Multimodal Cooperative Resolution of Referential Expressions in the DenK System

Leen Kievit and Paul Piwek
Computational Linguistics and A.I. Group,
Tilburg University & ITRI,
University of Brighton

To appear in: Bunt, H. & R.J. Beun, Proceedings of CMC'98.
Springer Lecture Notes in AI

Abstract

In this paper, we focus on the problem of the resolution of multimodal referential expressions in a cooperative man-machine communication setting. We do not only concentrate on how to resolve references involving multiple modalities, but we also indicate how the system can respond cooperatively in case this resolution should fail.

1 Introduction

Our approach to the resolution of referential expressions is embedded in the DenK-architecture (see Ahn et al., [1995]; Bunt et al., this volume)[1], a generic cooperative multimodal user interface. The idea is that we have three parties in a cooperative environment: a user, a domain and a cooperative assistant. The *domain* may be any (simulated) self-contained part of the real world (e.g., VCR's, television sets or software applications). The user is shown a graphical representation of the domain and she can directly manipulate objects in the domain using a mouse. The *assistant* is there to help the user in the interaction with the domain. Such assistance is provided through multimodal communication with the user (English utterances and graphical selection/highlighting). The assistant can answer questions of the user about the domain and it can perform actions inside the domain upon request of the user.

In the actual implementation of the DenK-architecture –carried out at Tilburg University and Eindhoven University of Technology– the domain is a simulation of an electron microscope using an object oriented display processor.

[1]DenK stands for Dialogue Modelling and Knowledge Acquisition. 'Denk' is the Dutch equivalent of 'think'.

The assistant is mainly programmed in Prolog and consists of several modules: a parser, a semantic interpretation module (which uses underspecification and several strategies for ambiguity resolution), a reasoning module, and a dialogue management module which oversees the interaction with the user and can generate responses to the user's utterances, including clarificatory questions in case of ambiguous utterances.

Because of the different modalities involved in the interaction with the system, the system is also suited to handle multimodal references to objects: a user can select an object and use a deictic term, or refer to something that is visible on the screen using a pronoun. Reversely, if the system wants to identify an object to the user, there is a choice of modality. The system can highlight the object or describe it in natural language. The generation of referring expressions will not be taken up in this paper.

In this paper, we concentrate on the module in the DenK system that finds referents for referring expressions and on the dialogue manager's options in cases where a referential expression cannot be resolved.

The paper is built up as follows: in the next section we will introduce the format of the formal representations of the user's utterances and the different sources of information which the assistant uses in interpretation (in particular, reference resolution). In section 3 we will describe the multimodal resolution algorithm and the way the dialogue manager can respond to cases where a referent could not be identified. That section ends with some examples. Section 4 compares our work with related research. Finally, we draw some conclusions.

2 Contexts in Dialogue

In the DenK-system, the user can interact with the cooperative assistant via typed input (possibly combined with object selection using the mouse). The assistant responds to such communicative actions on the basis of the actions themselves and the context in which the actions are performed. The assistant computes a response in two stages: the *interpretation* and the *reaction* stage. In the interpretation stage, a formal representation of what the user meant to say is computed on the basis of the typed input and the context. In the reaction stage, a reaction is computed on the basis of the formal representation of the user's utterance and the context[2].

Interpretation The output of the interpretation stage is a so-called *annotated segment* (Piwek [1995]). An annotated segment (AS) consists of a type theoretical *segment* and an *annotation*. Segments are expressions of Constructive Type Theory (CTT, see, e.g., Barendregt [1992]; De Bruijn [1980]; Martin-Löf [1984]).

[2]In the reaction stage the system also determines whether the user's utterance counts as a direct or indirect speech act. For details, see Beun and Piwek ([1997]).

