

[22]In reality, things aren't quite this simple, as pointed out by Carl Pollard (p.c.), since agreement features of pronouns and their antecedents do not always match, as in the number mismatch in the following example (as well as in the previously mentioned similar problem in (Dialogue 3)):
Are you sure you checked **every hotel***?*
Yes, **they** *are all full.*

As the system attempts to find out more specific information (imagine that it is filling out a template), it asks subquestions, such as (2), (4), and (6). After each subquestion, a new entry is added on top of the QUD stack, and therefore a new CDRS as well, e.g., the set of discourse referents in the top QUD entry after (2) is (CDRS 2).

(CDRS 2) {(w:city NYC)(x:area ?)(y:person user)}

When a subquestion is answered, as in (3), the CDRS of the current QUD is updated, e.g., the referent (x:area ?) becomes (x:area Manhattan), and a new referent introduced in the answer is added: (z:area CentralPark). However, once a question is completely answered it is popped off the stack. Thus, after (3) is completely processed as an answer to (2), the stack is popped, and later subquestions are also popped after processing (5) and (7). Therefore, when we arrive at (8), the QUD stack is just as it was after (1), since all of the intervening subquestions have been popped. This approach accounts for the observation that more recently mentioned entities, such as *Manhattan* or *Central Park*, are less likely as antecedents for *it* than those from (CDRS 1), which are closer in terms of hierarchical discourse structure.

In order to determine the antecedent for *it*, *rank_accessible_referents* only has to consider (CDRS 1), returning a subset from which (x:person user) is removed, because a person, being animate, does not match the restrictions of *it*. Thus, the search for possible antecedents has been significantly constrained by using the CDRS associated with the QUD. Among the remaining elements, the most likely antecedent is (y:hotel ?), which we call an *unanchored discourse referent*, since it is not yet bound to an actual instance of a hotel. This might be ranked highest by some versions of centering theory, because it is a complement of the verb, while the other referents were introduced by adjunct phrases (*for June 15th* and *in New York*). In general, however, pragmatic plausibility must be considered as an additional filter when determining whether a candidate is a potential antecedent. For example, (z:date D1) can be ruled out because it is not plausible for dates to have swimming pools.

4.2 Definite Descriptions: Dialogues 2–4

Although definite descriptions can often be identified with antecedents from the CDRS in essentially the same way as pronouns (since each CDRS is a subset of the CG Domain), they are not required to corefer with a maximally salient discourse referent. Therefore, our algorithm specifies three ways for a definite reference to be resolved. First, we check whether the CDRS accessible on the QUD stack contains a unique element that matches the restriction of the definite operator. Second, if there is no salient antecedent of the appropriate type, then we attempt to find a unique entity in the CG which satisfies the restriction. Third, if this fails, we use accommodation where possible to introduce an entity from the application's database into the

```
resolve_term(Term)
Let OP(VAR, RESTR, NS) = Term

%%% Process embedded formulas inside-out
(CULF_R,  CDRS_R)  = determine_CULF(RESTR)
(CULF_NS, CDRS_NS) = determine_CULF(NS)

REF_SET = all_accessible_referents(QUD, CULF_R) % possible anaphoric reference
If singleton(REF_SET),
   return(CULF_NS[VAR->INST], {(VAR REF_SET)} ∪ CDRS_NS)
else if |REF_SET| > 1,
   report failure of uniqueness presupposition
else % no salient antecedent, retrieve referent from common ground
   DomRestr = add_domain_restriction(VAR, CULF_R, QUD)
   REF_SET = retrieve_referents(VAR, DomRestr, CG)
   If singleton(REF_SET),
      return(OP[VAR,DomRestr,CULF_NS], {(VAR REF_SET)} ∪ CDRS_R ∪ CDRS_NS)
   else if |REF_SET| > 1,
      report failure of uniqueness presupposition
   else % attempt to accommodate, retrieve referent from application database
      REF_SET = retrieve_referents(VAR, DomRestr, ApplicationDB)
      If singleton(REF_SET),
        return(OP[VAR,DomRestr,CULF_NS], {(VAR REF_SET)} ∪ CDRS_R ∪ CDRS_NS)
      else if |REF_SET| > 1,
        report failure of uniqueness presupposition
      else report failure to accommodate
```

Figure 4: Resolution algorithm for the definite description operator

CG (see Figure 4).

(Dialogue 2)	1)	USER:	I want to make a reservation at a hotel close to Madison Square Garden.
	2)	SYS:	What dates will the reservation be for?
	3)	USER:	March 3rd and 4th.
	4)	SYS:	Would you like a single room?
	5)	USER:	Yes.
	6)	USER:	Also, I'll need a conference room on the 4th.
	7)	USER:	I'd prefer it if **the hotel** had one.

In (Dialogue 2), we focus on the resolution of *the hotel* in sentence (7). We first look for an appropriate antecedent in the CDRS accessible on the QUD stack, as in our treatment of pronominal anaphora, so we need to trace the stack for this dialogue. A request is made by the user in sentence (1), followed by a series of specific questions generated by the system. The QUD after (1) has the following CDRS:

(CDRS 1) {(x:person user) (y:reservation ?) (z:hotel ?) (w:place MSG)}

Subquestions are asked in (2) and (4) ((2) and (4) are subquestions of (1), which is treated as a question in the same way as sentence (1) of (Dialogue 1) was earlier, both being coerced to questions since they are statements of need or desire) and answered in (3) and (5), respectively, so the QUD stack is pushed and popped, but at (6), it is at the same state as it was after (1). (6) is interpreted as a request, so a new entry with (CDRS 6) is pushed onto the QUD on top of the QDL for (1).

(CDRS 6) {(x:person user) (v:conf-room ?) (u:date D4)}

In order to interpret the definite description anaphorically, we search for discourse referents whose type satisfies the explicit *hotel* restriction within the set of all accessible CDRS, viz., the union of CDRS 6 and CDRS 1. Since this set contains exactly one referent (z) which matches the *hotel* type, the uniqueness presupposition is satisfied and z is selected from CDRS 1 as the antecedent.

