Implementing the Binding and Accommodation Theory for Anaphora Resolution and Presupposition Projection

Johan Bos*
University of Edinburgh

Computational aspects of Van der Sandt's binding and accommodation theory (BAT) for presupposition projection and anaphora resolution are presented and discussed in this article. BAT is reformulated to meet requirements for computational implementation, which include operations on discourse representation structures (renaming and merging), the representation of presuppositions (allowing for selective binding and determining free and bound variables), and a formulation of the acceptability constraints imposed by BAT. An efficient presupposition resolution algorithm is presented, and several further improvements such as preferences for binding and accommodation are discussed and integrated in this algorithm. Finally, innovative use of first-order theorem provers to carry out consistency checking of discourse representations is investigated.

1. Introduction

The last decade has seen an increase of formal interest in combining what were previously thought as of being two distinct phenomena: anaphora and presupposition. In particular what I will refer to as **binding and accommodation theory** (BAT) (Van der Sandt and Geurts 1991; Van der Sandt 1992; Geurts 1999), in which presuppositional expressions are essentially analyzed as rich anaphora, played an important role here. Not only does this theory help us gain new insights into the nature of presuppositions, it also accounts for an impressive range of problems related to linguistic behavior of presuppositions.

In this paper I will put BAT in a computational perspective. The time is ripe to enter into this endeavor for two major reasons. First, because BAT is stipulated in the now well established and extensively formulated discourse representation theory (DRT) (Kamp and Reyle 1993), we have at our disposal a formalism covering a wide range of linguistic phenomena, including anaphora, plurals, tense, aspect, and scope ambiguities. Second, recent results in automated deduction, especially the performance of first-order theorem provers, open the doors for implementing a genuine inference component within a wider context of discourse processing (Blackburn et al. 2001). This is important, because in BAT, reasoning is required for correctly dealing with presuppositional phenomena.

I start by giving an overview of linguistic aspects of the problems introduced by presuppositional expressions, summarize DRT, and show how BAT accounts for presuppositional expressions (Section 2). I then explain what constitutes a proper representation for presuppositions, give examples of lexical entries for presupposition triggers within a compositional framework, and introduce formal tools required for

* Division of Informatics, 2 Buccleuch Place, Edinburgh EH8 9LW, Scotland UK. E-mail: jbos@cogsci.ed.ac.uk.

resolving presuppositions (Section 3). After presenting this formal machinery, I present an efficient version of the presupposition resolution algorithm in Section 4 and implement the various acceptability constraints imposed by BAT. Finally, in Section 5, I discuss implementational issues and report on the performance of the algorithm against a corpus of route instructions, and I investigate the use of general-purpose first-order theorem provers to carry out inference tasks imposed by BAT.

2. Preliminaries

To make this article self-contained, I will start by outlining the nature of presuppositions and what kind of problems they impose that need to be solved by any natural language understanding component performing discourse processing. Then I will present BAT and show how it accounts for these problems. Because BAT is presented as an extension of DRT, part of this section will be devoted to summarizing the main features of DRT.

2.1 Introducing Presuppositions

Presuppositions are those pieces of information that are taken for granted in a conversation or discourse. For instance, to make sense of (1),

(1) Vincent and Jules managed to clean the car.

we assume the existence of a car, two persons (named Jules and Vincent), and that the two persons found the car difficult to clean. These implications deviate from ordinary entailments. Note that example (1) entails that the car is clean, whereas example (2),

(2) If Vincent and Jules managed to clean the car, Jimmie would feel more comfortable.

does *not*, although example (2) includes some of the implications of example (1), namely, the existence of the two persons (Jules and Vincent) and the car, and that the two persons had difficulties in cleaning the car. The propositions that are still implied, even after embedding in the conditional, are called **presuppositions**. This is the crucial property of presuppositions: They are the result of implications that "survive" under negation and modal operators, in the antecedent of conditionals, and in questions (Van der Sandt 1992).

In English, most presuppositions in discourse are lexically driven. This means that there are certain lexical items that give rise to presuppositions, whereas others do not. In example (2), for instance, it is the definite article *the* that introduces the presupposition that there is a car and the implicative verb *to manage* that introduces the presupposition that Jules and Vincent had a hard time cleaning the car. Such expressions are called **presupposition triggers**, and in the examples that follow, I will underline ones relevant to each example.

Presuppositions are an important linguistic device in conversation, because when conveyed in utterances, they put constraints on the discourse context. An appropriate context for example (1) is, for instance, example (3):

(3) Jules and Vincent were driving in a car on their way to Jimmie. Due to an accident, it was completely covered with blood, and they had to clean it in short time before Bonnie (Jimmie's girlfriend) would find out. Finally, Vincent and Jules managed to clean the car.

Whereas the discourse in example (1), in isolation, presupposes the existence of a car, a context as set up in example (3) does not, for the trivial reason that there is a car introduced in the context itself. So (linguistic) contexts may contain information that "cancels" the presuppositions of a new contribution to the discourse or conversation. Therefore, terminological use in the literature on presupposition includes **part-time** or **elementary presuppositions**, to signify that presuppositions, once introduced by the trigger sentence, may be canceled by a context.

In contrast, consider the discourse in example (4), which is perceived as slightly odd. This perception of oddness is caused by the incompatibility between the context as set up by the first two sentences (namely, that there is a clean car) and the presupposition introduced by the third sentence (that Jules and Vincent had difficulties cleaning the car). The concept of incompatibility, or better, **inconsistency**, plays a central role in BAT, the theory of presuppositions in which I will base the resolution algorithm presented in this article.

(4) Jules and Vincent were looking after a car. The car was not dirty at all.
But Vincent and Jules managed to clean the car.

2.2 The Binding Problem, the Projection Problem, and Accommodation

By and large, there are three important themes related to presupposition: the binding problem, the projection problem, and accommodation. Almost all of the theoretical literature on presuppositions deals with these issues, and any computational account of presupposition has to say something about them. Let's first turn to the **presupposition binding problem**. An example like (5),

(5) A boxer nearly escaped from his apartment.

will clarify what comprises the binding problem. The trigger *his* induces the presupposition that a male individual has an apartment. However, it does not presuppose that just any male person has an apartment, nor that some boxer or other creature owns an apartment. It is the boxer who escaped who has an apartment. That is, the existentially quantified noun phrase *a boxer* ties together two types of information: ordinary asserted information (namely, that a boxer nearly escaped from an apartment) and presuppositional information (the apartment mentioned in the assertion belongs to the boxer mentioned in the assertion). As assertions and presuppositions obey different laws, it is no trivial matter to tie them together, and many accounts of presupposition have been shipwrecked on this rock (Van der Sandt 1992, pages 337–340).

The **presupposition projection problem** manifests itself in complex sentences. Presupposition triggers occurring in complex sentences, such as conditionals or disjunctive sentences, sometimes are projected onto the context, but sometimes disappear. For instance, example (6)

(6) If Mia dates Vincent, then her husband is out of town.

is a sentence presupposing that Mia has a husband. But the similarly constructed sentence in example (7) does not carry this presupposition:

(7) If Mia is married, then her husband is out of town.

This sentence does *not* presuppose that Mia has a husband. It is the bringing about of Mia's marital status in the antecedent of the conditional that neutralizes the presupposition of Mia's being married. Hence, in complex sentences there is no systematic way for dealing with presupposition triggers, as sometimes subparts of complex sentences carry presuppositions that are canceled in the main sentence.

Finally, let us consider **presuppositional accommodation**, as characterised by Karttunen (1974) and later formalized by Lewis (1979). I believe that accommodation plays a role in two related but different linguistic situations. The first of these is one in which presuppositions assert new information to the common ground without violating discourse coherency. The second situation is one best described as a hearer's discourse "repair strategy." Examples (8) and (9) illustrate the first type:

(8) Vincent informed his boss.

(9) Butch didn't realize there was a difference between a tummy and a potbelly.

The presuppositions conveyed by these utterances are that Vincent has a boss and that there is a difference between a tummy and a potbelly. Hearers have no problems accommodating these presuppositions into the common ground, even in cases in which the context includes no previous mention of them. Only if the discourse built up so far is incompatible with Vincent's having a boss (maybe he is a freelancer), then a hearer would probably refuse to accept example (8). But with the absence of information as to whether Vincent has a boss, the hearer adjusts his or her presuppositions to make sense of the new utterance or sentence. This is referred to as presuppositional accommodation.

Thus, presuppositions are, under certain circumstances, able to present new information to the discourse. However, the level of acceptance of accommodation differs considerably from context to context and according to the type of trigger used (Beaver 2002) and also depends on whether the hearer has access to context or not. Presuppositions triggered by genitive constructions (as in example (8)) and factives (as in example (9)) are known to accommodate easily. Most other presupposition triggers do not allow accommodation, because doing so would lead to incoherent discourse. Consider the following dialogue between Butch and his girlfriend after Butch has fought a match:

(10) *Fabian*: What about the man you fought?

Butch: Floyd retired too.

Butch's utterance in this dialogue presupposes that someone distinct from Floyd retired, a presupposition that is trivially true, as many people have retired already. But spoken without the knowledge that Butch ended his career, example (10) is odd, and a hearer will most likely start a clarification dialogue in such cases. However, example (10) is completely acceptable when one knows that Butch decided to retire after his fight with Floyd.

Nevertheless, although hearing example (10) in an ongoing dialogue without any mention of Butch's planning to retire will certainly raise some eyebrows, somebody who just joins an ongoing conversation and hears it will probably accommodate the associating presupposition, expecting that one of the topics addressed in this conversation was the retirement of somebody different from Floyd. This is when the other

role of presuppositional accommodation comes into play, constituting a situation in which hearers don't have access to the context and use accommodation as a repair strategy.

2.3 Discourse Representation Theory

Presupposition is a genuine discourse phenomenon. It should not come as a surprise that an adequate semantic theory for presuppositions would benefit from a formulation in a dynamic theory of meaning. Indeed, BAT is set in DRT (Kamp 1981; Kamp and Reyle 1993; Van Eijck and Kamp 1997), and because it heavily depends on it, I will briefly summarize the most prominent features of DRT here.

DRT is one of several formal semantic frameworks designed to deal with the problems related to discourse anaphora, but it is certainly unrivaled with respect to its impressive coverage of linguistic phenomena. Alternative formalisms for discourse semantics are file change semantics (Heim 1982) and dynamic predicate logic (Groenendijk and Stokhof 1991). The latter uses the syntax of ordinary first-order predicate logic, but with a different "context change potential" semantics, allowing existential quantifiers to bind variables outside their syntactic scope.

The linguistic phenomena that led to the development of dynamic theories such as DRT were mainly centered on the problems introduced by anaphora and indefinite noun phrases. Because anaphora are able to operate on an intersentential level, the traditional method of assigning closed formulas to sentences caused problems for discourse processing, and one had to resort to a number of ad hoc techniques for constructing the first-order formulas for natural language discourses. An appropriate first-order logic formula for the sentence *A woman snorts* would be $\exists x(\text{WOMAN}(x) \wedge \text{SNORT}(x))$, but if one continues the discourse with *She collapses*, there is a need to alter the formula into something of the form: $\exists x(\text{WOMAN}(x) \wedge \text{SNORT}(x) \wedge \text{COLLAPSE}(x))$. In other words, one has to extend the scope of the existential quantifier introduced in the translation of the first sentence to cover elements introduced in the second sentence. The so-called donkey sentences of Geach (donkeys and farmers were the main characters in the example sentences of Geach, which led to increased study of indefinite noun phrases and pronouns within formal semantics) caused similar compositionality problems. Although indefinite noun phrases normally invoke existential quantification, a proper first-order translation of *Every woman that gets a foot massage enjoys it* would result in a formula in which the variable introduced by the pronoun *it* is bound by the universal quantifier stemming from *a foot massage*, as shown by example (11):

(11) $\forall x \forall y(\text{WOMAN}(x) \wedge \text{FOOT-MASSAGE}(y) \wedge \text{GET}(x, y) \rightarrow \text{ENJOY}(x, y))$

DRT deals with these problems by introducing an intermediate level of semantic representation: discourse representation structures (DRSs). DRSs are pairwise structures consisting of a set of **discourse referents**, which stand for the entities that are introduced in the discourse, and a set of **conditions**, which describe the properties of these entities. Discourse referents function like quantifiers, in that they are able to bind variables appearing in DRS-conditions. However, the quantificational force of discourse referents depends on their structural embedding. DRSs are recursive structures, and DRSs embedded in the antecedent of an implicational condition give universal quantification to their discourse referents, whereas all other contexts assign existential quantification (the translation from DRSs to formulas of first-order logic given in Section 3 illustrates this nicely).