A segment is a sequence of introductions of the form $O : T$, where O is an object and T is the type of the object. Both objects and types can be either atomic or complex terms. One way to build up complex terms is by applying functions to an argument. For instance, 'A lens is on' can be represented as $x : lens, y : on - x$. The first introduction presents an object x of type $lens$. The y in the second introduction stands for a proof of the proposition $on - x$. This proposition is represented by a complex term formed by applying on, a predicate that needs one argument, to the object x.
In CTT, a proposition corresponds to the set of all proofs for that proposition. Thus, proofs are explicitly represented in the CTT language and can fulfill the same role as discourse referents in Discourse Representation Theory (DRT; Kamp and Reyle [1993]). Ahn and Kolb ([1990]) demonstrated that CTT can be seen as a higher order generalization of DRT.

To form an annotated segment, a CTT segment is paired with an annotation, which is a set of feature-value pairs. The annotation contains, among other things, information about what has traditionally been called the *force* of the utterance. The question 'which lens is on?' corresponds to the segment $x : lens, y : on - x$. This segment is also a representation for 'A lens is on'. Here, the annotation indicates the difference. In case of the wh-question, the annotation contains the information that x carries the feature ri (Requested Information). In the reaction stage, this information tells the system that it has to generate an answer, i.e., identify to the user the lens that is on.

Such annotated segments form the end product of the interpretation stage. To arrive at this end product the string that the user typed in has to be interpreted using contextual information. We will focus on one specific part of this process: *resolution of referential expressions*. We use the annotations of annotated segment to indicate which parts of a segment need to be filled in in order to get a fully specified representation of what the user means. Consider, for instance, the utterance 'The lens is on' containing the referential 'The lens'. Before resolution we have a segment $x : lens, y : on - x$ and the information in the annotation that $rb = [[x]]$. (rb stands for 'requires binding'). In words, an object has to found in the context which can be substituted for x.

The context here encompasses the *domain*, the *dialogue history* and the *common context* (information which the system believes to share with the user). The problem now is to determine in which context to look for the object, and furthermore to determine which object is meant if there is more than one object satisfying the predicate *lens*. For the latter purpose, we invoke the notion of *salience*. For instance, the salience of an object in the dialogue history depends on how recent the object was mentioned: the most recently mentioned object is most salient. In the domain, an object that is visible in a window on the screen is more salient than one that is hidden.

Formally, the dialogue history and the common context are represented using CTT enriched with annotations. Both are *part of* the 'cognitive state' of the system. On the other hand, the domain is system *external*: it is a simulation in an object oriented language of the electron microscope. However, the system can verify propositions externally; for instance, it can look for lenses in the domain.

Reaction We have already introduced the common context and the dialogue history as parts of the cognitive state of the system. Here, we add two additional contexts: the *private* context (containing information which the system thinks it does no share with the user) and the *pending* context. The pending context is a list that contains utterances of both of user and the system which still have to be dealt with (e.g., a command which still needs to be reacted to). In the reaction stage, the representation of the user's utterance is added to the pending context, but only after all utterances which are at that point no longer under discussion have been removed For instance, consider the following dialogue fragment:

(1) a. U: Switch on the lens.
b. S: ok (does it).
c. U: Now, increase its excitation.

When utterance (c). is interpreted, the pending context is equal to $[b, a]$. ¿From the content of (c), the system can infer that the user is satisfied with the system's response to utterance (a). This means both (a) and (b) have been processed succesfully and thus can be removed from the pending context. Utterance (c). is then added. The new pending context is $[c]$. Furthermore, the dialogue history is updated with the objects mentioned. In the case of utterance c. there is a reference to the lens (introduced in (a)) and its excitation. Representations of both objects are added to the dialogue history.