It is also possible for a definite description to have no explicit antecedent, as in *the Marriott*[23] in sentence (3) of (Dialogue 4). In such cases, an empty set of referents will be returned by *all_accessible_referents*, and our algorithm will attempt to retrieve a referent from the common ground. Before resolution, the content of this description is DEF 3, in which the variable ?NS is a placeholder for the unspecified nuclear scope of the def operator:

(DEF 3) $def[y, Hotel(y) \wedge Named(y, Marriott),$?NS]

The restriction of this term is obtained from the lexical entry for *Marriott*, which contains the information that it refers to a hotel, in addition to specifying its name. Although we rely on domain-specific knowledge in assuming that it refers to a hotel, we believe this assumption is reasonable, because the proper names for hotels can be automatically acquired from the hotel database used by the application.[24]

Now suppose that there are a number of Marriotts in the area. In an empty discourse context, this reference would have an unsatisfied uniqueness presupposition, so the system would need to ask the user which Marriott was intended. However, in

[23]Some might interpret the use of *the Marriott* in this dialogue as more of a name (i.e., the hotel chain as a whole, rather than a specific hotel near the airport), than a definite description. This is another difficulty that a system will have to cope with.

[24]We do not want to claim that linguistically this is the proper treatment in general, rather, it is a feature of having a fixed domain for the dialogue that we can take advantage of.

this case, uniqueness can be established by searching the QUD for an appropriate domain restriction, which can be conjoined with the explicit restriction given in (DEF 3). Since domain restrictions can be contextually supplied for most restricted operators, we interpret (DEF 3) as if there were an additional conjunct, which is schematically represented by QUD_RESTR(x) in (DEF 3′):[25]

(DEF 3′) $def[y, (Hotel(y) \wedge Named(y, Marriott) \wedge \text{QUD\_RESTR}(x)), ?\text{NS}]$

As in our treatment of anaphora, the key to constraining the search for an appropriate domain restriction is the QUD structure of the discourse. The entry on top of the QUD corresponds to question (1) of (Dialogue 4), whose CULF is (simplified):

(CULF 1) $\lambda[x, Hotel(x) \wedge Near(x, Airport), \exists[y, Date(y), HasVacancyOn(x, y)]]$

To determine whether any implicit domain restriction can be added to *the Marriott*, our algorithm calls *add_domain_restriction* to search the QUD for predicates that match the same basic type as the explicit restriction, *Hotel*. In (CULF 1) it finds the restriction $Hotel(x) \wedge Near(x, Airport)$,[26] which can be added in place of the virtual QUD_RESTR(x) conjunct in (DEF 3′) to further restrict the domain for *the Marriott*. This restriction (DEF 3″) is then used by *retrieve_referents* to find a matching referent in the CG.

(DEF 3″) $def[y, Hotel(y) \wedge Named(y, Marriott) \wedge Near(y, Airport), ?\text{NS}]$

It is important to note that the familiarity presupposition for a definite description does not require its referent to be previously mentioned in the discourse. In sentence (1) of (Dialogue 3), the referent for *the Holiday Inn* does not yet exist in our representation of the common ground, because the system initially has no knowledge that the user is aware of any particular Holiday Inns. In such cases, no objects are

[25]We do not actually include an explicit conjunct for the domain restriction in our implemented logical forms, because an implicit domain restriction may be added to virtually any restricted operator, as motivated by Roberts (1995), and it is of course possible for no new information to be added by domain restriction.

[26]Note that in this example, the user has no way of knowing if the system is using a *mention all* or a *mention some* strategy in its answer in (2). Thus (3) could take several meanings. The one assumed here (which seems the most likely): The user wants to check whether there is a Marriott at the airport with vacancies. Another potential interpretation might be: Being aware of an airport Marriott, the user wants to make sure that it too is full, before moving on. A third and perhaps less likely interpretation is: The user wants a Marriott at all costs, regardless of location near the airport. In instances such as these, the system will err on the side of assuming that elliptical questions relate as straightforwardly as possible to previous questions, since the sort of inferencing to determine the right interpretation is computationally expensive, and it is debatable whether a single interpretation is uniformly preferred by all speakers.

```
resolve_term(Term)
Let OP(VAR, RESTR, NS) = Term

%%% Process embedded formulas inside-out
(CULF_R,  CDRS_R)  = determine_CULF(RESTR)
(CULF_NS, CDRS_NS) = determine_CULF(NS)

DomRestr = add_domain_restriction(VAR, CULF_R, QUD)
case non-top-level OR non-question:  % non-presuppositional
    return (OP[VAR,DomRestr,CULF_NS], CDRS_R ∪ CDRS_NS)
case wh-question:  % presupposes some object satisfies DomRestr
    REF_SET  = retrieve_referents(VAR, DomRestr, CG)
    return (OP[VAR,DomRestr,CULF_NS], {(Var REF_SET)} ∪ CDRS_R ∪ CDRS_NS)
case polar-question:
    return (OP[VAR,DomRestr,CULF_NS], CDRS_NS)
```

Figure 5: Resolution algorithm for the lambda operator

returned from the CG by *retrieve_referents*, and the *definite* presuppositional term will remain with an unknown referent in the output of *determine_CULF*. Our approach to accommodation for such unsatisfied presuppositions (see Figure 4) is to look for a referent in the application's private database of facts about the domain of hotels, since this database represents all of the world knowledge that the system has available. Thus, the application must make its database readable to the dialogue system (i.e., it must provide an interface for read-access only). If the dialogue system finds a unique hotel named Holiday Inn, we can assume this hotel satisfies the user's presupposition. On the other hand, if it turns out that there are either no hotels named Holiday Inn in the application database, or multiple Holiday Inns, the system could report the failure of these presuppositions, rather than giving an uninformative simple negative answer to the user's question (1).