Traditionally, a DRS is presented as a boxlike structure, with discourse referents in the top part and conditions in the lower part of the box. The DRS for the example given earlier is shown in example (12).

(12) A woman snorts. She collapses.

x y
WOMAN(x)
SNORT(x)
y=x
COLLAPSE(y)

Here x and y are discourse referents for *a woman* and *she*, respectively. The anaphoric link between *she* and *a woman* is established by the condition $y = x$, and illustrates the role of discourse referents: They introduce discourse entities to which pronouns potentially can refer. In other words, discourse referents are candidate antecedents for future anaphoric reference.

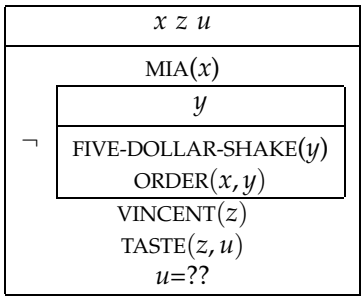
The key idea underlying DRT is that discourse referents appearing in embedded DRSs are not available as antecedents for pronouns. The internal structure of DRSs plays a central role in determining the possibility of anaphoric links between pronouns and their potential antecedents. Indefinite noun phrases always introduce their discourse referents locally[1] and hence are not **accessible** from outside a negation or implication. This is shown in the following examples, in which pronouns marked with an asterisk have no proper anaphoric antecedent (narrow-scope interpretation of the indefinite noun phrases is assumed in the examples, because in certain circumstances a wide-scope interpretation of indefinite noun phrases allows anaphoric links):

(13) Butch has a valuable watch. He keeps it in his apartment.
Butch has no valuable watch. He keeps it* in his apartment.
If Butch has a valuable watch, he will take care of it. He keeps it* in his apartment.
Mia ordered a five dollar shake. Vincent tasted it.
Mia didn't order a five dollar shake. Vincent tasted it*.

In the utterances of example (13), the discourse referents for *a valuable watch* and *a five dollar shake* are introduced in subordinated DRSs, excluding anaphoric links to pronouns in subsequent sentences. The DRS in example (14) shows how DRT deals with these observations: Because the discourse referent y introduced for *a five dollar shake* is part of an embedded DRS (introduced by negation), it is not accessible for u, the referent introduced for the pronoun *it*:

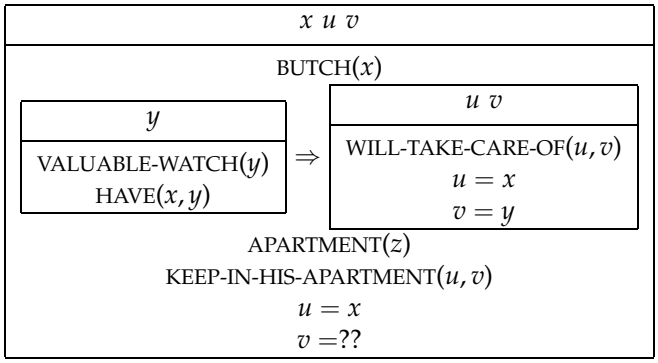
1 I am disregarding wide-scope or specific readings here.

(14) Mia didn't order a five dollar shake. Vincent tasted it*.



In other words, negation blocks anaphoric links. Similar anaphoric behavior is shown by disjunctive clauses (Kamp and Reyle 1993, page 185) and implicational sentences. The latter introduce DRS-conditions of the form $B \Rightarrow B'$, where discourse referents declared in B are accessible from B'. Example (15) illustrates this:

(15) If Butch has a valuable watch, he will take care of it. He keeps it* in his apartment.



The accessibility relation in DRT governs possible links between anaphoric expressions and their potential antecedents. It is defined on the structure of DRSs, which is normally stated in terms of **subordination**, a transitive relation. A DRS B subordinates a DRS B' if B' appears as a condition of B as argument of a negation, disjunction, or antecedent of a conditional, or if they form a DRS-condition of the form $B \Rightarrow B'$. I will merely make use of the term **accessibility path**, a list of DRSs, ordered by DRS subordination, to express accessible discourse referents for anaphoric expressions.

Finally, let us consider proper names. Proper names, once introduced in discourse, always seem available for future anaphoric reference, and so do first- and second-person pronouns. Therefore, in DRT, their discourse referents are introduced at the global level of DRS, even in cases in which they are realized as part of a subordinated context, as for the proper name *Butch* in Example (15).

2.4 Binding and Accommodation Theory

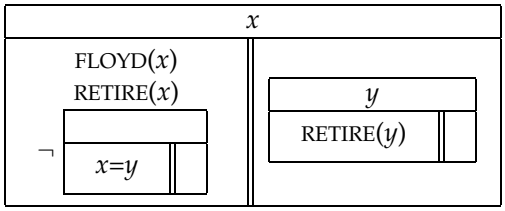
In BAT, presuppositions obey similar principles as anaphora do: Presuppositions are resolved to an antecedent, and moreover, resolution of presuppositions is constrained by discourse structure in the same way DRT defines accessibility of antecedents for ordinary pronouns. According to Van der Sandt, the main difference between presup-

positions and pronouns is that the former have descriptive content, which the latter lack.

To integrate presuppositions into the representation of discourse, Van der Sandt introduces sentence-DRSs, DRSs that are defined as a triple of a set of discourse referents, a set of conditions, and a (possibly empty) set of DRSs (Van der Sandt 1992, page 354). This set is referred to as the **A-structure**, the set of anaphoric DRSs. This definition is recursive, so it allows embedded anaphoric structures inside other anaphoric structures. Sentence-DRSs are preliminary representations, and no model-theoretic interpretation is defined for these structures. A completely resolved sentence-DRS (hence all of its A-structures are empty) is identical to an ordinary DRS, and is called a **proper DRS**.

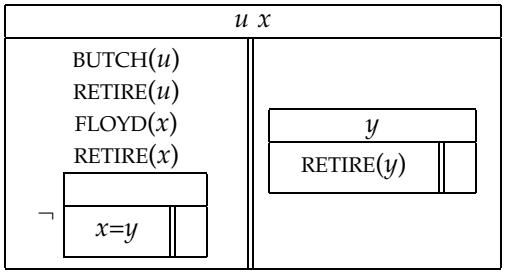
Throughout this section and the next one, I will depict sentence-DRSs as boxes divided into three parts, with the discourse referents at the top, the conditions at the bottom left, and the A-structure at the bottom right. Given this definition and notation of sentence-DRSs, let us consider some examples of presupposition triggers and explain BAT. For instance, Butch's utterance in example (10) will be associated with the following DRS:

(16) Floyd retired too.



This sentence-DRS (let us call it *B*) contains a presuppositional expression, namely, the one triggered by the particle *too*. Given a context associated with another DRS *C*, *B* is resolved by merging *C* with *B* (merging here meaning taking the unions of the sets of discourse referents, conditions, and A-structures), followed by resolving all members of the A-structure to accessible discourse referents. The resolution process, as Van der Sandt (1992) formulates it in his article, is here only informally described, by means of giving an example. Assume that in the previous context DRS *C* just describes that Butch retired; then merging *C* with *B* yields a DRS in which the sets of discourse referents, conditions, and A-structures from both DRSs are unified:

(17) Butch retired. Floyd retired too.



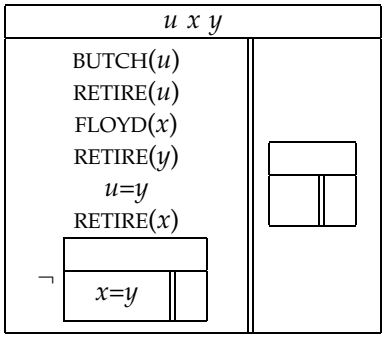
In principle, sentence-DRSs can have any number of anaphoric entities in their A-structures. In our example, the A-structure consists of one DRS. Now, resolving

elements of the A-structure is subject to either binding or accommodation. Let us first look at binding. Members of the A-structure can be bound by an accessible antecedent. Note that elements of the A-structure are resolved only if their A-structures are empty. Nested anaphoric structures are thus resolved by resolving the most deeply nested anaphoric DRS first. Binding proceeds by identifying the discourse referents with established referents and transferring their associating conditions to the binding site (Van der Sandt 1992, page 357).

Now let us consider accommodation. This option resolves a member of the A-structure by creating an antecedent and becomes available if no suitable antecedent for an anaphoric expression can be found. Accommodation is accomplished by transferring the anaphoric referent, including its conditions, to the accommodation site (which, as with binding, must be an accessible DRS).

Going back to our example, we resolve the anaphoric DRS by appealing to the binding option. The anaphoric discourse referent x is identified by the accessible discourse referent y, and all the conditions belonging to y are transferred to the binding site. This gives us the following DRS:

(18)

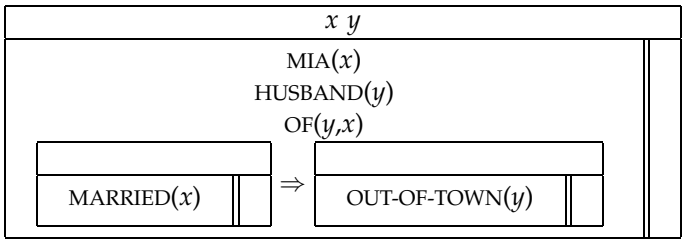


This DRS contains an empty A-structure and hence is a proper DRS. Note that accommodation would yield a similar DRS, with the only difference being the lack of the equality condition. Further, according to BAT, accommodation can be applied to any level of discourse, and to indicate these possibilities for accommodation at various levels, the concepts of **global**, **intermediate**, and **local accommodation** are used, corresponding to the various landing sites for a presupposition. Hence, in more complicated examples, binding and accommodation give rise to several possibilities, and Van der Sandt's algorithm will yield a large number of potential solutions. These solutions are filtered by a number of constraints, some of which I will introduce below, when I discuss how BAT deals with the binding problem and presupposition projection. These **acceptability constraints** will be discussed and implemented in detail in Section 4.3.

BAT tackles the projection problem by imposing several constraints with respect to consistency and informativeness. The local versions of these constraints impose the requirement that sub-DRSs (DRSs that are embedded in conditions of DRSs) be informative and consistent with respect to the information that subordinates them. These acceptability constraints can be viewed as the pragmatic component of BAT, because there is a clear correspondence to the Gricean conversational maxims ("do not say what you believe to be false," "do not say things that are irrelevant," and so on). As an illustration of local informativeness, consider example (19). Global accommodation of *her husband* is impossible here, because the sentence does not presuppose that Mia is married. BAT predicts this, because example (19) violates local informativeness: The antecedent DRS of the conditional does not contain new information with respect to

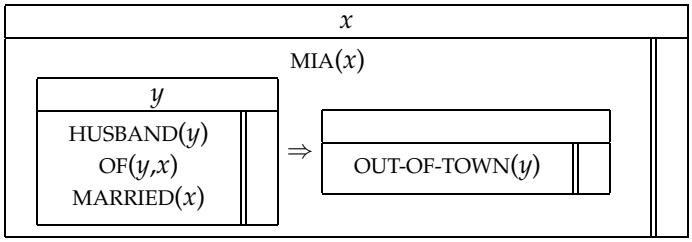
the global DRS (using background knowledge that women who have husbands are married).

