Contextual Reasoning in Multimodal Dialogue Systems: Two Case Studies

Johan Boye, Mats Wirén and Joakim Gustafson
Voice Technologies
TeliaSonera Sweden
{Johan.Boye|Mats.Wiren|Joakim.Gustafson}@teliasonera.com

Abstract

This paper describes an approach to contextual reasoning for interpretation of spoken multimodal dialogue. The approach is based on combining recency-based search for antecedents with an object-oriented domain representation in such a way that the search is highly constrained by the type information of the antecedents. By furthermore representing candidate antecedents from the dialogue history and visual context in a uniform way, a single machinery (based on β-reduction in lambda calculus) can be used for resolving many kinds of underspecified utterances. The approach has been implemented in two highly different domains.

1 Introduction

This paper describes an approach to contextual reasoning and its application to two radically different domains, both of which make use of spoken multimodal dialogue. The first system is ADAPT, which allows the user to look for apartments for sale in central Stockholm (Bell et al. 2001). Apartments are represented in a relational database and are displayed as icons on an interactive map. The second system is the NICE fairytale game, in which the user collaborates with an animated character to solve a problem in an immersive 3D world (Gustafson et al. 2004a).

Both domains are, each in its own way, sufficiently restricted that no serious problems are posed by lexical or structural ambiguity. Likewise, the use of quantification is limited and only rarely leads to ambiguity problems. In contrast, interaction in both domains abound with deictic and anaphoric expressions such as pronouns, definite descriptions and ellipses. These expressions refer to things in the visual surrounding as well as to objects that have been mentioned in the previous dialogue. Thus, all utterances have to be interpreted by way of reasoning about the objects in the combined dialogue and visual context.

Although naïve reference resolution methods — such as preferring the most recent grammatically compatible antecedent — may perform remarkably well (see Hobbs 1978, Mitkov 1998), dialogue applications typically must bring more knowledge into play in order to perform well. Often some logic-based reasoning using a representation of the task and domain is adopted. An early example of this is the focus representation of Grosz (1977), based on the partitioned semantic networks of Hendrix (1975). Another example is the resolution component of the CLE (Alshawi 1992). However, there is a computational price to be paid for the general semantic reasoning-based methods. Particularly for limited domains, it seems that it would be useful to find a less complex approach.

The approach described here replaces general-purpose semantic reasoning with a much more restricted type system. By combining this with the recency principle — thus looking for the most recently referred object of a compatible *type* — we obtain a highly efficient but still accurate search strategy. This approach works well in two fairly complex domains to be further described below. Moreover, by representing candidate antecedents from the dialogue history and visual

context in a uniform way, the same machinery can be used for resolving many kinds of underspecified utterances.

2 Systems and domains

2.1 The ADAPT system

The graphical user interface of the apartment database system ADAPT is shown in Figure 1. The system features an animated talking agent providing the user with information about apartments currently for sale. The system also displays the geographic locations of apartments under discussion as clickable icons with distinct colours on a map. Graphical input and the textual result from the speech recognizer are jointly interpreted by the system.

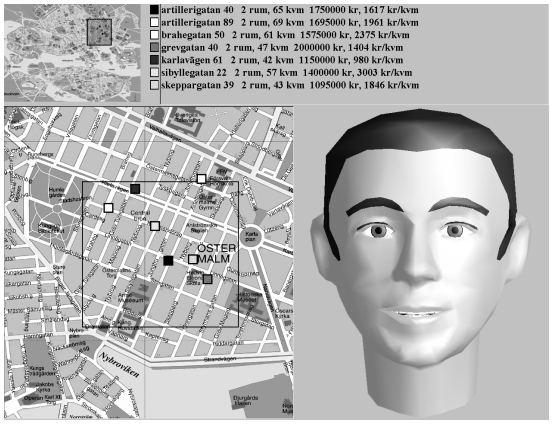

Figure 1. The graphical interface of the ADAPT system.

The following is a typical dialogue fragment with the ADAPT system.

A1. **User:** Are there any two-room apartments on the South Side that cost less than 2 million?
A2. **System:** Do you have any other requirements?
A3. **User:** A balcony would be nice.
A4. **System:** [Displays colored icons on the map] There are seven such apartments. They are shown on the map.
A5. **User:** I see… The green apartment… how much does it cost?
A6. **System:** The green apartment costs 1.5 million.
A7. **User:** What about the red one?
A8. **System:** The red apartment costs 1.9 million.
A9. **User:** Okay… Is there anything cheaper?

Note that even this short dialogue contains a fair variety of anaphora: pronouns, definite descriptions (often using the distinct colours of the displayed icons), elliptic questions (utterance A7) and elliptic answers to questions (A3). Also, graphical pointing actions (of which there are no instances in the particular dialogue here) are often accompanied by deictic noun phrases.