4.3 Generalized Domain Restriction: (Dialogue 3)

Consider next the quantificational determiner *every* in sentence (3) of (Dialogue 3). It should be clear that the user is not asking about every morning for all time, but only about all mornings during the planned trip. As with definite descriptions, our algorithm allows the restriction of most operators with semantically contentful restrictions[27] to be further specified by information from the QUD, so the

[27] Domain restriction is not usually applicable to pronouns and other expressions that have little explicit content, because these expressions depend on recovering a salient antecedent in order to determine the type of the referent, rather than searching for a particular type of object in the common ground.

interpretation of *every morning* will differ depending on whether the dialogue began with question (1a) or (1b). Now, if it is the case that the Holiday Inn has a breakfast buffet on weekdays only, it is important for the system to answer (3) appropriately, as in (4a) and (4b), depending upon the context created by (1a) and (1b).

(Dialogue 3)	1)	USER:	Does **the Holiday Inn** have any vacancies for a) Tuesday, 12/4 - Friday 12/7? b) Thursday, 12/6 - Saturday 12/8?
	2)	SYS:	Yes, several.
	3)	USER:	Do they have a breakfast buffet **every morning**?
	4)	SYS:	a) Yes, Monday through Friday. b) No. There's a breakfast buffet Monday through Friday, but none on Saturday.

To determine the domain restriction for *every morning*, *add_domain_restriction* searches the QUD for predicates that match the same basic type as the explicit restriction, *morning*. In this case, we take the basic type to be a temporal entity, so it will search for temporal descriptions in the QUD.[28] By using the QUD stack to constrain the search, *every* will quantify over any temporal entities that are found at a level of discourse structure closest to the current segment, but crucially not over every temporal entity in the entire common ground. Thus, to determine the response in (4a), only the date range mentioned in (1a) is relevant, and a positive response can be given, since the question relates to weekdays. In (4b) however, the date range includes a Saturday, so the system should generate a negative response. Thus, the system has employed domain restriction in answering *Yes* in (4a) and *No* in (4b), but note that in both cases that the system reports also to the user that the breakfast buffet is available *Monday through Friday*, even though these days do not exactly match the date ranges provided by the user. The reason for this is simple: In attempting to be as cooperative as possible, the system tries to provide complete information where it can.[29] In this manner, the user is not misled (that there is no buffet available on Monday), for example, had the system also used only the domain restricted dates in (4a), and answered: *Yes, Tuesday through Friday*. Here, as elsewhere, there are a number of tradeoffs in what sorts of strategies yield the most cooperative system, and clearly these could benefit from empirical evaluation of human-to-human systems, as well as human response to the given system, once it is complete.

[28]A complete explanation of this situation might require the system to infer the domain goals of the user. However, when the QUD contains some descriptions of the appropriate type, we can use them as an approximate domain restriction, thereby avoiding the computational expense of full plan inference (see section 5).

[29]But, of course, this does not extend to situations where complete answers would be too long to be useful, i.e., the *mention some* vs. *mention all* distinction previously mentioned, for example in answering the question: *which hotels have vacancies*, where a *mention all* interpretation would be rather uncooperative if thousands of hotels had vacancies.

```
resolve_term(Term)
  % Assume nuclear scope of Term is of the form: $\phi$[OldExpr->NewExpr]
  while QUD stack is not empty {
     QUD-CULF = CULF of QUD[top]
     QUD-CDRS = CDRS of QUD[top]
     (NewExpr, CDRS1)  = determine_CULF(NewExpr)
     if NewExpr is a generalized quantifier,
        let SubstLF = match&substitute(QUD-CULF,restriction(NewExpr),NewExpr)
     else (NewExpr is a predicate)
        let SubstLF = match&substitute(QUD-CULF,NewExpr,NewExpr)
     if null(SubstLF)
        or SubstLF is not interpretable as a subquestion of QUD-CULF,
           pop(QUD)
     else return (SubstLF, priority_union(CDRS1, QUD-CDRS))
     % priority-union(X,Y) is like set union, but when some members of X and
     % Y have the same type, only the member of X is included in the result.
     }
```

Figure 6: Resolution algorithm for WH_Ellipsis

4.4 Elliptical Questions: (Dialogue 4)

(Dialogue 4)	1)	USER:	Which hotels near the airport have vacancies?
	2)	SYS:	The Holiday Inn and Sheraton have vacancies.
	3)	USER:	How about **the Marriott?**
	4)	SYS:	No, the airport Marriott doesn't have any vacancies.

(Dialogue 4) is a somewhat more complex dialogue, including an elliptical question as well as several definite descriptions. It illustrates how our approach generalizes to the larger class of presuppositional constructions which we identified in the introduction. Let us focus on the interpretation of sentence (3), *How about the Marriott?*, which is assigned the following ULF:

(ULF 3) $Wh\_Ellipsis[\phi, Question(\phi),$
$$\phi[X \rightarrow (def[y, Hotel(y) \wedge Named(y, Marriott), ?\textsc{ns}])]]$$

ϕ is a variable referring to some contextually salient question, and the definite description corresponding to *the Marriott* is to be substituted for some term (X) within ϕ. Recall that the variable ?NS is a placeholder for the unspecified nuclear scope of the *def* operator.

The presuppositional operators process the logical form of an utterance *inside-out*, i.e., the embedded context resolution problems are handled first, so first the *def* term corresponding to *the Marriott* is resolved, as we discussed in §4.2 on definite descriptions, and *add_domain_restriction* produces the refined description (DEF 3″):

(DEF 3″) $def[y, Hotel(y) \wedge Named(y, Marriott) \wedge Near(y, Airport), ?\text{NS}]$

Next, the top-level *Wh_ellipsis* term in (ULF 3) is resolved, according to the *resolve_WH_Ellipsis* algorithm of Figure 6. ϕ must be a question, so we retrieve the question on top of the QUD stack, and attempt to identify ϕ with its CULF (CULF 1).

(CULF 1) $\lambda[x, Hotel(x) \wedge Near(x, Airport), \exists[y, Date(y), HasVacancyOn(x, y)]]$

We must now find a term within (CULF 1) for which the term corresponding to *the Marriott* can be substituted. Our *match&substitute* algorithm looks for terms whose restrictions specialize a common basic type, so it again finds the restriction on the (top-level) λ-term containing the *Hotel* predicate in (CULF 1).