(19) If Mia is married then her husband is out of town. (*global accommodation*)



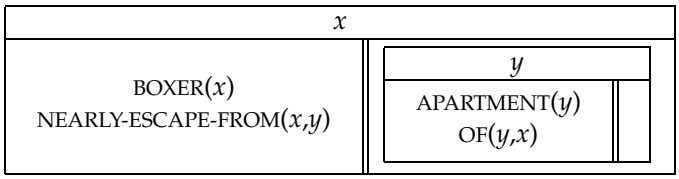
Whereas the acceptability constraints rule out example (19) as a possible interpretation, there are other possible interpretations because of the availability of other accommodation sites for *Mia's husband*. One of these possibilities, intermediate accommodation, is shown in example (20) (a third possibility, not shown here, is local accommodation, in which the consequence of the implicational condition is the landing site for the presupposition). Intermediate accommodation does not violate local informativeness here: All the sub-DRSs are informative with respect to their global context.

(20) If Mia is married then her husband is out of town. (*intermediate accommodation*)



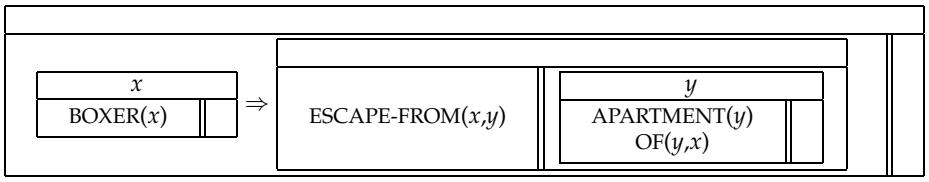
BAT masters the binding problem by combining presupposition and assertion into one semantic representation that allows shared use of discourse referents. This is shown by the DRS for example (21) (in which *his* has already been resolved to the boxer, and *his apartment* is still subject to resolution):

(21) A boxer escaped from his apartment.



The shared representation solves the binding problem, but as a direct consequence, accommodation might result in ill-formed DRSs. To overcome this, there is a further acceptability constraint on resolution, known as the **free variable trap**, which imposes the requirement that DRSs resulting from the resolution algorithm not contain free variables. This prevents global accommodation of *his apartment* in example (22).

(22) Every boxer escaped from his apartment.



To sum up, BAT gives us three important insights with respect to resolving presuppositions. First of all, the resolution algorithm used in BAT is nondeterministic, so several ways of interpreting presuppositions are possible. In most cases, there is a clear preference for certain of these interpretations, but in others, presuppositions present genuine ambiguities. Second, presuppositional binding involves *compatibility* with, rather than entailment of, its antecedent. This "matching" feature enables presuppositions to add information to the discourse, even if they are bound to an antecedent rather than accommodated. Finally, by treating presupposition on a par with anaphora, BAT kills two birds with one stone. The resolution algorithm will cover both anaphora and presuppositions, and additional resources for resolution (such as mechanisms for preference ranking) will help in dealing with both anaphora and presupposition.

3. Representing Discourse and Presuppositions

This section is concerned with proper representations for discourse and presupposition, with the interpretation of discourse (aiming to being able to perform inferences), and further provides tools necessary for discourse processing and presupposition resolution. I will start with some formalities and define the syntax of standard DRSs and the syntax of α-DRSs (DRSs that contain unresolved presuppositions). I will argue that the representation for sentence-DRSs, as originally introduced for presuppositions in BAT, is insufficient for several interpretation tasks and introduce a new format.

3.1 Representing and Interpreting Discourse

DRSs capture the semantic content of a discourse. They form the medium for discourse understanding, because they come with a model-theoretical interpretation. The interpretation given here is one via a translation to first-order logic. This is advantageous from a practical and computational perspective, because one can use automated theorem provers for first-order logic to implement some of the acceptability constraints imposed by BAT or indeed carry out other kinds of inferences not related to anaphora resolution and presupposition accommodation.

The syntax of DRSs and DRS-conditions is defined by simultaneuous recursion, with respect to a set of first-order variables and a vocabulary describing the predicate symbols and their respective arities.

Definition

The syntax of DRSs and DRS-conditions is defined according to the following seven clauses:

1. If $\{x_1 \dots x_n\}$ is a finite set of variables, and $\{\gamma_1 \dots \gamma_m\}$ is a finite set of DRS-conditions, then the ordered pair $\langle \{x_1 \dots x_n\}, \{\gamma_1 \dots \gamma_m\} \rangle$ is a basic DRS.

2. If R is a relation symbol for an n-place predicate and $x_1 \dots x_n$ are variables, then $R(x_1, \dots, x_n)$ is a basic DRS-condition.
3. If x_1 and x_2 are variables, then $x_1 = x_2$ is a basic DRS-condition.
4. Every basic DRS-condition is a DRS-condition.
5. If B is a DRS, then $\neg B$, $\Box B$, and $\Diamond B$ are DRS-conditions.
6. If B_1 and B_2 are DRSs, then $B_1 \vee B_2$, and $B_1 \Rightarrow B_2$ are DRS-conditions.
7. If x is a variable and B is a DRS, then $x : B$ is a DRS-condition.

Given a DRS $B = \langle D, C \rangle$, D is called the domain of B, members of C are the conditions of B, and members of D are called B's discourse referents. Clause 1 of the definition defines DRSs in the standard way. The basic conditions (clauses 2–3) are defined just as in standard DRT. Clause 5 introduces negation and the modal operators, and clause 6 disjunction and implication. Clause 7 is nonstandard; it introduces a modal operator that explicitly associates variables ranging over possible worlds with DRSs. It is therefore related to constructs used in hybrid logics (Blackburn 2000). We will use it in our fragment of English to represent sentential complements.

DRSs are interpreted in an indirect manner, with the help of a translation function that maps DRSs to first-order formulas (under the same vocabulary of predicate symbols and with respect to the same set of variables). This translation is implemented as the function $(.,.)^{fo}$, from first-order variables (ranging over possible worlds) and DRSs to ordinary first-order formula syntax. The complete translation is shown in the following definition.

Definition

The translation $(.,.)^{fo}$ from DRSs to first-order logic is defined according to the following nine clauses:

1. $(w, \boxed{\begin{array}{c} x_1 \dots x_n \\ \hline \gamma_1 \dots \gamma_m \end{array}})^{fo} \stackrel{\text{def}}{=} \exists x_1 \cdots \exists x_n((w, \gamma_1)^{fo} \wedge \cdots \wedge (w, \gamma_m)^{fo})$
2. $(w, R(x_1, \dots, x_n))^{fo} \stackrel{\text{def}}{=} R(w, x_1, \dots, x_n)$
3. $(w, x_1 = x_2)^{fo} \stackrel{\text{def}}{=} x_1 = x_2$
4. $(w, \neg B)^{fo} \stackrel{\text{def}}{=} \neg(w, B)^{fo}$
5. $(w, B_1 \vee B_2)^{fo} \stackrel{\text{def}}{=} (w, B_1)^{fo} \vee (w, B_2)^{fo}$
6. $(w, \boxed{\begin{array}{c} x_1 \dots x_n \\ \hline \gamma_1 \dots \gamma_m \end{array}} \Rightarrow B)^{fo} \stackrel{\text{def}}{=} \forall x_1 \cdots \forall x_n(((w, \gamma_1)^{fo} \wedge \cdots \wedge (w, \gamma_m)^{fo}) \rightarrow (w, B)^{fo})$
7. $(w, \Diamond B)^{fo} \stackrel{\text{def}}{=} \exists v(R(w, v) \wedge (v, B)^{fo})$
8. $(w, \Box B)^{fo} \stackrel{\text{def}}{=} \forall v(R(w, v) \rightarrow (v, B)^{fo})$
9. $(w, v : B)^{fo} \stackrel{\text{def}}{=} (R(w, v) \wedge (v, B)^{fo})$

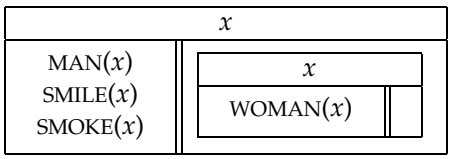
This translation from DRSs to first-order logic is based on the one given in Kamp and Reyle (1993) extended with Moore's proposal for modal operators (Moore 1980). It behaves linearly on the size of the input, so the computational overhead is kept low. I will use it to implement the acceptability constraints imposed by BAT on presupposition resolution that require inference, to wit, the check for consistency and informativeness.

3.2 Representing Presuppositions

What is a proper representation for elementary presuppositions? There seems to be common agreement, in most of the accounts in presupposition theory, that presuppositions represent expressions of propositional type. Hence, to use a DRS to represent a single presupposition seems a natural choice.

But there are further issues that play a role in deciding a suitable representation for presuppositions. Two operations on DRSs used in BAT are merge reduction and presuppositional binding,[2] and both require a precise definition of free and bound variables. However, sentence-DRSs allow "ambiguous" bindings. Consider, for instance, the DRS in example (23) (again I will underline the relevant presupposition triggers in the following examples):

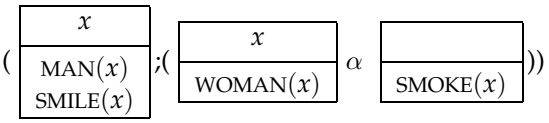
(23) A man smiles. <u>The woman</u> smokes.



This sentence-DRS contains an A-structure with a single DRS. It is unclear whether the occurrence of variable x in the condition SMOKE(x) is bound by the discourse referent x in the outermost DRS or by the discourse referent declared in the DRS within the A-structure. Following the definition of sentence-DRSs, the discourse referent x in the A-structure does not in fact bind the occurrences of x in the main DRS. Furthermore, given the fact that A-structures can host more than one DRS, situations with ambiguous bindings might appear easily.

These unintuitive and ambiguous bindings are unpleasant and force one to reconsider representing unresolved anaphoric expressions in DRT. The representation that I prefer uses a new operator, α, combining two DRSs to form a new DRS. This disallows ambiguous bindings while keeping the same expressive power:

(24) A man smiles. <u>The woman</u> smokes.

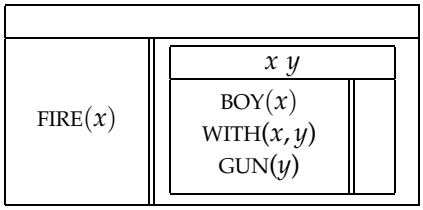


2 I am considering presuppositional binding in which, after identifying the antecedent discourse referent, all bound occurrences of the anaphoric discourse referents are replaced by the variable name of the antecedent referent. This operation is preferred to adding an equality condition between the antecedent and anaphoric referent to the DRS, for two reasons: It decreases the search space for finding antecedents during subsequent instances of presupposition resolution, and it makes the inference problems derived from these DRSs less difficult.

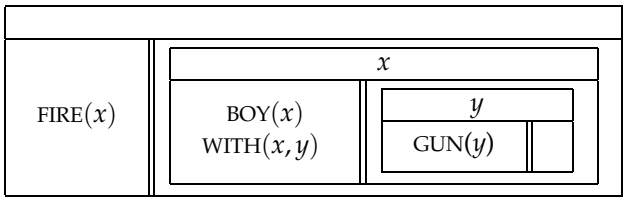
I believe that the representation in example (24) is more intuitive as well, because it is presupposition that comes first in an utterance. The α operator reflects this, because its left argument is the presuppositional part, and its right argument the assertive part. Like A-structures, the α operator allows recursion and therefore nested presuppositions.