After the pending context has been updated, the system examines its first element and reacts to it. We distinguish three possibilities: (i) The formal representation of the utterance is completely unambiguous, enabling the user to generate an appropriate response. In case of a question, the system may base its response on it *private* context. The context typical contains information about regularities in the domain (e.g., *If the diffraction lens is off, the microscope is in imaging mode*) which the user is not familiar with. The generated response to the user's input is communicated and simultaneously added to the pending context. By adding it to the pending context, a subsequent reaction of the user can be interpreted in the light of the preceding utterance of the system. (ii) The formal representation is not fully unambiguous. In particular, the system may not have been able to resolve all referential material in the user's utterance. In that case, the system can generate a clarificatory question, e.g.:

(2) a. U: Switch the lens on.
b. S: Which lens.
c. U: The condenser lens.
d. S: ah, ok (does it).

(iii) Finally, the representation of the user's utterance may not contain sufficient clues for the system to ask a clarificatory question. For instance, some of the words that the user used may not be part of the lexicon. In such cases, the system generates an error message, stating that it does not understand the user, and why it does not understand the user. For example:

(3) a. U: Switch the magnation lens on.
    b. S: I do not know the word 'magnation'.

So, in this section we saw that in the DENK-system there are several contexts that are involved in both interpretation and reaction: the domain, the dialogue history, the common context, the private context and the pending context.

In the next section we will see how some of these different sources of information are used by the resolution algorithm and the dialogue manager.

3 Multimodal cooperative reference resolution

The resolution algorithm is based on the following assumptions:

- Referents for referential expressions can come from different sources available to the dialogue partners.
- The intended referent for an expression is the *most salient fitting object*.
- Exactly when an object is fitting and when it is salient, depends on the kind of referential expression.
- When no fitting salient object can be found, or several objects are fitting and equally salient, the algorithm should halt and report the outcome.

Given these considerations, the algorithm is constructed as a guided search through the available sources. For each kind of referential expression, a *search strategy* is defined. A strategy consists of one or more *resolution steps*. Each resolution step in turn consists of 4 elements:

- The *selection* of possible objects (candidate referents) from a single context.
- The application of a number of *filters* on the set of candidates.
- The ordering of candidates by saliency.
- An *evaluation* of the result.

So, each step consists of a sequence **selection-filtering-ordering-evaluation**. The selection and filtering make sure that the remaining candidates are at least fitting objects, the ordering allows us to make use of the saliency. Now, the algorithm halts as soon as one of the resolution steps finds a unique object (positive outcome) or finds several objects and cannot choose which is the intended one (negative outcome). This means that sometimes not all resolution steps are actually performed. This in turn guarantees that we use the *least possible effort* to find the referent, although we must be careful not to make a premature choice. This means the precise order of the resolution steps is a matter of careful consideration and is subject to empirical testing.
The current version of the resolution strategies is based on (Beun & Kievit, 1995).

We will now give some examples of the algorithm, to demonstrate how it handles several situations.

Take dialogue (1) in section 2. The first sentence contains a referential expression of type `def_art` (noun phrase with definite article). The annotated segment would be $\langle [x : lens, y : on - x], [rb = [[x]], mt = [x : def\_art, ra = [[y]]\rangle$ (mt stands for 'marker type', ra for 'requested action').
Below we give the resolution strategy for `def_art`. Each step in the strategy is defined in terms of: (1) the context from which candidate objects are selected, (2) the set of filters which are applied to the set of candidate objects, (3) an ordering algorithm for the filtered set of candidates and (4) a set of evaluation clauses for the ordered and filtered set of candidates.

```
res_step(
        candidates(hypothetical),
        filters([prove(basic+hypothetical)]),
        ordering([]),
        evaluation([ length>1 means fail,
                     length=1 means succeed,
                     otherwise continue ])
        ),
res_step(
        candidates(private),
        filters([prove(common),
                 observe,
                 prove(private)]),
        ordering([dialogue]),
        evaluation([ length=0 means continue,
                     dialogue_pos_first<10 means succeed,
                     otherwise store ])
        ),
res_step(
        candidates(domain),
```

```
            filters([prove(common),
                     observe,
                     prove(private),
                     same_window]),
            ordering([distance]),
            evaluation([ length>0 means succeed,
                         otherwise continue ])
            ),
    res_step(
            candidates(focus_association),
            filters([]),
            ordering([]),
            evaluation([ length=1 means succeed,
                         otherwise continue ])
            ),
    res_step(
            candidates(accommodation),
            filters([]),
            ordering([]),
            evaluation([ length=1 means succeed,
                         otherwise continue ])
            ),
    res_step(
            candidates(storage),
            filters([]),
            ordering([]),
            evaluation([ length>0 means succeed,
                         otherwise fail ])
            )
```