The operator and restriction of this term (i.e., (CULF 1)) are replaced by those from (DEF 3″) and the variables are unified, but the nuclear scope of (DEF 3″) is unspecified, so the nuclear scope of (CULF 1) remains unchanged in the result:

(3″) $def[x, Hotel(x) \wedge Named(y, Marriott) \wedge Near(x, Airport),$
$\exists[y, Date(y), HasVacancyOn(x, y)]]$

(3″) is (almost) the CULF for *How about the Marriott?*, but it must be noted that it should be interpreted as a polar question, since the λ-term characteristic of a wh-question has been replaced by a definite description.[30]

Thus, both the elliptical question and the domain restriction of the definite description are processed by the same overall strategy: They are interpreted by incorporating information contained in the question under discussion.

5 SDRT and its Relation to the Present Theory

In the preceding sections of this paper, we presented an architecture for a dialogue system, and described how it would handle several example dialogues. But, of course, in the research on discourse, a number of such theoretical approaches have been developed. In this section, we briefly describe one widely-used system, Segmented Discourse Representation Theory (SDRT), and then contrast it with our own, especially in terms of what we estimate to be the computational resources required for each approach.

[30]When all top-level λ-terms in a wh-question have been replaced, it is interpreted as a polar question.

5.1 The Roots of SDRT

SDRT, as its name implies, is a descendant of DRT (for a comprehensive introduction to DRT, see Kamp & Reyle (1993)). DRT is a semantic theory which focuses on the representation of discourse, and may be thought of as a dynamic, procedural theory, reflecting the idea that sentences are interpreted step-by-step, as they are uttered. As such, it presents a radical departure from Montague grammar, in both its reliance on the procedural nature of a discourse, and its focus on series of sentences rather than a single sentence.

Like Montague grammar, DRT offers a logical representation of utterances, but diverges in where and how the logical representations are stored. In Montague grammar, syntactic structures are translated directly into logical structures, and these logical structures may then be interpreted with respect to a model. It is in theory possible to eliminate the translation step, since the logical structures derived are a function solely of the parts from which they were formed (i.e., the whole is no greater or less than the sum of its parts). In DRT, however, there is also a representation of the context in which a sentence occurs. Thus DRT has an additional level of representation, and it is in this level that logical representations of sentences are stored (this will become clearer below). Because of these differing levels, interpretation in DRT proceeds rather differently. Rather than interpreting a sentence directly with respect to a model, in DRT the truth of a discourse is defined in terms of whether the information contained in the representation of the discourse can be embedded in a model.[31] It is in this sense that many view DRT as noncompositional, since the interpretation of an utterance is not a function of just the parts of the utterance, but rather a function of the utterance and the previous context, yielding an updated context. This view is not really correct, however, since there are also compositional versions of DRT (for example, Zeevat (1989)).

The development of DRT was partially motivated by phenomena that were problematic for Montague grammar, such as intersentential anaphora. For example, in the discourse:

(5) Exactly one person made a reservation. She's staying a week.

A Montagovian system cannot handle this type of discourse, because the antecedent for the pronoun *She* occurs across a sentence boundary. And the solution of conjoining the logical representations of the two sentences does not work either, because the process of quantifying in the term *Exactly one person* after the sentences are conjoined creates the wrong meaning, i.e., we get:[32]

(6) $\exists y \forall x((Person(x) \wedge Made\_a\_reservation(x) \wedge Stay\_a\_week(x)) \iff x = y)$

[31]Much of the material in this section relies heavily on chapter 7 of Gamut (1991).

[32]In this section, we avoid the generalized quantifier notation used elsewhere, simply to maintain a more transparent relation to the Montague style.

where some other person could have made a reservation, rather than what we want:

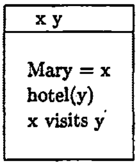
(7) $\exists y \forall x((Person(x) \wedge Made\_a\_reservation(x) \iff x = y) \wedge Stay\_a\_week(y))$

DRT resolves a number of such difficulties. In DRT, the variables are called *markers*, and the formulas in which take markers as arguments are called *conditions*. Markers are introduced for indefinites and proper names, and are a way to keep track of the individuals mentioned in a discourse–the *discourse referents*. There are two common and equivalent notations for DRT, a linear notation, and DRT's distinctive "box" notation. We will use the latter here. The general idea is that the context of a discourse is represented by the box, and in the box are the markers and the conditions. Thus, the boxes help determine the scope (DRT theorists tend to refer to the *accessibility* of a marker) of the markers, and the graphic representation provides an intuitive feel for exactly what is accessible when. For example, in the one sentence discourse:

(8) Mary visits a hotel.

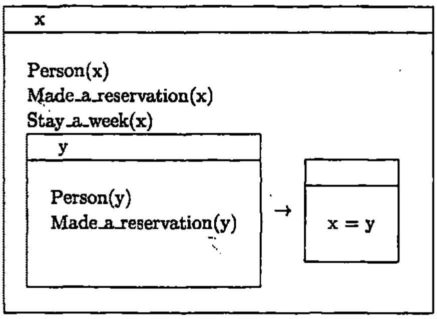
There will be markers, x and y, for the two discourse referents mentioned (*Mary* and *a hotel*) as is shown in the following box, which is called a DRS[33] (Discourse Representation Structure), where markers are at the top, and conditions below:

(9)



We will not go into the details here of how such DRSs are constructed, nor the rules for accessibility or interpretation, rather, at this stage we are trying to give the reader a basic feel for how DRT works, so that we can give a similar sketch of SDRT. Continuing on then, a DRS for the discourse in (5), would look as follows:

(10)



[33]This should not be be conflated with the CDRS, mentioned in earlier sections of the paper.