I would like to address another representational issue here. In Van der Sandt's (1992) original formulation of BAT, all discourse referents appearing in the domain of a DRS in the A-structure are anaphoric. This leads to unexpected behavior, as exemplified by the sentence-DRSs for the two sentences in example (25):

(25) The boy with a gun fires.



The boy with the gun fires.

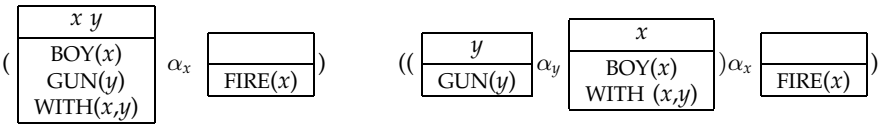


The sentences in (25) contain noun phrases with restricted relative clauses. In the first example the relative clause contains an indefinite noun phrase *a gun*, whereas in the almost identical second example, it contains a definite (hence presuppositional) noun phrase. These two sentence-DRSs do not, however, reflect the difference in meaning of the utterances they represent. The indefinite noun phrase *a gun* gets an anaphoric interpretation, because it is part of the A-structure. The anaphoric potential of the two utterances, according to the sentence-DRSs for the two utterances in example (25), is almost identical, the only difference being that the DRS for the second utterance allows binding or accommodation on two different levels (instead of one level) of discourse structure. Summing up, A-structures do not allow for **selective binding**, with the unwanted side effect that indefinite noun phrases are turned into definite ones.

To allow for selective binding, I introduce the notion of **principal anaphoric referent**. The operator α is indexed with the principal anaphoric referent to indicate which discourse referent of a presuppositional DRS is anaphoric. In fact, I assume that each presupposition trigger has a unique principal anaphoric referent.[3] Now let us consider the DRSs for the same utterances in (25) in the new formats:

3 Although this seems a small and innocent adjustment to the representation of elementary presuppositions, it has great impact on our understanding what characterizes presuppositions. Under this view, presuppositions are not merely DRSs, but they are DRSs plus an additional pointer, in the form of a distinguished discourse referent, to a context.

(26) The boy with a gun fires. The boy with the gun fires.



Although the entire DRS of the left-hand side of the α-operator is said to be presuppositional, resolution will affect only the principal discourse referent. For *The boy with a gun*, this is x, and only x is identified with an antecedent discourse referent; y, introduced for the indefinite noun phrase *a gun*, is not treated as anaphoric. This is in contrast to the DRS for *The boy with the gun*, which contains nested α-DRSs. In short, α allows selective binding, whereas Van der Sandt's A-structures are based on unselective binding.

I will refer to this new format for DRSs, encoding unresolved anaphoric DRSs, as **α-DRSs**. Like sentence-DRSs, α-DRSs are intermediate representations and have no interpretation. The best way to view them is as underspecified representations encoding the ambiguities of anaphoric expressions in a compact manner. The syntax of α-DRSs is defined as follows:

Definition
The syntax of α-DRSs is defined on the basis of the following four clauses:

1. If $\{x_1 \ldots x_n\}$ is a finite ordered set of variables and $\{\gamma_1 \ldots \gamma_m\}$ is a finite ordered set of α-DRS-conditions, then the ordered pair $\langle\{x_1 \ldots x_n\}, \{\gamma_1 \ldots \gamma_m\}\rangle$ is a basic α-DRS.
2. Every basic α-DRS is an α-DRS.
3. If B_1 and B_2 are α-DRSs, then so is $(B_1;B_2)$.
4. If B_1 and B_2 are α-DRSs, and x is a discourse referent declared in the domain of B_1, then $(B_1\ \alpha_x\ B_2)$ is an α-DRS.

Note that in clause 1 of the definition, ordered sets are used rather then plain sets for discourse referents and DRS-conditions; this will make definition of the resolution algorithm easier. DRS merging (clause 4) is used in many alternative formulations of DRT (Muskens 1996; Van Eijck and Kamp 1997; Kuschert 1999). The merge employed here, ";", is adopted from Muskens and behaves semantically the same as dynamic conjunction in dynamic predicate logic (Groenendijk and Stokhof 1991).

The syntax of α-DRS-conditions is defined as follows:

Definition
The syntax of α-DRS-conditions is defined according to the following four clauses:

1. Every basic DRS-condition is an α-DRS-condition.
2. If B is an α-DRS, then $\neg B$, $\Box B$, and $\Diamond B$ are α-DRS-conditions.
3. If B_1 and B_2 are α-DRSs, then $B_1 \vee B_2$ and $B_1 \Rightarrow B_2$ are α-DRS-conditions.
4. If x is a variable and B is an α-DRS, then $x : B$ is an α-DRS-condition.

So the syntax for α-DRSs subsumes the syntax of DRSs. I will refer to α-DRSs that contain no DRSs of the form (B α_i B') as **presupposition-free**. All other α-DRSs are referred to as **presupposition-containing**. (The same terminology will be used for α-DRS-conditions.) It is easy to show that presupposition-free α-DRSs are proper DRSs.

In Table 1, I give several examples of lexical entries of presupposition triggers, assuming a compositional semantics in the Montagovian tradition based on Muskens's (1996) compositional DRT (Muskens, 1996). The syntactic categories used in the table are N (noun), DET (determiner), NP (noun phrase), ADJ (adjective), and V (verb). Further, p and q are used to denote variables ranging over properties, and s to denote variables ranging over propositions. The table lists various kinds of presupposition triggers, including the definite determiner, a proper name (*Mia*), a factive verb (*to realize*), a sortally restricted predicate (*bachelor*), and the iterative adjective *other*.

3.3 Operations on Discourse Representations

In this section I will formulate two operations on discourse representations required for presupposition resolution: renaming and merging. Furthermore, I will introduce the concepts of free and bound variables in discourse representations, because we need to implement the free-variable constraint. This will also give us a better understanding of the problems that arise in renaming and merging.

A **free variable** in this context is an occurrence of a variable in a DRS that is not declared as a discourse referent in the domain of the immediate DRS in which it occurs or in the domain of a superordinated DRS; all other variables are **bound**. To be more precise, I define two functions, FREE and BOUND, that compute the free and bound variables for DRSs and DRS-conditions:

Definition

Free and bound variables in α-DRSs are defined according to the following nine clauses:

1. $\text{FREE}(\langle\{x_1 \ldots x_n\}, \{\gamma_1 \ldots \gamma_m\}\rangle) = (\text{FREE}(\gamma_1) \cup \ldots \cup \text{FREE}(\gamma_m))\backslash\{x_1 \ldots x_n\}$;
   $\text{BOUND}(\langle\{x_1 \ldots x_n\}, \{\gamma_1 \ldots \gamma_m\}\rangle) = \{x_1 \ldots x_n\}$
2. $\text{FREE}(B\alpha B') = (\text{FREE}(B) \cup \text{FREE}(B'))\backslash\text{BOUND}(B)$;
   $\text{BOUND}(B\alpha B') = \text{BOUND}(B) \cup \text{BOUND}(B')$
3. $\text{FREE}(B; B') = (\text{FREE}(B) \cup \text{FREE}(B'))\backslash\text{BOUND}(B)$;
   $\text{BOUND}(B; B') = \text{BOUND}(B) \cup \text{BOUND}(B')$
4. $\text{FREE}(R(x_1, \ldots, x_n)) = \{x_1, \ldots, x_n\}$;
   $\text{BOUND}(R(x_1, \ldots, x_n)) = \emptyset$
5. $\text{FREE}(x{=}y) = \{x, y\}$;
   $\text{BOUND}(x{=}y) = \emptyset$
6. $\text{FREE}(B \Rightarrow B') = (\text{FREE}(B) \cup \text{FREE}(B'))\backslash\text{BOUND}(B)$;
   $\text{BOUND}(B \Rightarrow B') = \emptyset$
7. $\text{FREE}(B \vee B') = \text{FREE}(B') \cup \text{FREE}(B)$;
   $\text{BOUND}(B \vee B') = \emptyset$
8. $\text{FREE}(\gamma) = \text{FREE}(B)$ if γ is of the form $\neg B$, $\Box B$, $\Diamond B$;
   $\text{BOUND}(\gamma) = \emptyset$ if γ is of the form $\neg B$, $\Box B$, $\Diamond B$
9. $\text{FREE}(x : B) = \{x\} \cup \text{FREE}(B)$;
   $\text{BOUND}(x : B) = \emptyset$

Table 1
Lexical entries for various presupposition triggers and presupposition-free expressions.

Category	Lexical Semantics	Example
N	$\lambda x.$ [; BOXER(x)]	boxer
N	$\lambda x.($ [y ; MALE(y), y=x] α_y [; ¬ [; MARRIED(x)]] $)$	bachelor
DET	$\lambda p.\lambda q.(($ [x ;] $;p(x));q(x))$	a
DET	$\lambda p.\lambda q.$ [; ([x ;] $;p(x)) \Rightarrow q(x)$]	every
DET	$\lambda p.\lambda q.(($ [x ;] $;p(x))\alpha_x q(x))$	the
DET	$\lambda p.\lambda q.(($ [x ; FEMALE(x)] $\alpha_x(($ [y ; OF(y,x)] $;p(y))\alpha_y q(y))))$	her
NP	$\lambda p.($ [x ; BODY(x)] $;p(x))$	somebody
NP	$\lambda p.($ [x ; MIA(x)] $\alpha_x p(x))$	Mia
NP	$\lambda p.($ [x ; FEMALE(x)] $\alpha_x p(x))$	she
ADJ	$\lambda p.\lambda x.($ [; BLUE(x)] $;p(x))$	blue
ADJ	$\lambda p.\lambda x.(($ [y ;] $;p(y))\alpha_y ($ [; ¬ [; x=y]] $;p(x)))$	other
V	$\lambda s.\lambda x.$ [b ; b:s, BELIEVE(x,b)]	believes
V	$\lambda s.\lambda x.($ [b ; b:s] α_b [; REALIZE(x,b)] $)$	realizes

For renaming variables in DRSs, a (total) function r is used whose arguments are provided by an auxiliary (total) function σ that maps a variable to a stringed sequence of a copy of itself. The use of σ makes the values r assigns to a variable x sensitive to the number of occurrences of x in a DRS: $\sigma(x) = x$ if x has not appeared (bound) so far in a DRS, $\sigma(x) = xx$ if x has appeared (bound) once, xxx if x has appeared (bound) twice, and so on. Putting it differently, σ is a counter. It counts the number of occurrences of declarations of discourse referents in a DRS. On the basis of that count, the renaming function r maps a variable onto a new, previously unused occurrence.