Suppose this is actually the start of the dialogue. The first resolution step yields no candidates, because the so-called hypothetical context (which corresponds to objects introduced in the antecedent part of sentences of if...then sentences) is empty. Next, candidates are taken from the private context. This yields 5 possible candidates as the system knows 5 different lenses. However, since the common context is empty, none of these pass the `prove(common)` filter. In such cases, the next filter is tried (`observe`). This one actually lets all of the candidates pass, since the only requirement is that they are lenses. The same goes for the final filter, `prove(private)`. So, after filtering we still have all lenses as possible referents. Now ordering takes place. Since there has been no dialogue yet, ordering by dialogue recency does not change the order. But note that in the evaluation, we require the dialogue position of whatever lens happens to be first in the candidate list to be less than 10 objects ago[3]. Objects that do not appear in the dialogue get assigned a position of 1000. Given the evaluation criteria for this resolution step, it also fails (although the algorithm

[3]This 10 is a value that is subject to change on empirical grounds

stores the results temporarily, so that if all else fails, the system can use these objects to offer the user a choice).
The third resolution step starts with getting candidates from the domain. This again yields all 5 lenses, but now we have a filter (`same_window`) that checks which of the lenses is visible in the currently active window and which might let one of the lenses pass while blocking the other 4. This then is an instance of mixing the modalities: a visual clue might influence saliency of objects referred to in a lingual utterance.
Suppose the only lens that is visible in the active window is the mini condenser for which the CTT representation is `mini_condensor-tb-m1:lens`[4], then this will be the referent that needs to be substituted for x. The resulting CTT segment is: $\langle [y : excite - mini\_condensor - tb - m1], [ra = [[y]] \rangle$. In the reaction stage of the response the system will make sure the ra annotation gets taken care of by actually exciting (switching on) the lens. The system then can respond with 'ok' as in the second sentence of the first dialogue.

In the third sentence of the first dialogue, we see a possessive pronoun 'its'. The annotated segment for the sentence is $\langle [t : *, x : t, y : increase - (exc - x)], [rb = [[x, t]], mt = [x : pronoun], ra = [y], tr = [t \leq lens]] \rangle$[5].
Now, the resolution strategy for pronouns is:

```
res_step(
        candidates(hypothetical),
        filters([type_restrictions,not_generalized_subject]),
        ordering([]),
        evaluation([ length=1 means succeed,
                     length>1 means fail,
                     otherwise continue ])
        ),
res_step(candidates(dialogue),
        filters([type_restrictions]),
        ordering([dialogue]),
        evaluation([ length=0 means continue,
                     dialogue_pos_first<10 means succeed,
                     otherwise continue ])
        ),
res_step(
        candidates(accommodation),
        filters([]),
        ordering([]),
        evaluation([ length=1 means succeed,
```

[4]We consistently misspell 'condensor' in the CTT representation.
[5]The annotation tr is used to restrict the variable type t to values consistent with 'lens'. This makes sure the application of the function exc is allowed. This function gives for each lens its excitation and is the translation of the word 'excitation' in conjunction with the use of the possessive in the sentence.

```
                        otherwise fail ])
        )
```

Again, there is no hypothetical context, so the first resolution step does not provide an answer. The second resolution step starts by getting candidates from the dialogue. This means the last 10 objects are considered. This yields the list $[act54 : on - (mini\_condensor - (tb - m1)), mini\_condensor - (tb - m1) : lens]$. The $act54$ object was created by the system when it executed the earlier command and was added to the dialogue context to also allow reference to events via their proofs.