The major point to take from here is that the person referred to in the first sentence, is available to be referred back to (i.e., it is accessible), illustrated graphically by the fact that all the conditions on x are inside (however deeply embedded) the same box where x is introduced as a marker.[34]

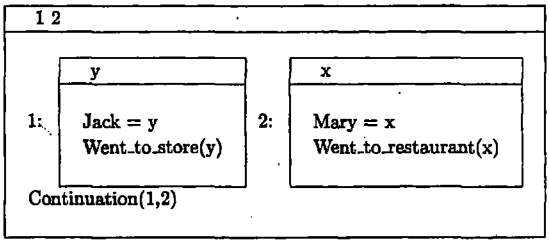
5.2 SDRT

SDRT (Asher (1993), Lascarides & Asher (1993)) builds on DRT by adding the notions that DRSs should be related to one another in a more formal way (i.e., rather than simply being contained in or merged into, for example, the same larger DRS). SDRT employs rhetorical relations, along the lines of Mann & Thompson (1987), to relate DRSs. These rhetorical relations, the number and types of which are often debated in the literature, include relations such as *elaboration*, *explanation*, *background*, and *contrast*, and are an attempt to describe the way one unit of a discourse (often a sentence) relates to another (i.e., what pragmatic purpose does it serve). Thus, SDRT is a theory of the structure and content of discourse.

SDRSs are like DRSs, but each SDRS has a unique label, and in addition to containing DRSs and other SDRSs, an SDRS can contain conditions where rhetorical relations are the predicates and labels of SDRSs are the arguments. In short then, in SDRT we see a new type of marker, i.e., the labels representing SDRSs, and a new type of condition, but (at least graphically) things are very much the same as in DRT. For example, in the discourse:[35]

(11) Jack went to the store. Then Mary went to the restaurant.

The SDRS constructed might look as follows, where we assume the rhetorical relation is *Continuation*:[36]

(12)


[34]Things in DRT are obviously much more complex than this, e.g., discourse markers are accessible when DRSs are *subordinate* to each other, for example when connected by →, the markers of the *antecedent* (i.e., left) DRS are available to the *consequent* (i.e., right) DRS.
[35]We use a simple example here, to avoid issues with quantifiers, anaphora, etc.
[36]Again, this is a simplification of what the theory would yield.

By relating SDRSs with rhetorical relations, SDRT represents the hierarchical structure of the discourse, and captures important meaning differences between discourses such as (13) and (14):

(13) Jack smiled. Then Mary told her joke.

(14) Jack smiled. Mary had told her joke.

where in the first case the rhetorical relation would likely be *Continuation* and in the second case *Explanation*.

In providing this hierarchical and discourse content, SDRT provides quite a bit of information which can be used in difficult areas, such as the presupposition and anaphor resolution discussed earlier in this paper. But the additional information does come at a price, compared to DRT. Whereas in DRT, an update function (i.e., a function to update the global discourse context after a new utterance has occurred), can be as simple as unioning the markers and conditions of the new utterance with those of the discourse so far, in SDRT the update function becomes much more complex (see Asher & Lascarides (1998b) for a detailed discussion). In addition to computing which rhetorical relation to use, the attachment point for the new information must also be computed. Asher and Lascarides accomplish these steps with a nonmonotonic logic, and a number of constraints on the types of discourses which can occur. For example, much like Gricean maxims, attachments and relations which maximize discourse coherence are preferred. By constraining the SDRS construction in this way, Asher and Lascarides make strong claims about which information is available at later points in a discourse, and they provide a mechanism for adjusting the discourse structure which can be adapted depending on the current theory, or for that matter the language or style of discourse.

In addition to the aspects already mentioned, SDRT extends DRT by attempting to model the intentional structure of the participants in a discourse. This can have advantages, in terms of discourse processing, since as a number of works have shown (e.g., Grosz & Sidner (1986)), modeling dialogue can require knowledge of the plans of the participants. But, of course, this addition also comes at a computational cost, as will be discussed shortly.

Extending SDRT to dialogue is straightforward, and indeed seems a natural fit. In Asher & Lascarides (1998a), new rhetorical relations are introduced which are appropriate for questions in dialogue, including Question Answer Pair (QAP), Indirect Question Answer Pair (IQAP), and Question Partial Answer Pair (QPAP), along with new axioms on when the relations can be used.[37] The (computed) intentional structure of the participants plays a key role in using these new relations. Because

[37]Here, as elsewhere, for brevity we do not go into the inner workings of the SDRT approach, but again try to provide an overview of the benefits and costs.

of this, Asher and Lascarides must maintain separate SDRSs for each discourse participant, since not only may different participants' intentions vary, but so may their perceptions of intentions (e.g., x believes y intends z, and y knows x believes y intends z, and x thinks y doesn't know x believes y intends z, and so on).

5.3 Comparing the Current Approach with SDRT

We are now at a good point to view the simplified architecture presented earlier in this paper in SDRT terms, and to compare the two approaches. It should be emphasized that we are not attempting to point out weaknesses in SDRT here. Rather, we hope to demonstrate that in certain contexts, such as query-based human/computer dialogue where the domain goals are fixed, that a simplified architecture will often be sufficient, and therefore efficiency gains come at a low cost in terms of coverage. Indeed, in many ways, the simplified approach is a subset of SDRT. Further, our approach is a work in progress, since the entire system is not yet implemented,[38] and we hope that as the work in SDRT advances, it will help inform our own.

First, one of the obvious differences between SDRT and our approach is that we maintain uniform discourse structures, rather than separate discourse structures for each participant.[39] We are able to make this step because we do not (at least not in the discourse structures, see the next section) explicitly model the intentional structure of the participants. Indeed, when the domain goal(s) are fixed, as in the case of a hotel reservation system, the intentions of the human participant can be assumed (i.e., the overarching goal of making a hotel reservation). From the perspective of SDRT, our approach could be viewed as one where all participants happen to have the same SDRSs, i.e., they agree on their perceptions of each other. This brings up an important theoretical point. If interlocuters are engaging in cooperative dialogue, then they are following the rules of the dialogue game, i.e., they are obeying Gricean maxims, and the like. Therefore, we might want to require that there be uniform discourse structures, such as the CG. Discourse is a communal activity, and for it to be successful, the participants have to agree on what the goals and state of the discourse are.[40] From this point of view, the separate discourse structures of SDRT do not appear to be highly motivated. One final point regarding the issue of uniform vs. separate discourse structures is that, in general, maintaining single, uniform structures for the discourse, rather than for each participant, should reduce both the space and time involved in computation.

[38]Consequently the efficiency gains are necessarily only an estimate. Plus, we are not aware of any completed implementations of SDRT, although a promising demo was given by Bohlin & Larsson (1999). So, the comparison in this section is a bit difficult from the get-go, since we compare our theory and partial implementation with a (at this point) more explicit theory and no implementation; nevertheless, some important differences do arise.