The renaming function r maps a sequence of variables to a new, fresh discourse variable. The function r is injective, meaning that distinct argument sequences to r produce distinct values. In other words, the values r assigns to x, y, z, xx, yy, zz, xxx, yyy, zzz, and so on are all different. In sum, with the help of r and σ, we make explicit that two discourse referents declared under the same name in two different domains in a DRS get two different names. The definition of renaming in DRS (where for a set of discourse referents U, UPDATE(σ, U, σ') holds just if $\forall x(\sigma'(x) = \sigma(x) \leftrightarrow x \notin U)$ and $\forall x(\sigma'(x) = \sigma(x)x \leftrightarrow x \in U)$ hold) is as follows:

Definition
DRS renaming is defined according to the following eight clauses:

1. RENAME$(\sigma, \langle\{x_1 \ldots x_n\}, \{C_1 \ldots C_m\}\rangle) =$
   $\langle\sigma', \langle\{r(\sigma'(x_1)) \ldots r(\sigma'(x_n))\}, \{C'_1 \ldots C'_m\}\rangle\rangle$
   if UPDATE$(\sigma, \{x_1, \ldots, x_n\}, \sigma')$ and RENAME$(\sigma', C_i) = \langle\sigma'', C'_i\rangle$ for some σ''
2. RENAME$(\sigma, (B_1; B_2)) = \langle\sigma'', (B'_1; B'_2)\rangle$
   if RENAME$(\sigma, B_1) = \langle\sigma', B'_1\rangle$ and RENAME$(\sigma', B_2) = \langle\sigma'', B'_2\rangle$
3. RENAME$(\sigma,(B_1\ \alpha_x\ B_2) = \langle\sigma'',(B'_1\ \alpha_{r(\sigma'(x))}B'_2)\rangle$
   if RENAME$(\sigma, B_1) = \langle\sigma', B'_1\rangle$ and RENAME$(\sigma', B_2) = \langle\sigma'', B'_2\rangle$
4. RENAME$(\sigma, R(x_1, \ldots, x_n)) = \langle\sigma, R(r(\sigma(x_1)), \ldots, r(\sigma(x_n)))\rangle$
5. RENAME$(\sigma, x = y) = \langle\sigma, r(\sigma(x)) = r(\sigma(y))\rangle$
6. RENAME$(\sigma, B_1 \Rightarrow B_2) = \langle\sigma, B'_1 \Rightarrow B'_2\rangle$
   if RENAME$(\sigma, B_1) = \langle\sigma', B'_1\rangle$ and RENAME$(\sigma', B_2) = \langle\sigma'', B'_2\rangle$ for some σ''
7. RENAME$(\sigma, B_1 \vee B_2) = \langle\sigma, B'_1 \vee B'_2\rangle$
   if RENAME$(\sigma, B_1) = \langle\sigma', B'_1\rangle$ and RENAME$(\sigma, B_2) = \langle\sigma'', B'_2\rangle$ for some σ' and σ''
8. RENAME$(\sigma, \neg B) = \langle\sigma, \neg B'\rangle$ if RENAME$(\sigma, B) = \langle\sigma', B'\rangle$ for some σ'

The function RENAME maps an ordered pair consisting of a σ-function and a DRS to a new σ-function and the translated DRS. Clauses 1–3 of the definition handle DRSs, clauses 4–8 handle the DRS-conditions. With respect to variables not declared in a universe of a DRS, σ remains unchanged. For variables that are declared as discourse referents in a DRS, σ increases the values for these variables by one. Clauses 2, 3, and 6 show that the σ-function produced by the first DRS is passed through to the second DRS, following the definition of DRS-subordination.

With the renaming function at our disposal, we now can define **pure DRSs** as DRSs that have undergone renaming. The conversion of a DRSs potentially changes its **discourse meaning**, and this is actually the key function of renaming: maximizing the

context change potential of a DRS. To preserve the **logical meaning** while renaming, we need to put restrictions on the use of renaming. As DRSs can bind variables outside their scope (for instance, through use of the merge operator), applying the renaming procedure only to a DRS B that is actually part of another DRS B' would possibly affect the logical meaning of a DRS. Therefore, only *complete* DRSs should be renamed; that is, if a sub-DRSs is renamed, any DRS that superordinates it must be renamed too.

Let me now relate this to practical discourse processing. Assume that processing a text proceeds in an incremental way, starting with processing the first sentence, until a DRS is eventually derived for the entire text. At some point during this process, after completing the analysis of a sentence, part of the text is translated into a DRS. At this stage of processing, the obtained DRS is complete (it is not part of another DRS, as the rest of the text is still unanalyzed), and it can be renamed without changing its logical meaning, while maximizing its context change potential. The rest of the discourse is then processed with respect to the renamed DRS.

Hence, given a pure DRS, we can replace a DRS $(B;B')$ with a new DRS that is constructed by taking the unions of the discourse referents and the conditions of B and B', respectively. We will specify the merging of DRSs with the help of a function MERGE from DRSs to DRSs. This function is defined recursively:

Definition
Merge reduction for pure DRSs is defined according to the following two clauses:

1. $\text{MERGE}(\langle D_B, C_B \rangle) = \langle D_B, C_B \rangle$
2. $\text{MERGE}((B;B')) = \langle D_{\text{MERGE}(B)} \cup D_{\text{MERGE}(B')}, C_{\text{MERGE}(B)} \cup C_{\text{MERGE}(B')} \rangle$

Carrying out merge reduction simplifies the DRS structure and facilitates use of the standard accessibility definition. Moreover, using MERGE, it is straightforward to define the set of discourse referents within a universe of a pure DRS B, namely, $U_{\text{MERGE}(B)}$. I will make use of this when I implement pronoun and presupposition resolution in the next section. Finally, merge reduction yields DRSs that can be transformed into first-order logic using the translation function given in Section 3.1.

4. Presupposition Resolution

In this section I will reformulate Van der Sandt's presupposition resolution algorithm in terms of α-DRSs with the aim of reducing the generate-and-test nature of Van der Sandt's original formulation. Even for relative simple examples, the search space of possible interpretations is vast. Consider the following example:

(27) If Bonnie finds a corpse in her house, the dead body will frighten her.

Without applying any acceptability constraints, resolving the four triggers in example (27) will yield 1,280 different solutions. If binding is preferred to accommodation, the example will still give rise to 525 possible interpretations. It is clear that a generate-and-test approach, in which the acceptability constraints are applied to completely resolved DRSs, will be extremely inefficient. The algorithm implemented in this section applies, as far as possible, the acceptability constraints to partially resolved DRSs, and thereby reduces the search space.

To deal with the different anaphoric behavior of noun phrases in English, I will propose a typology of noun phrases reflecting their properties with respect to binding and accommodation. Further, I will precisely formulate the acceptability constraints imposed by the resolution algorithm and outline how to add preference ranking to the solutions generated by the algorithm.

4.1 The Resolution Algorithm

Presupposition resolution in discourse processing is assumed to take place on an utterance-by-utterance basis. Therefore, the resolution algorithm takes as input the DRS constructed for the discourse so far and an α-DRS for the new utterance and outputs a new DRS. The algorithm is recursive in nature, and given an α-DRS A with n presupposition triggers, each step in the resolution process will resolve one trigger and decrease n by one, until A is presupposition-free (i.e., $n = 0$).

The idea behind the resolution algorithm is to detect any violations of the acceptability constraints as soon as possible in the process of resolution, thereby restricting the search space. After each resolution step the acceptability constraints are checked for violation. Some of the acceptability constraints, however, are not defined for α-DRSs and can be applied only to the finally resolved DRS.

I will present the algorithm in a notation borrowed from logic programming, using negation as failure, backtracking for efficiency, and unification for term manipulation. Given the definition below, it is straightforward to implement the algorithm using a programming language like PROLOG. DRSs are represented as an ordered pair of lists, and PROLOG-style variables are used to represent discourse referents and first-order variables. Further, I will use the following notational conventions: A, B, and C are used to denote DRSs, L and M to denote DRS-conditions, and P and Q to denote accessibility paths.

The main predicate of the algorithm consists of the following two clauses:

resolve(C,B,B) ← presup-free(B), consistent(B), informative(C,B).
resolve(C,B,E) ← alfa(B-D,[]-[A|P]), move(A,P,D), resolve(C,D,E).

The first *resolve* clause terminates the recursion if all presuppositions are resolved, then checks the resulting DRS for consistency and informativeness (see Section 4.3). The second, recursive *resolve* clause makes use of two further predicates that I introduce now: *alfa*, for determining the first presupposition trigger in the α-DRS, and *move*, which either binds or accommodates the trigger to an accessible level of discourse.

Let us consider first *alfa*, which is defined for a pair of α-DRSs (or α-DRS-conditions) and a pair of accessibility paths. (Recall that an accessibility path is a list of levels of DRSs, starting with the presupposition trigger and ending with the global level of discourse.) This is the definition for α-DRSs:

alfa((AαB)-(CαB),P-Q) ← alfa(A-C,P-Q).
alfa((AαB)-(C;B),P-[A,acc(C)|P]) ← presup-free(A).
alfa((A;B)-(C;B),P-Q) ← alfa(A-C,P-Q).
alfa((A;B)-(X;C),P-Q) ← presup-free(A), alfa(B-C,[bin(A,X)|P]-Q).
alfa(⟨U,L⟩-(A;R),P-Q) ← alfa(L-M,[acc(A),bin(⟨U,M⟩,R)|P]-Q).

What *alfa* effectively does is traverse the DRS structure, thereby building up the accessibility path, until it hits a presupposition trigger. The accessibility path is constructed as a list of binding or accommodation sites. Binding sites are marked as bin(A,B), where A is the original site (i.e., a DRS) and B the result of binding. Accommodation sites

are marked as acc(A), where A is the result of accommodation. For instance, consider the two clauses for the DRS merge in the definition above, in which the input DRS is of the form (A;B). If A contains a presupposition, then alfa(A-C,P-Q) holds, and the resulting DRS will be set to (C;B). If A is presupposition-free, B will be traversed for presuppositions (resulting in C) and a binding site X for A will be introduced on the accessibility path represented by P and Q. Possible binding sites are also introduced by basic DRSs. Basic DRSs furthermore introduce a possible accommodation site in case one of their complex conditions contain α-DRSs. The clauses for DRS-conditions are defined as follows:

alfa([X|L]-[X|M],P-Q) ← basic(X), alfa(L-M,P-Q).
alfa([¬B|L]-[¬C|L],P-Q) ← alfa(B-C,P-Q).
alfa([□B|L]-[□C|L],P-Q) ← alfa(B-C,P-Q).
alfa([◇B|L]-[◇C|L],P-Q) ← alfa(B-C,P-Q).
alfa([x:B|L]-[x:C|L],P-Q) ← alfa(B-C,P-Q).
alfa([B∨A|L]-[C∨A|L],P-Q) ← alfa(B-C,P-Q).
alfa([A∨B|L]-[A∨C|L],P-Q) ← presup-free(A), alfa(B-C,P-Q).
alfa([B⇒A|L]-[C⇒A|L],P-Q) ← alfa(B-C,P-Q).
alfa([A⇒B|L]-[(E;D)⇒C|L],P-Q)← presup-free(A), presup-free(A), alfa(B-C,[bin(A,E),acc(D)|P]-Q).

Note that, because we use ordered sets of DRS-conditions, the predicate *alfa* behaves in a deterministic way, and it returns the first presuppositional DRS that itself is presupposition-free. Further note that the accessibility path reflects the accessibility relation defined in DRT, which is mirrored by the clauses for *alfa*. For instance, note the difference in definition between the implicational and disjunctional condition.

The accessibility path returned by *alfa* forms a skeleton for a resolved DRS, which will be instantiated based on the decision as to whether to bind or accommodate the presupposition, and on which level. For a site encoded by acc(A), accommodation involves identifying the presuppositional DRS with A. For binding, it involves matching the presupposition with DRS A resulting in a new DRS B for a site of the form bin(A,B). This process is implemented by the predicates *move*, *bind*, *accommodate*, and *skip*. Let us first consider *move*. As BAT dictates, resolution involves either binding or accommodation:

move(A,P,B) ← bind(A,P), ¬sortal-violation(B), ¬binding-violation(B).
move(A,P,B) ← accommodate(A,P), ¬free-variables(B).