The filter `type_restrictions` checks the types of both candidate objects by comparing it with the tr annotation. Only the second object survives, resulting in a candidate list containing just the mini condenser. Since that has dialogue position 2, it is acceptable and selected as the intended object. This also results in the binding of t to 'lens'. The resulting annotated segment thus is $\langle [y : increase - (exc - (mini\_condensor - (tb - m1)))], [ra = [y]] \rangle$.

Finally we look at the second dialogue. Let us examine in detail the states that the pending context traverses in the course of this small dialogue. We start with the user's utterance 1. The annotated segment corresponding to this utterance is put on the pending context. In the interpretation stage, the resolution algorithm tries to fill in the rb (requires binding) gaps in the annotated segment. In this case, the resolution algorithm fails to do so. The fact that the resolution algorithm does not succeed is stored in the error feature of the annotated segment: $\langle [x : lens, y : on - x], [rb = [[x]], mt = [x : def\_art], ra = [y], error = [uo(x, [])]] \rangle$. The error feature now contains the item $uo(x, [])$. This means that x is an *unresolved object*. The empty list ($i.e.$, $[]$) indicates that the algorithm found no suitable candidates for x.

In the generation stage, the system looks at the top of the pending context and tries to generate a response to the utterance it finds there. In this case, the system produces a clarificatory question. The system constructs an annotated segment which represents this clarificatory question: $\langle [x : lens, y : on - x], [cl\_ri = [x], ra = [y], mean = [pos]] \rangle$. This annotated segment can be paraphrased as follows: 'For which lens do you ask me to switch it on?'. In actual fact, the system produces the elliptical utterance 'Which lens?' and adds the annotated segment to the pending context.[6] So now, we have a pending context containing the user's initial utterance and the system's clarificatory question: $[1, 2]$.

Subsequently, the user's answer to the clarificatory question is added to the pending context: $[1, 2, 3]$. The following annotated segment corresponds to the answer: $\langle [z : lens, p : condenser - z], [rb = [[z]], mt = [z : def\_art]] \rangle$. Now,

[6] More precisely, the pending context consists of indices representing utterances. These indices have several features, amongst which there is one for the annotated segment and another one which records the speaker of the utterance.

before the resolution algorithm is applied, the pending context is first updated. The system tries to unify the representations of the utterances 1. and 3.[7] In this case, it succeeds. The clarificatory question is removed from the pending stack and the revised version 1' of the user's original utterance remains on the pending context: $\langle [x : lens, p : objective - condensor - x, y : on - x], [rb = [[x]], mt = [x : def\_art], ra = [y]] \rangle$. The system now applies the resolution algorithm to this representation. Due to the extra information $(p : condenser - x)$, the algorithm now succeeds.

In the reaction stage, the system looks at the pending context, and finds an annotated segment with the ra (requested action) feature: $\langle [y : on-(objective\_condenser-tb - m1)], [ra = [y]] \rangle$. In other words, the system now has to execute an action such that a proof for the proposition $on-(objective\_condenser-tb-m1)$ comes into existence. The system produces the action in question, i.e., it switches on the condenser lens and reports this to the user saying 'ok'. Simultaneously, the system updates the pending context with a representation of the feedback utterance 'ok'. So now we have: $[1', 4]$. In fact, the system also maintains a record of the relation between utterances, In this case $acknowledge(1', 4)$. If the next utterance of the user is a new (non-clarificatory) question or command, $1'$ and 4 are removed from the pending context. Furthermore, the common context is now updated with the information that the user and the system know that the condensor lens has been switched on.

4 Related research

We now compare our approach to reference resolution with some of the many proposals found in the literature.

Centering Theory and Discourse Structure In Grosz et al. ([1983]), anaphora resolution is tied to the notions of *forward-* and *backward-looking centers.* Centers are (intensional) objects in the world, that may be *realized* by noun phrases i.e. noun phrases may be used to refer to these objects.