[39]Another type of QUD structure by Ginzburg (1996) maintains a separate QUD for each participant, but we do not believe this to be necessary for this type of dialogue.

[40]Thanks to Craige Roberts for pointing this out to me.

A more important difference centers on our reliance on the QUD stack. The QUD functions as both the representation of the discourse hierarchy and as the implicit store of some of the rhetorical relations between utterances (i.e., it serves to relate questions and answers, and superquestions with subquestions), while the informational content of the discourse is ultimately stored in the CG. In SDRT, on the other hand, all of this information (as well as the intentional structure) is stored in the SDRSs. This has the advantage of making all the information available in one place for processing, but sacrifices modularity (we return to this in the next section). The implicit relations in the QUD are not unlike those used for questions in SDRT. For example, each question and answer form a question/answer pair, and these relations must be computed under both approaches. However, because we assume that the discourse revolves around the idea of answering the current question under discussion, until the domain goals are completed, we have a much smaller inventory of relations to choose from.[41] We must also keep track of the relation between subquestions and superquestions (again, this is accomplished by using the stack-like data structure, where any question on top of another is a subquestion), and, of course, compute when they occur. Finally, we must also interpret requests and statements of desire as questions. Given this approach, our system does especially well in cooperative dialogues, where users do not switch back and forth between unrelated questions, since such switching might cause the QUD to be popped, and therefore need to be adjusted, once earlier questions are resumed. Thus, our system, as described, does not presently tolerate asides well, but certainly this is not a theoretical limitation of the system. By taking advantage of cue-phrases (such as *by the way*, *that reminds me*, etc.), a system can discern when the question under discussion is being changed to a (possibly) unrelated one, and adjust its discourse structures accordingly.[42] And, indeed, if a dialogue is coherent, interlocuters must adequately signal such changes in the topic of discussion, or they are violating the maxims of cooperative dialogue.

In our approach, the relation between questions and answers is indeed implicit, since we do not store the answer alongside the question (although, importantly, we do store a representation of the discourse referents and information updates contained in answers, in the CULF and CDRS of the relevant question). And, once a question is completely answered, it is removed from the stack. This means, again, that our machinery, like in SDRT, must be able to compute what entails an answer to a question, and must be sensitive to user input to help indicate when questions have been answered successfully (indeed, we rely on this information, and let the user "drive" the dialogue). Note that we do not really lose any information in popping the QUD, since we keep a record of the information content in the CG, as well as a record of the hierarchical structure, since each utterance record, what we call

[41]One of the criticisms of theories that use rhetorical relations is not only that they are difficult to compute, but that those typically used may not be exhaustive enough for many discourses.

[42]An obvious, although inelegant solution would be to simply pop questions onto a second stack-like data structure, where they could be accessed as necessary for re-pushing onto the QUD. Thus, each dialogue would have its own discourse structures as its disposal. We would also need algorithms for backtracking, just as humans do (i.e., *the Okay, now where are we?*).

a Move, points to the QUD at the state it was in when the utterance occurred. By maintaining the QUD (i.e., pushing and popping, as well as updating), we too make claims about which discourse referents are accessible–but unlike SDRT, there's only one possible attachment point, at the top of the QUD. Again, we believe this represents a significant savings in computation.

This now brings us to the question of search space. When we encounter discourse referents which need to be resolved, the first place we look (the strategy differs from operator to operator, but this is the most common approach) is the top of the QUD, where there is a single question, and a number of referents contained in the CDRS corresponding to that question. The search may involve looking at super questions (i.e., lower on the stack) and ultimately, as described in section 4, looking in the CG or possibly being accommodated. We posit that a large proportion of the time, we will find the material in the QUD (the QUD thus functions somewhat analogously to a cache, in terms of finding the most recent or salient referents). Depending on the axioms used in the SDRT implementation, a search may be much slower. However, this appears to be a point where the systems are similar, if the SDRT implementation is sufficiently constrained (e.g., only referents on the right frontier are accessible). But, once again, the computation which must take place beforehand (i.e., what will be attached where) is more expensive in the SDRT case.

In terms of coverage, SDRT appears to be the winner, at least compared to the coverage of our implementation so far. For example, at the moment, our implementation provides no mechanism to refer to entire utterances anaphorically, much less spans of dialogue, or to general ideas/intentions which can be inferred from it. Our hypothesis is that users interacting with a computer will not only tolerate such deficiencies, but quickly adapt to them. Nevertheless, we would be severely mistaken to say that we have complete coverage. But, like other theory vs. implementation concerns raised in this paper, this is not a theoretical limitation, just a mechanism which has not yet been built. Since we already track each utterance (e.g., in the move history, M), and since segments of discourse are represented in the QUD, we could certainly add the means to refer back to utterances or segments of discourse anaphorically (perhaps by some representation or pointer to the appropriate object in the knowledge base). Ideally, our use of the QUD would also help to narrow the search space in searching for the referents–in the case of utterances and discourse segments–since interlocuters would be likely to refer back to the most salient discourse entities (much like earlier our discussion regarding pronouns; for a further look at referring back to 'larger' entities (i.e., whole utterances, propositions, etc.) anaphorically, see Roberts (1995)).

The last, and perhaps most important area in terms of computational cost is the question of modeling intentional structure. We do not, at least not as part of the dialogue system itself, explicitly model the intentional structure of the participants. Thus, as just mentioned, we will miss any references to plans, as well as some resolution problems that rely on the correct identification of intentions. However, given the exceedingly expensive nature of computational inference in calculating plans, we believe this to be a significant advantage in a domain such as the one described. Again, we are able to take this step because much of the plan structure can be inferred from the domain, and from the roles served by the computer and humans (the computer always helps the human make a reservation, not the other way around). Additionally, we do provide a mechanism for planning to be incorporated by applications (see the next section); we simply do not integrate it into the dialogue system proper.