The first clause of *move* binds the presupposition to a DRS on the accessibility path and then checks the acceptability constraints (see Section 4.3 for the definition of these constraints). The second clause invokes accommodation, followed by a check on free variables (again, see Section 4.3). Let us first consider binding. Binding is possible only on binding sites, where the presuppositional information is matched with the DRS on the binding site, resulting in a new DRS. Binding is defined recursively, for there might be several appropriate binding sites:

bind(⟨D,L⟩$_X$,[bin(⟨D′,L′⟩,⟨D∪D′,L∪L′⟩)|P]) ← X∈D′, skip(P).
bind(A,[S|P]) ← bind(A,P), skip([S]).

Accommodation is very similar to binding, but only accommodation sites on the accessibility path are relevant and come into question. Accommodation now boils

down to unification of the presuppositional information with the DRS on the accommodation site. Again, there might be several accommodation sites (corresponding to global, local, or intermediate levels of discourse), so a recursive definition is appropriate:

accommodate(A,[acc(A)|P]) ← skip(P).
accommodate(A,[S|P]) ← accommodate(A,P), skip([S]).

Finally, we need to define *skip*. This function takes care of all accommodation and binding sites that were not selected as antecedent for the presupposition trigger. For a possible accommodation site, this will result in identifying its DRS with an empty DRS, and for a binding site, the resulting DRS will be unchanged with respect to its original one:

skip([]) ← true.
skip([acc($\langle\emptyset,\emptyset\rangle$)|P]) ← skip(P).
skip([bin(A,A)|P]) ← skip(P).

This is the core of the algorithm, but various extensions are possible. In the following section I will take different types of presuppositional triggers into account, because some expressions allow for accommodation and others do not, and some can be resolved only at the global level of discourse, whereas others are not sensitive to subordinated levels. Furthermore, I will formulate the acceptability constraints and investigate means to account for preferences in solutions. (As it stands, the algorithm will produce a set of solutions, all equal to one another. But as noted in the discussion of BAT, there are sometimes clear preferences for certain solutions.)

4.2 Classifying Presupposition Triggers

The resolution algorithm, as formulated in the previous section, does not discriminate among different types of anaphoric expressions. With regard to noun phrases, it is well known that the choice of referring expression affects coherence in discourse (Grosz, Joshi, and Weinstein 1995). Perhaps related to this observation is the fact that pronouns, definite descriptions, and proper names all vary in terms of their capacity for binding and accommodation. The performance of the algorithm would strongly benefit from taking these differences into account, because it would further narrow down the search space.

With respect to accommodation, some noun phrases allow accommodation on any level, whereas others accommodate only globally. Third-person anaphoric pronouns normally do not allow accommodation, with the exception of discourse-initial occurrences. Reflexive pronouns do not have the ability to accommodate, for they are intrinsically anaphoric. Definite descriptions, especially genitive constructions, have the power to accommodate on all levels (see example (7)). Proper names allow accommodation only on the global level. If one also considers first- and second-person pronouns, which belong to the family of deictic expressions, it can be concluded that this class of expressions does not allow accommodation at all, simply because deictic expressions refer to objects presumed in the context of interpretation.

For binding, the differences among noun phrases are not so marked. Most of them allow binding on all levels of discourse structure, with the exception of proper names.[4]

4 However, Geurts (1997) claims that proper names are able to bind nonglobally; he uses examples such as *If a child is christened "Bambi," and Disney Inc. hear about it, then they will sue Bambi's parents.*

Table 2
Binding and accommodation behavior of different α-types for noun phrases.

α-type	Local Binding/Accommodation	Global Binding/Accommodation	Description
`ref`	yes/no	yes/no	reflexive pronouns
`pro`	yes/no	yes/yes	third-person nonreflexive pronouns
`nam`	no/no	yes/yes	proper names
`dei`	no/no	yes/no	first- and second-person nonreflexive pronouns
`def`	yes/yes	yes/yes	definite descriptions

Antecedents of deictic expressions are assumed to be available at the global level of discourse, for they are part of the current context of interpretation, and so reference to objects at subordinated levels of discourse is not an option for deictic expressions.

To account for the different referential behavior of noun phrases, we classify them in terms of ***α*-types**. The α-types for English noun phrases and their properties are listed in Table 2. To integrate these properties into the resolution algorithm, we need a way to determine whether we resolve a particular presupposition at a global or nonglobal level of discourse. Given an accessibility path, it is unequivocal to define the conditions for operating on a global level of discourse, because subordinated levels of DRS introduce an accommodation site, which we represent as acc(A) for a DRS A, on the accessibility path. Therefore, a binding site is global if there is no accommodation site on the remainder of its accessibility path:

$$\text{global(P)} \leftarrow \neg\ \text{acc(A)} \in \text{P}.$$

With this machinery we are able to revise the definition for accommodation, by making it sensitive to different α-types. This results in the following clauses:

$$\text{accommodate}(\text{A}^{\text{nam}},[\text{acc(A)}|\text{P}]) \leftarrow \text{global(P)}, \text{skip(P)}.$$
$$\text{accommodate}(\text{A}^{\text{pro}},[\text{acc(A)}|\text{P}]) \leftarrow \text{global(P)}, \text{skip(P)}.$$
$$\text{accommodate}(\text{A}^{\text{def}},[\text{acc(A)}|\text{P}]) \leftarrow \text{skip(P)}.$$

Similarly, we can revise the definition of binding by making it sensitive to different α-types. This yields the following clauses:

$$\text{bind}(\langle \text{D},\text{L}\rangle^{\text{nam}}_{\text{X}},[\text{bin}(\langle \text{D}',\text{L}'\rangle,\langle \text{D}\cup\text{D}',\text{L}\cup\text{L}'\rangle)|\text{P}]) \leftarrow \text{global(P)},\text{X}\in\text{D}',\text{skip(P)}.$$
$$\text{bind}(\langle \text{D},\text{L}\rangle^{\text{dei}}_{\text{X}},[\text{bin}(\langle \text{D}',\text{L}'\rangle,\langle \text{D}\cup\text{D}',\text{L}\cup\text{L}'\rangle)|\text{P}]) \leftarrow \text{global(P)},\text{X}\in\text{D}',\text{skip(P)}.$$
$$\text{bind}(\langle \text{D},\text{L}\rangle^{\text{pro}}_{\text{X}},[\text{bin}(\langle \text{D}',\text{L}'\rangle,\langle \text{D}\cup\text{D}',\text{L}\cup\text{L}'\rangle)|\text{P}]) \leftarrow \text{X}\in\text{D}',\ \text{skip(P)}.$$
$$\text{bind}(\langle \text{D},\text{L}\rangle^{\text{ref}}_{\text{X}},[\text{bin}(\langle \text{D}',\text{L}'\rangle,\langle \text{D}\cup\text{D}',\text{L}\cup\text{L}'\rangle)|\text{P}]) \leftarrow \text{X}\in\text{D}',\ \text{skip(P)}.$$
$$\text{bind}(\langle \text{D},\text{L}\rangle^{\text{def}}_{\text{X}},[\text{bin}(\langle \text{D}',\text{L}'\rangle,\langle \text{D}\cup\text{D}',\text{L}\cup\text{L}'\rangle)|\text{P}]) \leftarrow \text{X}\in\text{D}',\ \text{skip(P)}.$$

This might be a rather rocky approximation to discriminating among different noun phrases, but it will greatly improve the performance of the algorithm. Whether a finer classification is required, or whether further types to deal with other kinds of presupposition triggers (such as factives) are needed, remains subject for future corpus studies.

4.3 Acceptability Constraints

The resolution algorithm imposes several acceptability constraints on resolved or partially resolved DRSs. For completely resolved DRSs, there are constraints on consistency and informativeness. For partially resolved DRSs (i.e., α-DRSs), there are constraints on sortal compatibility, binding, and the occurrences of free variables. Let us first consider consistency and informativeness.

As an illustration of the constraints on **consistency**, suppose we have a DRS B. If we can prove that $\neg\exists w(w,B)^{fo}$ is valid, then we know that B is inconsistent. If, on the other hand, we find that $\exists w(w,B)^{fo}$ is satisfiable, we know that B is consistent. In terms of our previous formulation of the resolution algorithm, this translates as

$$\text{consistent(B)} \leftarrow \text{proof}(\neg\exists w(w,B)^{fo}), \text{fail}.$$
$$\text{consistent(B)} \leftarrow \text{satisfiable}(\exists w(w,B)^{fo}).$$

Now let us consider **informativeness**, with respect to a DRS A representing the previous discourse and a new DRS B. If we prove that $\forall w((w,A)^{fo} \rightarrow (w,B)^{fo})$, we know that B is not informative with respect to A. On the other hand, if we are able to show that both $\exists w((w,A)^{fo} \wedge (w,B)^{fo})$ and $\exists w((w,A)^{fo} \wedge \neg(w,B)^{fo})$ are satisfiable formulas, we can say that B is informative with respect to A. This can be coded as follows:

$$\text{informative(A,B)} \leftarrow \text{proof}(\forall w((w,A)^{fo} \rightarrow (w,B)^{fo})), \text{fail}.$$
$$\text{informative(A,B)} \leftarrow \text{satisfiable}(\exists w((w,A)^{fo} \wedge (w,B)^{fo})),$$
$$\text{satisfiable}(\exists w((w,A)^{fo} \wedge \neg(w,B)^{fo})).$$

The constraint on **sortal compatibility** can be seen as a local consistency check. It takes place after binding, and it uses a sortal ontology to ensure that discourse referents with different sorts are not identified with each other. This eliminates any possibility for anaphoric expressions that describe entities to refer to discourse referents for temporal information or possible worlds and so cuts the search space of antecedents enormously. In terms of the algorithm, sortal-violation(B) will hold for a DRS B if there is an accessibility path in B with a discourse referent that has inconsistent properties.

The **binding constraint** is a linguistic confinement primarily dealing with restrictions of antecedents of anaphoric object noun phrases. Binding constraints are similar to the C-command constraints found in linguistic theory, but I will give a simplified formulation here and deal with (di)transitive verbs only. Binding constraints deal with two complementary cases. First of all, it checks whether a reflexive pronoun in object position is bound to the subject noun phrase. Second, it checks whether a nonreflexive anaphoric noun phrase in object position is not bound to the subject noun phrase.

Finally, there is the constraint on **bound variables** for α-DRSs. With the help of the definition of free and bound variables given in Section 3.3, it is straightforward to include this constraint in the resolution algorithm. This can be accomplished efficiently by traversing the DRS in a top-down manner, collecting bound variables on each accessibility path. Free variables are then detected when one of the variables occurring in a basic condition is not a member of the set of bound variables collected on that accessibility path.

4.4 Preferences

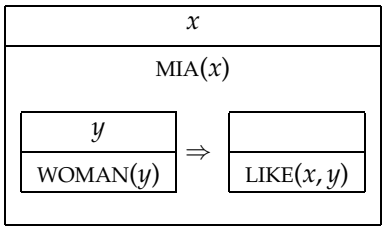
In this section I will investigate how to account for ranking interpretations. The resolution algorithm will produce a set of solutions without stating any preferences among

candidates in the solution set produced. In general, it has been noted that in most cases, binding is strongly preferred to accommodation (Van der Sandt 1992), and that global accommodation is preferred to local accommodation (Heim 1983). This meshes well with a claim put forward in theories of discourse coherency that the inference demands placed on a hearer correlate positively with the perceived coherency of a discourse (Grosz, Joshi, and Weinstein 1995), because it makes perfect sense to ascribe a higher inference load to accommodation than binding.

Given the resolution algorithm as defined here, one way to invoke a ranking mechanism among potential solutions is to include **scores** and make them sensitive to different α-types. Scores could be represented as numbers between 0 and 1, reflecting the rank of the solution in the solution set. Starting with a score of 1 for a particular solution, accommodation will decrease the score (for instance, by multiplying the current score by 0.1), whereas binding will not. Cases of nonglobal accommodation will further lower the score.