The approach taken in the DenK system uses some of results from centering theory. For instance, there is a filter on candidates for pronouns through which those candidates pass that were referred to using a pronoun the previous time. However, contrary to centering theory, we do not assume that every sentence has a single backward looking center. Utterances may be linked to the preceding dialogue by any number of anaphoric expressions. Moreover, since links to the 'preceding context' in a multi-modal system may exist in multiple modal

[7] If the user's response to the clarificatory question cannot be unified with the user's original utterance, the system asks the user to reformulate the answer or stop, as in the following interaction: U: Is this lens on? S: Which lens? U: The magnation lens. S: I do not know the word 'magnation'; try again or stop. U: I mean the magnification lens. S: Yes, the magnification lens is on.

dimensions, it may be that an utterance has no link at all with the preceding utterance. Also, we assume that every object that is mentioned is a forward looking center, except in some cases such as universally quantified objects or objects in *if...then* constructions.

Grosz and Sidner ([1986]) presents a far richer theory than we use in the DenK-system, to account for possible topic-changes in discourse and resulting problems for reference. Although we agree that discourse structure should play a role in reference resolution in general, we feel that for contemporary man-machine interfaces there is no need to take it into account. People using such interfaces will refrain from making interruptions and stay on-topic, since the interface will have a domain-specific, limited vocabulary.

If the role of discourse structure is taken away from Grosz and Sidner's theory, what remains is a theory of reference based on focus spaces. These focus spaces can be identified with the dialogue context and the pending context in our system. As a result, similar predictions will be made by Grosz and Sidner's model and by our model, as far as references to objects introduced in previous dialogue are concerned.

The Capture system The way reference resolution is done in the DenK-system is in some way similar to what was done in the Capture system (Alshawi [1987]). In Capture, noun phrase reference resolution is also done by first selecting a set of entities that satisfy the constraints found in the linguistic expression, and subsequently filtering this set until a unique referent is found (in the case of singular noun phrases). Also, as in Alshawi's system, some constraints are dropped if they are so strong as to discard all candidates.

A difference with Alshawi's system is the use of constraints coming from predicates. In Alshawi's system, a predicate imposes sortal constraints on its arguments, and these constraints can be used to filter the candidate set. However, in the DenK-system, the user is not expected to be an expert in the domain of discourse. This means that the user may not be aware of some of the constraints on predicates. If the system were to use these constraints anyway, it might select the wrong referent. Since we do allow the reference resolution to fail, and have equipped the dialogue manager with ways to handle such situations, we prefer resolution failure above possibly selecting the wrong referent.

The above is also true in case the constraints are not strng enough to identify a unique referent. In Alshawi's system, the algorithm would then make an arbitrary choice. We however prefer to be cautious and have the dialogue manager report interpretation failure.

MMI^2 In the MMI^2 system (Multi Modal Interaction with Man Machine Interfaces for knowledge based systems; Binot et al. [1992]), a different approach to multi-modal reference is taken. There is no context representation for dialogue history. Every interaction, be it lingual, visual or otherwise, is added to

the global history without extracting specialized context updates. The extraction of relevant information is done only when it is needed by some module, for instance reference resolution.
While this has the advantage of saving time, a lot of the work that is needed for extracting context-relevant information from an utternace has to be done anyway in order to be able to determine the meaning of the utterance.
Another advantage of the way the MMI^2 system keeps track of the interaction history, is the flexibility it offers for extending a system with other context-sensitive modules without ever having to change the format of the stored representations. However, this does assume that the meaning format used to store the interactions is itself complete. Since if one were to add another modal dimension, it could be that the reoresentation is not rich enough to cover all possible intricacies of that modality.
Another important difference between the DenK system and the MMI^2 system, is that the latter ignores the differences between the modalities for the purpose of reference resolution. The DenK system on the contrary initially limits its search space to certain contexts based on the kind of referential marker used.