Thus, an implementation using the architecture we described should be significantly faster than one using SDRT, but the speed-up does come at a coverage cost. And, while in the final section we will describe advantages that a modular approach brings, an integrated approach such as SDRT certainly offers organizational advantages, in that all of the information is in one place (i.e., the intentional structure is included in the SDRSs, not in a separate planning component, and the hierarchical structure is at the same level as the informational content–relations in and between SDRSs–rather than being in separate but linked data structures such as the QUD and CG).

6 Conclusions

In the previous sections of this paper, we have focused on specific parts of the dialogue system, mainly the QUD and CG, and discussed their role in the presupposition resolution process. In this final section, we present the overall architecture, and discuss the benefits of using a modular approach to the dialogue system, and then finally outline our plans for future work.

6.1 Overview of Architecture Described

As shown in Figure 7, the dialogue system is comprised of a number of modules, including the CG, the QUD, a Parser, and a Generator. Each of these modules has a number of operations specified for them, and are only accessible to one another via their interfaces. Thus, as is normally the case with object-oriented design, each part of the system is viewed as a separate entity, with a number of different tasks it can be asked to do. For example, the QUD may be asked to pop itself, the CG to return the answer to a query, and the Parser to parse the utterance spoken by the user. Also shown in the diagram are the operators and algorithms which guide the resolution process, grouped as a unit, although each operator may be viewed as an object (more accurately an instance of an object) in its own right, and the set of conversational moves, the *Move History* (which was abbreviated in earlier sections as *M*). Once an utterance is superficially parsed, the system checks the output of the parser for any resolution necessary (i.e., for the presence of any presuppositional operators), then asks the operators to evaluate themselves, again by accessing the data structures when necessary, as shown in section 3 earlier. When finished, the now contextually understood logical forms (CULFs) and other information from the sentences are passed to the appropriate data structure modules.

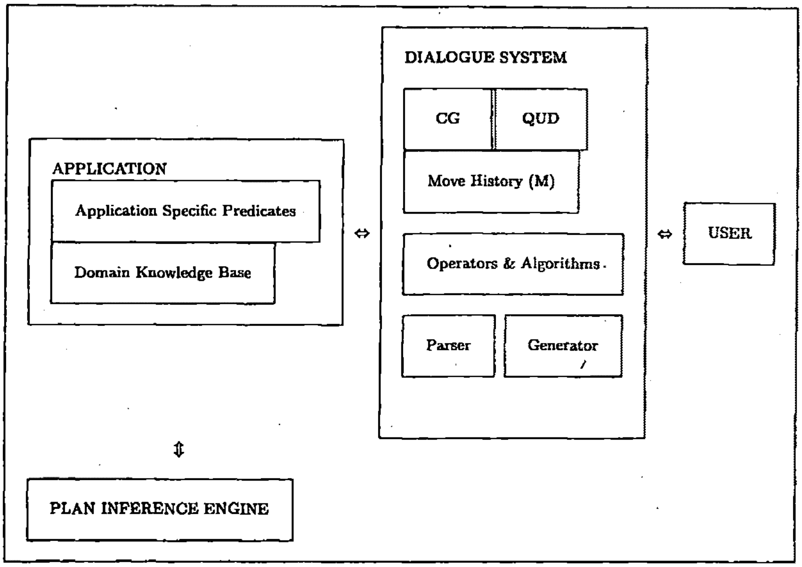


Figure 7: dialogue system architecture

The true modularity of the system comes into play with regard to the application. An application, such as the hotel reservation system described in this paper, may access any of the modules of the dialogue system, and indeed, may inform them. For example, in this instance, the application may have specific predicates which are applicable to the domain, for example *have-vacancies(h,d)* (where h is a hotel and d is a date), that would most likely not be relevant for other domains such as an airline ticket booking system. The dialogue system itself will have been initialized with a number of expected predicates for dealing with queries, monetary transactions, and the like. Application specific predicates may then be made available to the dialogue model (i.e., for this application only) to supplement the terminology in the CG for processing. This is consistent with the idea that most users contacting a hotel reservation system will know how hotels function. Thus the system has enough information to handle generic query-processing dialogue, and can be customized for specific task domains.

The overall interaction of the application and the dialogue system can be seen as a series of messages being passed. The application may receive an indication (say, from a user interface) that the user has communicated something. The application then asks the dialogue system to begin processing the input. The system will, as described, parse and resolve the input, and after updating the various data structures, signal the application, and optionally pass the logical form to the application.[43] The application may then (if desired) access the data structures, in order to decide what step it will take, if any. In this sense, the dialogue system is a *dumb* system. It stores the information, and then allows the application to retrieve information, to get an accurate view of the state of the discourse at any time. Here is where an optional planning engine might come into play. As mentioned earlier, plan inference is expensive, and may not always be desirable. But, if opted for, an application can pass information from the dialogue system to a planner (see Figure 7), to help it decide what conversational act it should next perform. Again, we see this modularity as an advantage, because it allows an application to be as intelligent as it can computationally afford to be. Certainly, we assume some kind of planning capability must be available for an application to be successful, but for some applications, this may amount to nothing more than form-filling (i.e., checking which items still need to be filled in and querying the user accordingly–in any case plan inference is not yet a focal point in the current research project).

Similarly, when the system desires to "speak" to the user, it can pass a logical form representing the content to the generator. The generator then, depending on its sophistication, may access the discourse structures, to see exactly what type of utterance is preferred (for example, a good generator may use pronouns, if it can determine what discourse referents are the most salient). Again, the overall modularity of the system comes into play here, because the generator may be developed by a different set of people than those who work on the application or the parser or the dialogue system. We hope, therefore, that provided appropriate interfaces, we have created an architecture which can not only be used with a number of different task-oriented applications, but can also be continually improved upon, as each module is developed.