Van der Sandt's constraints on local informativeness and local consistency are further criteria for preference ranking. Unlike the global versions of informativeness and consistency, the local constraints cannot be "hard" constraints, for if they were, they would rule out otherwise fine solutions. Put differently, rejection of DRSs on the basis of violating the local informativeness constraint seems inappropriate. I will illustrate this observation with the discourse and its translation in a DRS (example (28)).

(28) Mia likes every woman.



The DRS in example (28) violates the local informativeness constraint. The sub-DRS containing the information that there is a woman is already expressed by the superordinated DRS, given the background knowledge that Mia is a woman. From a grammatical point of view, however, example (28) is a legitimate sentence. Rejecting it on the basis of violating local informativeness seems unjustified. On the other hand, the local constraints help in dealing with the presupposition project problem. Therefore, we take a suggestion made by Beaver (2002) and use the local constraints as a further criterion for ranking potential solutions of the resolution algorithm. This ranking could be realized by decreasing the score of a particular DRS each time it violates local informativeness or local consistency.

Incidentally, Van der Sandt does not give a precise formulation of local informativeness and local consistency, and it is not straightforward what would constitute a precise formulation (Beaver 1997, 2002). I will give a novel formulation of the local constraints with the help of a function that, given a DRS, returns a set of pairs of DRSs and the DRSs that they subordinate. Given this function, it is straightforward to define local informativeness and local consistency. This function, *supersub*, is defined, using PROLOG notation, as follows (the definition given here is restricted to the clauses for implication, negation and basic conditions; clauses for the remaining conditions can

be easily derived from these):

supersub(⟨D,[Sub⇒B|L]⟩,A-(A;⟨D,L⟩),Sub) ← true.
supersub(⟨D,[B⇒Sub|L]⟩,A-(A;⟨D,L⟩;B),Sub) ← true.
supersub(⟨D,[Sub⇒B|L]⟩,A-Sup,Sub) ← supersub(B,(A;⟨D,L⟩)-Sup,Sub).
supersub(⟨D,[B⇒C|L]⟩,A-Sup,Sub) ← supersub(C,((⟨D,L⟩;B);A)-Sup,Sub).
supersub(⟨D,[Sub∨B|L]⟩,A-(A;⟨D,L⟩),Sub) ← true.
supersub(⟨D,[B∨Sub|L]⟩,A-(A;⟨D,L⟩;B)),Sub) ← true.
supersub(⟨D,[Sub∨B|L]⟩,A-Sup,Sub) ← supersub(B,(A;⟨D,L⟩)-Sup,Sub).
supersub(⟨D,[B∨Sub|L]⟩,A-Sup,Sub) ← supersub(B,(A;⟨D,L⟩)-Sup,Sub).
supersub(⟨D,[¬Sub|L]⟩,A-(A;⟨D,L⟩),Sub) ← true.
supersub(⟨D,[¬B|L]⟩,A-Sup,Sub) ← supersub(B,(A;⟨D,L⟩)-Sup,Sub).
supersub(⟨D,[X|L]⟩,A-(⟨D,[C]⟩;B),Sub) ← basic(X),supersub(⟨∅,L⟩,A-B,Sub).

The *supersub* predicate is recursively defined to handle arbitrarily deeply embedded sub-DRSs. The crucial step in *supersub* is removing the conditions that contain the subordinated DRS. Applying *supersub* to example (28), we get the following two pairs:

⟨super-DRS: $\boxed{\begin{array}{c} x \\ \hline \text{MIA}(x) \end{array}}$, sub-DRS: $\boxed{\begin{array}{c} y \\ \hline \text{WOMAN}(y) \end{array}}$⟩

⟨super-DRS: $\boxed{\begin{array}{c} x\ y \\ \hline \text{MIA}(x) \\ \text{WOMAN}(y) \end{array}}$, sub-DRS: $\boxed{\begin{array}{c} \\ \hline \text{LIKE}(x,y) \end{array}}$⟩

We can now formulate the local constraints in terms of informativeness as follows: A pair ⟨super-DRS: A, sub-DRS B⟩ is locally informative if (A;B) is informative with respect to A and locally consistent if (A;⟨∅,[¬B]⟩) is informative with respect to A.

Adding the local constraints as further criteria for ranking potential readings produced by the resolution algorithm will give accurate predictions for the interpretation of many problematic cases discussed in Section 2.2. Testing the local constraints involves first-order theorem proving, and it is useful to add some heuristics to obtain more efficient implementations. A valuable heuristic is one that distinguishes subordinated levels of discourse in the old DRS (i.e., the DRS capturing the portion of the discourse processed so far) from subordinated levels of discourse in the DRS of the newly processed utterance. This avoids repeated application of local constraints to the same subordinated levels of discourse over and over again.

5. Implementation and Performance

The resolution algorithm is implemented as part of a natural language understanding system. I will describe the general architecture underlying this system and the implementation of the algorithm and the acceptability constraints and present performance results obtained from applying the algorithm to a corpus of route instructions.

5.1 Architecture

Open Agent Architecture (OAA) (Cheyer and Martin 2001) is used as prototyping environment to implement the presupposition resolution component as part of a natural language understanding system. OAA is a collection of software agents that communicate with each other via a facilitator, a piece of middleware that distributes requests to appropriate agents and returns the responses to the requester. OAA makes it convenient to combine different components that are required in natural language process-

ing, such as speech recognition or parsing, the presupposition resolution component, and theorem provers, because OAA agents can be implemented in different programming languages and run simultaneously on different machines (Bos and Oka 2002). The resolution component is realized as an OAA agent implemented in PROLOG.

5.2 Acceptability Constraints

To implement inference, a theorem prover as well as a model builder is used, both encapsulated as OAA agents. The theorem-proving agent is used to find a counterproof for the DRS translated into first-order logic. The model-building agent is used to check whether the same DRS is satisfiable. So, although we are faced with the limitations for reasoning with first-order logic (validity is undecidable in first-order logic, and model generation is restricted to finite models), these limitations are reduced to a minimum. For each inference problem, the two inference agents attack the problem in parallel, and as soon as one of them finds an answer (a model or a counterproof), their task is completed.

The three acceptability constraints that do not require first-order inference (proper binding, bound variables, and sortal compatibility) are not implemented as separate agents but instead are part of the resolution agent. Proper binding is checked via a neo-Davidsonian semantics to describe events in terms of their thematic relations (Parsons 1990). Binding is violated when a (di)transitive verb has a reflexive pronoun as object and the discourse referents for the agent and patient denote different objects, or when a (di)transitive verb has a nonreflexive object and the discourse referents for agent and patient denote the same object. The check for free variables is rather straightforward, given the definitions in Section 3.

Sortal violations are detected using a conceptual ontology. Based on WordNet (Fellbaum 1998), this ontology is substantially adapted and extended to deal with anaphora resolution in BAT. As usual, it reflects background knowledge in the form of inheritance (is-a) and disjointness. The three (disjoint) top concepts in this ontology are GROUP (a collection of things), SITUATION (a condition in which certain propositions hold or do not hold), and THING (an individual object that is talked about). The last is further divided into ABSTRACTION (a thing without mass) and ENTITY (a thing with mass). The concept ENTITY has two subconcepts: OBJECT (a nonliving entity) and ORGANISM (a living entity). OBJECTS are divided into ARTIFACTS (human-made things), NATURAL-OBJECTS (things that are found in nature), and SUBSTANCES (things that are indivisible). The subconcepts of ORGANISM are HUMAN, ANIMAL, and PLANT.

In the case of English pronouns, there is a need to distinguish between third-person singular male, female, and neuter pronouns, as well as, of course, plural pronouns. The plural pronouns are the easiest to deal with and introduce a discourse referent with condition GROUP; hence they cannot bind to situations or things. Three mutually disjoint concepts are used for singular pronouns: MALE (for *he*), FEMALE (for *she*), and UNISEX (for concepts that disallow binding of *he* and *she*). The neuter pronoun *it* comes with the feature NONHUMAN, so we allow it to refer to any nonhuman entity (this is obviously not entirely accurate, as in certain situations, *it* can be used to refer to persons). To prevent reference from singular pronouns to plural entities, we further define GROUP disjoint from MALE, FEMALE, and NONHUMAN.

The sortal violation checker is implemented in PROLOG, where the inheritance information is stored in the PROLOG database by clauses of the following form:

```
sort(ENTITY(X)) ← sort(ORGANISM(X)).
sort(ENTITY(X)) ← sort(OBJECT(X)).
sort(ORGANISM(X)) ← sort(HUMAN(X)).
sort(ORGANISM(X)) ← sort(ANIMAL(X)).
```

sort(ORGANISM(X)) ← sort(PLANT(X)).
sort(MALE(X)) ← sort(MAN(X)).
sort(FEMALE(X)) ← sort(WOMAN(X)).

Disjointness relations are implemented by clauses of the following form:

inconsistent ← sort(ORGANISM(X)), sort(OBJECT(X)).
inconsistent ← sort(HUMAN(X)), sort(ANIMAL(X)).
inconsistent ← sort(MALE(X)), sort(FEMALE(X)).

For each sortal compatibility check, the discourse referents are skolemized, and the basic conditions of the resolved DRSs are asserted to the database. The following clause links these basic conditions to sorts:

sort(S) ← basic(S).

The PROLOG inference engine then attempts to prove a sortal incompatibility by trying to find an instance of a discourse referent that has two conflicting properties, within the transitive closure of the is-a relation, here implemented via the predicate *sort*. Using negation as failure, sorts are compatible if ¬*inconsistent* can be proven. Consider the following example illustrating sortal incompatibility:

(29) Suppose the result of binding is a DRS in which the two basic conditions MAN(X) and WOMAN(X) are applied to the same variables. Asserting this to the database as *basic*(MAN(*a*)) and *basic*(WOMAN(*a*)), it is possible to conclude *sort*(MAN(*a*)) as well as *sort*(WOMAN(*a*)). From this, we are able to conclude *sort*(MALE(*a*)) and *sort*(FEMALE(*a*)), and we can prove that *inconsistent* holds.

Summarizing, the sortal compatibility check is used as a filter for the more general consistency check, for which fully fledged first-order theorem proving is used. If it is impossible to prove *inconsistent*, it is assumed that the antecedent discourse referent is compatible with its binder. As I will show in the next section, this filter reduces the search space in resolution enormously.

5.3 Performance

The resolution algorithm was tested on a corpus of route instructions collected in a scenario in which somebody explains to a mobile robot how to reach a certain destination. The corpus, collected in the IBL project (Lauria et al. 2001), comprises 283 utterances in 72 different route instructions, spoken by 24 different native English speakers. A typical sequence is the following:

(30) *Instructor*: Go to the university!
Robot: How do I get to the university?
Instructor: Go straight ahead until you reach the post office. Just past the post office turn left over the bridge. Keep walking, there will be a building on the right and a building on your left. Keep walking until you come to a train station on the left hand side and the university is opposite the train station.

The corpus was processed on utterance-by-utterance basis, starting with a new DRS for each new route instruction. Only the first (consistent) solution returned by

Table 3
Occurrences of triggers in 283 utterances taken from spoken route instructions.

α-type	n	Percentage	Average per utterance
`ref`	1	0.1	0.04
`pro`	42	4.7	0.15
`nam`	50	5.6	0.18
`dei`	380	42.3	1.34
`def`	425	47.3	1.50
Total	898	100.0	3.17

Table 4
Average CPU times for DRS resolution relative to the number of processed utterances.