5 Conclusions

We have described the DenK system which can be considered multi-modal in the following respects: it allows for multi-modal input from the user (typed input and pointing) and can generate multi-modal output (typed output and higlighting). The system operates in a setting which makes multi-modal interaction a natural way of interacting: user and system work on and talk about a domain which both can observe.

In our paper, we describe in detail how the system deals with resolution of referential expressions in such a multi-modal setting. This is done by specifying an algorithm which can look for objects in the different contexts that are present in such a setting. In this respect, our approach extends much of the research on resolution of referential expressions (which concentrates mainly on discourse) in two respects: we deal i. with dialogue and ii. with a situation where objects may be located in a domain which is external to the cognitive states of the interlocutors.

Furthermore, the system is fitted with a mechanism to provide cooperative feedback through clarification questions and to re-evaluate ambiguous utterances on the basis of the answers given by the user to these questions.

Finally, we would like to point out that a paper version of the algorithm has been evaluated on the basis of naturally occurring dialogues. It turned out that the predictions made by the algorithm are closely matching actual dialogue behaviour (see Verbeem [1996]).

References

[1987] Alshawi, H. (1987) *Memory and context for language interpretation*, Studies in Natural Language Processing, Cambridge University Press.

[1995] Ahn, R., Beun, R.J., Borghuis, T. Bunt, H. & Van Overveld C. (1995) 'The DenK-architecture: a fundamental approach to user-interfaces', *Artificial Intelligence Review* **8** *(3)*, 431-445.

[1990] Ahn, R. & Kolb, H-P. (1990) 'Discourse Representation meets Constructive Mathematics', in László Kálmán & László Pólos (eds), *Papers from the Second Symposium on Logic and Language*, Akademiai Kiado, Budapest, 105-124.

[1992] Barendregt, H.P. (1992) 'Lambda Calculi with Types', in S. Abramsky, D. Gabbay and T. Maibaum (eds), *Handbook of Logic in Computer Science*, Oxford University Press, Oxford.

[1995] Beun, R.-J. & Kievit, L.A. (1995) *Resolving definite expressions in DenK*, DenK report 95/16, SOBU, Tilburg–Eindhoven.

[1997] Beun, R.J. & Piwek, P. (1997) *Pragmatische features in DenK: PRAGTAGS*, DenK Report 97/29, SOBU, Tilburg–Eindhoven.

[1992] Binot, J.L., Debille, L., Sedlock, D. & Vandecapelle, B. (1992) 'Multimodal Integration in MMI^2: Anaphora Resolution & Mode Selection', in H. Luczak, A. Cakir and G. Cakir (eds.), *Work With Display Units–WWDU'92*. Berlin: Technische Universität - Institut für Arbeitswissenschaften.

[1980] De Bruijn, N.G. (1980) 'A Survey of the Project Automath', in J.R. Seldin & J.P. Hindley (eds), *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalisms*, Academic Press, 589-606.

[1983] Grosz, B.J., Joshi, A.K. & Weinstein, S. (1983) 'Providing a Unified Account of Definite Noun Phrases in Discourse', in *21st Annual Meeting of the Association for Computational Linguistics / Proceedings of the Conference.*

[1986] Grosz, B.J. & Sidner, C.L. (1986) 'Attention, Intentions and the Structure of Discourse', *Computational Linguistics* **12** *(3)*, 175 -204.

[1993] Kamp, H. & Reyle, U. (1993) *From Discourse to Logic: Introduction to Model-theoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory*, Kluwer Academic Publishers, Dordrecht.

[1984] Martin-Löf, P. (1984) *Intuitionistic Type Theory*, Bibliopolis, Naples.

[1995] Piwek, P. (1995) *Annotated Type Theory for Information Representation*, DenK Report 95/17, SOBU, Tilburg–Eindhoven.

[1996] Verbeem, M. (1996) *An evaluation of Resolve*, DenK report 96/28, SOBU, Tilburg–Eindhoven.