6.2 Summary and Future Work

We have presented what we feel is a streamlined and modular architecture for a human/computer dialogue system, and have attempted to demonstrate how the discourse structures of such a system can be used to facilitate efficient resolution of a number of phenomena which we take to be presuppositional, such as definite reference

[43]Another step that must take place in the implementation, which like many other implementation-specific details is not covered at length in this paper, is what we call *domain specialization*, from the language produced by the parser, to that used by the dialogue system and application. Domain specialization is a kind of logical coercion, which involves the specialization of general predicates into domain specific ones. For example, as mentioned earlier, application specific predicates such as *have-vacancies* won't be directly produced in the logical forms outputted by the parser, but are used in the knowledge base. This specialization may be implemented in the operators, but this sacrifices modularity (since we would not, for example, in every application always want an operator to insert a date argument any time it saw questions regarding vacancies). A better approach is to allow the application to provide a set of domain specialization rules, which the parser may access, to produce logical forms using the appropriate predicates.

and anaphora. We have also briefly described how this approach can be extended to other presuppositional phenomena, such as domain restriction and ellipsis.

In comparing our approach to SDRT, we have pointed out a number of deliberate design features in our approach (e.g., the lack of a persistent intentional model of the user) which we believe make the system more computationally tractable, and certainly more easily implemented. Indeed, these sorts of resource-saving sacrifices are often necessary in practical implementations.

Our architecture allows different developers to focus on different parts of the process. The current research aim has been to develop the overall interfaces, and to write the resolution algorithms for the individual operators. In the future, we hope to integrate our work with other modules, such as a generator, and to develop a user interface, so that the entire system may be empirically evaluated. Once this step is accomplished, we can then compare our system performance with real-world data, in an attempt to both better inform our theory of discourse and improve our system performance.

REFERENCES

ASHER, NICHOLAS. 1993. *Reference to Abstract Objects in Discourse*. Dordrecht, The Netherlands: Kluwer Academic Publishers.

——, & ALEX LASCARIDES. 1998a. Questions in dialogue. *Linguistics and Philosophy* 23.237–309.

——, & ——. 1998b. The semantics and pragmatics of presupposition. *Journal of Semantics* 15.239–300.

BOHLIN, PETER, & STAFFAN LARSSON. 1999. Godis and the dialogue move engine toolkit. In *Demonstration Abstracts from the 37th Annual Meeting of the Association for Computational Linguistics*, College Park, Maryland. Association for Computational Linguistics.

CARLSON, LAURI. 1983. *Dialogue Games: An Approach to Discourse Analysis*. Dordrecht: Reidel.

GAMUT, L.T.F. 1991. *Logic, Language, and Meaning, volume II*. Chicago, IL: The University of Chicago Press.

GINZBURG, JONATHAN. 1996. Interrogatives: Questions, facts and dialogue. In *Handbook of Contemporary Semantic Theory*, ed. by S. Lappin. Oxford: Blackwell Publishers.

GROENENDIJK, JEROEN, & MARTIN STOKHOF. 1984. *Studies on the Semantics of Questions and the Pragmatics of Answers*. University of Amsterdam dissertation.

GROSZ, BARBARA, & CANDACE SIDNER. 1986. Attention, intentions, and the structure of discourse. *Computational Linguistics* 12.175–204.

GROSZ, BARBARA J., ARAVIND K. JOSHI, & SCOTT WEINSTEIN. 1995. Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics* 21.203–225.

GROSZ, B.J. 1981. Focusing and description in natural language dialogues. In *Elements of Discourse Understanding*, ed. by B. Webber A. Joshi & I. Sag, 84–105. New York: Cambridge University Press.

HEIM, IRENE. 1982. *On the Semantics of Definite and Indefinite Noun Phrases.* University of Massachusetts at Amherst dissertation.

KAMP, HANS, & UWE REYLE. 1993. *From Discourse to Logic.* Dordrecht, The Netherlands: Kluwer Academic Publishers.

KASPER, ROBERT T., PAUL C. DAVIS, & CRAIGE ROBERTS. 1999. An integrated approach to reference and presupposition resolution. In *Proceedings of the ACL'99 Workshop on the Relationship Between Discourse/Dialogue Structure and Reference*, College Park, Maryland.

LASCARIDES, ALEX, & NICHOLAS ASHER. 1993. Temporal interpretations, discourse relations, and commonsense entailment. *Linguistics and Philosophy* 16.437–493.

MACGREGOR, ROBERT M. 1991. Using a description classifier to enhance deductive inference. In *Proceedings of the Seventh IEEE Conference on AI Applications*, 141–147.

MANN, WILLIAM C., & SANDRA A. THOMPSON. 1987. Rhetorical structure theory: A theory of text organization. *ISI Reprint Series* ISI/RS-87-190.

ROBERTS, CRAIGE. 1995. Domain restriction in dynamic semantics. In *Quantification in Natural Languages*, ed. by A. Kratzer E. Bach, E. Jelinek & B.H. Partee. Kluwer Academic Publishers.

——. 1996. Information structure in discourse: Towards an integrated formal theory of pragmatics. In *OSU Working Papers in Linguistics, Vol 49: Papers in Semantics*, ed. by Jae-Hak Yoon & Andreas Kathol. (also available via the Internet at ftp://ling.ohio-state.edu/pub/roberts/infostructure.ps).

——. 1998. The place of centering in a general theory of anaphora resolution. In *Centering Theory in Discourse*, ed. by A.K. Joshi M.A. Walker & E.F. Prince. Oxford: Clarendon Press.

——. 2000 (to appear). Uniqueness in definite noun phrases. *Linguistics and Philosophy* . (also available via the Internet at ftp://ling.ohio-state.edu/pub/roberts/uniqueness.ps).

RUSSELL, BERTRAND. 1905. On denoting. *Mind* 66.479–493.

STALNAKER, ROBERT. 1979. Assertion. In *Syntax and Semantics, Volume 9*, ed. by Peter Cole.

SURI, LINDA Z., & KATHLEEN F. MCCOY. 1994. Raft/rapr and centering: A comparison and discussion of problems related to processing complex sentences. *Computational Linguistics* 20.301–317.

VAN DER SANDT, ROB. 1992. Presupposition projection as anaphora resolution. *Journal of Semantics* 9.333–377.

——. 1999. Domain restriction. In *Focus: Linguistic, Cognitive, and Computational Perspectives*, ed. by Peter Bosch & Rob van der Sandt. Cambridge, UK: Cambridge University Press.

ZEEVAT, HENK. 1989. A compositional approach to discourse representation theory. *Linguistics and Philosophy* 12.95–131.