Number of utterances	1	2	3	4	5	6	7	8	9
CPU time	0.542	0.709	1.140	1.654	1.177	1.495	6.037	2.462	0.773
n	68	60	54	41	27	12	8	4	3

the resolution algorithm was considered for subsequent processing of the route instruction. A total of 898 referential expressions appeared in the 283 utterances of the corpus. As Table 3 shows, pronouns and proper names are relatively rare in these route instructions, but on average there are 1.5 definite noun phrases per utterance.

The average number of accommodation sites for a presupposition trigger in this corpus was 7.5. (This relatively high number can perhaps be attributed to the way DRSs are nested into each other in representing route instructions and the way utterance grounding is realized in the DRS. Discussion of these issues, however, falls outside the scope of this article.) The average number of potential antecedents (i.e., accessible discourse referents) for binding a presupposition trigger was 16.7. These statistics illustrate the immense search space in presupposition resolution.

The implemented resolution algorithm performs with an average CPU time (measured on a Sun Blade 100 workstation with 1 GB memory and a 500 MHz processor) of 1.21 seconds to transform an unresolved α-DRS into a proper DRS (disregarding the consistency checking; see below). Table 4 shows the average CPU times for DRS resolution relative to the number of processed utterances and so illustrates the dependence of processing time on the size of the DRS capturing the previous discourse.

To find out which of the acceptability constraints contribute the most in narrowing down the search space, the number of attempts and success/failure rate were computed for sortal compatibility, proper binding, and bound variables, after a presuppositional DRS has been resolved or accommodated. Most of the credit for reducing the size of the search space goes to checking for sortal violations, which were detected 8,111 times in 8,303 attempts (97%). Only 99 (1.04% of 9,466 cases) α-DRSs were found to contain free variables. Similarly rare were cases of binding violation (73 occurrences in 9,156 considered cases). Still, it pays off to verify these constraints on partially resolved α-DRSs. For instance, the average CPU time for resolving a DRS that violated the bound variable constraint during resolution was 2.5 seconds $(n = 35)$ when this constraint was checked partially, but 15.0 seconds when it was checked on fully resolved representations.

Finally, let us consider the findings regarding the use of first-order inference engines to implement consistency checking of DRSs. This is a very hard task: Dialogues

such as that in example (30) generate up to several hundred thousand clauses. Moreover, off-the-shelf provers are not designed for linguistic problems. Instead, they are mostly tuned to mathematical problems.

Several theorem provers and model builders were put to the test, including Hans de Nivelle's (1998) BLIKSEM, which is optimized for the "guarded" fragment of first-order logic, Bill McCune's OTTER and MACE (McCune and Padmanabhan 1996; McCune 1998), and the theorem prover SPASS (Weidenbach et al. 1999). For this particular task the model builder MACE and the theorem prover SPASS clearly outperformed the other inference engines; they were able to find an answer within 30 seconds for 66% of the 283 inference problems assigned to them (the majority of the DRSs being consistent) in CPU times varying from 2.5 to 29.9 seconds (average 13.0 secs).

These results are perhaps too limited to justify the inclusion of first-order theorem proving in today's natural language understanding components. Nevertheless, I believe that first-order theorem provers will play an important future role in computational semantics for three reasons. First of all, automated theorem proving is a promising, emerging field. Moreover, most of the first-order inference engines, albeit general purpose, are designed to cope with nonlinguistic problems, and cooperation of computational linguists with researchers in the area of automated deduction might improve the performance of these inference engines on linguistic inference problems. Second, the current approach is nonincremental. After a new utterance is combined with the previous DRS, the complete newly constructed DRS is translated to first-order logic and checked for consistency, without appealing to previous inference results at all. It is likely that inference-based natural language understanding would benefit from an incremental approach, particularly with regard to model building. Third, there is room for improvement in the formulation of the inference problem itself. Future work should address the use of sorted logics, include experimenting with other modal formulations, and consider the use of discourse structure to limit the size of DRSs to be checked for consistency.

6. Conclusion

The implementation of BAT presented in this article covers a wide spectrum of referential expressions, ranging from simple pronouns to rich presupposition triggers. Compared to Van der Sandt's (1992) original formulation of BAT, the implementation discussed here offers improvements on both the representational and inferential levels. Representational aspects of my reformulation of BAT include a new syntax for unresolved DRSs, which allows for selective binding and moreover provides a means of defining free and bound variables on these structures and hence enables us to implement the free-variable constraint on partially resolved DRSs. The inferential aspects involve a formulation of the acceptability constraints concerning consistency and informativeness, in particular the local versions of these, and an implementation of these constraints using general-purpose theorem provers for first-order logic.

The core of the algorithm is realized as a PROLOG agent within the OAA environment. Off-the-shelf first-order theorem provers are used as agents to perform the required reasoning tasks. Results of tests conducted on a corpus of spoken route instructions suggest that the core of the algorithm performs reasonably well, certainly when one considers the enormous search space that is involved in presupposition resolution. The importance of reducing the search space is demonstrated in an experiment in which the free-variable constraint, imposed on partially resolved representations, shows a substantial increase in performance compared to when it is imposed on completely resolved representations. Furthermore, to distinguish among the different

anaphoric natures of various presuppositional expressions, a classification of α-types for presuppositions triggered by noun phrases is proposed and used to increase efficiency in the implementation of the resolution algorithm.

Although the algorithm presented in this article already includes a number of heuristics to state preferences among a number of candidates of resolved discourse representations, it could further benefit from suggestions made in centering theory (Grosz, Joshi, and Weinstein 1995) to express salience among discourse referents. In the terminology of centering, discourse referents can naturally be seen as **centers**, with forward-looking centers being a subset of the accessible discourse referents and the unique backward-looking center being a designated discourse referent resembling the focus of discourse. These aspects of centering theory will enable us to include a notion of coherence in BAT, but how to establish the ranking of forward-looking centers (to reflect relative prominence) in a DRT-based formalism remains a subject of future research.

Acknowledgments

The research reported in this article originated at the Department of Computational Linguistics at the University of the Saarland in Germany and was further developed at the Institute for Communicating and Collaborative Systems at the University of Edinburgh. I wish to thank Patrick Blackburn and Manfred Pinkal for supervision, discussion, and advice. I am grateful to David Beaver, Hans Kamp, Ewan Klein, Michael Kohlhase, Hans de Nivelle, and Rob van der Sandt for their valuable suggestions, which greatly improved the presupposition resolution algorithm presented in this work. Tetsushi Oka helped design and implemented the inference agents in the Open Agent Architecture. The corpus of route instruction was collected and transcribed by Guido Bugmann, Stasha Lauria, Theo Kyriacou, and Joe Frankel (as part of the IBL project, under grants GR/M90023 and GR/M90160). Finally, I would like to thank two anonymous reviewers for this journal for their valuable suggestions.

References

Beaver, David I. 1997. Presupposition. In Johan Van Benthem and Alice Ter Meulen, editors, *Handbook of Logic and Language*. Elsevier, Amsterdam, and MIT Press, Cambridge, pages 939–1008.

Beaver, David I. 2002. Presupposition Projection in DRT: A critical assessment. In David Beaver, Luis Casillas, Brady Clark, and Stefan Kaufmann, editors, *The Construction of Meaning*. Stanford University Press, Stanford, California, pages 23–43.

Blackburn, Patrick. 2000. Representation, reasoning, and relational structures: A hybrid logic manifesto. *Logic Journal of the IGPL*, 8(3):339–365.

Blackburn, Patrick, Johan Bos, Michael Kohlhase, and Hans de Nivelle. 2001. Inference and computational semantics. In Harry Bunt, Reinhard Muskens, and Elias Thijsse, editors, *Computing Meaning*, volume 2. Kluwer, Dordrecht, pages 11–28.

Bos, Johan and Tetsushi Oka. 2002. An inference-based approach to dialogue system design. In *Proceedings of COLING 2002*.

Cheyer, Adam and David Martin. 2001. The Open Agent Architecture. *Journal of Autonomous Agents and Multi-Agent Systems*, 4(1/2):143–148.

de Nivelle, Hans. 1998. A resolution decision procedure for the guarded fragment. In C. Kirchner and H. Kirchner, editors, *15th International Conference on Automated Deduction, CADE-15*. Springer-Verlag, Berlin, pages 191–204.

Fellbaum, Christiane, editor. 1998. *WordNet. An Electronic Lexical Database*. MIT Press, Cambridge.

Geurts, Bart. 1997. Good news about the description theory of names. *Journal of Semantics*, 14:319–348.

Geurts, Bart. 1999. *Presuppositions and Pronouns*. Elsevier, London.

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

Grosz, Barbara J., Aravind K. Joshi, and Scott Weinstein. 1995. Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics*, 21(2):203–225.

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

Heim, Irene. 1983. On the projection problem for presuppositions. In M. Barlow, D. Flickinger, and M. Westcoat, editors, *Proceedings of the Second Annual West Coast Conference on Formal Linguistics*, pages 114–126.

Kamp, Hans. 1981. A theory of truth and semantic representation. In Jeroen Groenendijk, Theo M. V. Janssen, and Martin Stokhof, editors, *Formal Methods in the Study of Language*, pages 277–322.

Kamp, Hans and Uwe Reyle. 1993. *From Discourse to Logic: An Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and DRT*. Kluwer, Dordrecht, the Netherlands.

Karttunen, L. 1974. Presuppositions and linguistic context. *Theoretical Linguistics*, 1:181–194.

Kuschert, Susanna. 1999. *Dynamic Meaning and Accommodation*. Ph.D. thesis, Universität des Saarlandes, Saarbruecken, Germany.

Lauria, Stanislao, Guido Bugmann, Theocharis Kyriacou, Johan Bos, and Ewan Klein. 2001. Training personal robots using natural language instruction. *IEEE Intelligent Systems*, 16(5):38–45.

Lewis, David. 1979. Scorekeeping in a language game. In R. Bäuerle, U. Egli, and A. von Stechow, editors, *Semantics from Different Points of View*, volume 6 of *Springer Series in Language and Communication*. Springer-Verlag, Berlin, pages 172–187.

McCune, W. 1998. Automatic proofs and counterexamples for some ortholattice identities. *Information Processing Letters*, 65(6):285–291.

McCune, W. and R. Padmanabhan. 1996. *Automated Deduction in Equational Logic and Cubic Curves*. *Lecture Notes in Computer Science* (AI subseries). Springer-Verlag, New York.

Moore, Robert C. 1980. Reasoning about knowledge and action. Technical Report 181, SRI International, Menlo Park, California.

Muskens, Reinhard. 1996. Combining Montague semantics and discourse representation. *Linguistics and Philosophy*, 19:143–186.

Parsons, Terence. 1990. *Events in the Semantics of English: A Study in Subatomic Semantics.* MIT Press, Cambridge.

Van der Sandt, Rob A. 1992. Presupposition projection as anaphora resolution. *Journal of Semantics*, 9:333–377.

Van der Sandt, Rob A. and Bart Geurts. 1991. Presupposition, anaphora, and lexical content. Technical Report 185, IBM, Wissenschaftliches Zentrum, Institut für Wissensbasierte Systeme, August.

Van Eijck, Jan and Hans Kamp. 1997. Representing discourse in context. In Johan Van Benthem and Alice Ter Meulen, editors, *Handbook of Logic and Language*. Elsevier, Amsterdam, and MIT Press, Cambridge, pages 179–240.

Weidenbach, Christoph, Bijan Afshordel, Uwe Brahm, Christian Cohrs, Thorsten Engel, Enno Keen, Christian Theobalt, and Dalibor Topic. 1999. System description: Spass version 1.0.0. In Harald Ganzinger, editor, *16th International Conference on Automated Deduction, CADE-16*, volume 1632 of *Lecture Notes in Artificial Intelligence*. Springer, Berlin, pages 314–318.