2.2 The NICE fairy-tale game

The NICE fairy-tale game system lets the user enter into an animated 3D world and communicate with a number of animated characters by means of spoken dialogue. So far, the first scene of the game has been implemented, featuring a single character, called Cloddy Hans (inspired by one of the characters of an H C Andersen fairytale). This first scene takes place in Andersen's authoring laboratory, where the key device is a fairytale machine. The machine has a magic quality, namely, that if one inserts objects into its designated slots (for "magic things", "dangerous things", and so on) and pulls a lever, it creates a new fairytale in which the objects come to life. Since Cloddy Hans would like to create a fairytale of his own, he needs help from the user in understanding where to put different things. The first scene therefore develops into a collaborative "put that there" task, where Cloddy Hans and the user have to agree on what to do. Figure 2 shows a screen shot from the game.

Figure 2. A screen shot from the NICE system. Cloddy Hans is standing in front of the fairy-tale machine, next to the shelf with objects.

The following fragment is from a dialogue with an eleven-year old user.

N1. **User:** I want you to go to the shelf.
N2. **Cloddy Hans**: Okay. [Walks over to the shelf.] What do you want me to do now?
N3. **User:** I want you to pick up the bag.
N4. **Cloddy Hans:** Do you want me to pick up the money sachet?
N5. **User:** Yes, pick up the sachet. [Clicks on the money sachet.]
N6. **Cloddy Hans:** [Picks up the sachet.] That went well!
N7. **User:** Then I want you to go to the slots.
N8. **Cloddy Hans:** Okay. [Walks over to the machine.]
N9. **User:** Now I want you to put the money sachet in the farthest slot.

Both systems allow users to refer to objects graphically by a mouse-compatible device. However, while users of ADAPT can scroll the map as they please, users of the NICE system cannot manipulate the camera at will. Rather, the camera follows Cloddy Hans as he moves around, and hence situations will arise where graphical reference is not possible.

2.3 Corpora

The ADAPT corpus is based on one Wizard-of-Oz collection with 16 subjects (Bell et al 2001), and one independent data collection with 24 subjects using the fully functional system (Edlund and Nordstrand 2002). The user tasks were to find an apartment obeying certain constraints (in the former case) and to find ones that the subject might want to live in (in the latter case).

The current NICE corpus is based on a semi-automated Wizard-of-Oz collection with 10 children aged 11–15 (Gustafson et al. 2004). The subjects were informed about the scenario described in the previous section and were instructed to collaborate with Cloddy Hans to put some (unspecified) things into the machine. (We have not yet collected any data using the existing, fully functional system corresponding to this scenario.)

All corpora examples in this paper have been translated from Swedish to English by the authors.

3 Referential phenomena

3.1 Knowledge sources

One way of studying referring expressions for the purpose of developing focus management and reference resolution is to look at what knowledge sources are needed to interpret them.

Perhaps the most obvious knowledge source in a graphics-based multimodal system is the *visual context*. Two examples of this are utterances A5 (”the green apartment”) and N9 (”the slot furthest away”) in the dialogue fragments in Section 2. As can be seen, definite descriptions include both visually salient properties and (in NICE) the relative position of 3D objects. Descriptions of the latter often include complex ordinal and directional expressions, like “the third tube from the left” or “the hole which is second from the right”. (Whereas currently each object in the shelf is unique, the four slots in the machine have to be distinguished by means of some other property.)

Another obvious knowledge source is the *preceding dialogue*, for example, utterance A9 (”anything cheaper”). Here the user expresses a desire which refers to the price of a previous apartment (presumably the green one).

Sometimes a record of *past events* is also needed to resolve a reference, as shown by the following example from the NICE corpus:

N10. **User:** Where we put the magic wand… there you can put it.

Here the clause ”Where we put the magic wand” is referring to a slot of the fairytale machine via its relation to a previous action.

Finally, a model of the *domain* is needed, as shown in the following example:

N11. **User:** I want you to take the hammer.
N12. **Cloddy Hans**: Okay. [Takes the hammer.]
N13. **User:** Then I want you to go to the machine… And put it in the first tube.

Here, it is obvious that “it” in utterance N13 corresponds to the hammer because of the way the particular objects and actions are related in this domain. However, a naive recency-based

model without this information would rather associate ”it” with the machine.[1]

Summing up, all of the above knowledge sources frequently come into play in ADAPT and NICE, with the exception of past events that are only rarely used.

3.2 Referential usage

A problem which is complementary to the one above is how referential expressions are constructed depending on which knowledge source is involved. In particular, how do people refer to objects that are present in the visual display but that have not yet been referred to in the dialogue? Also, to what extent are objects *outside* of the current visual display referred to? This is important for the purpose of determining how the current focus should be updated with respect to objects from the visual environment.

As for the first question, people frequently use definite descriptions to refer to visually displayed objects right from the first turn of the dialogues, without the objects ever having been mentioned. Both in ADAPT and NICE, there is a variety of characteristic properties that can be combined to describe objects — for example, in ADAPT the colour of the icon, the number of rooms and the street of the apartment, etc.

Pronouns are sometimes used without previous mentioning of the referred object in the dialogue, but then only in combination with a graphical pointing action:

N14. **User:** Go to the shelf.
[Cloddy Hans confirms and walks up to the shelf.]
N15. **User:** [Graphical pointing at diamond.] Take it.

As for objects outside of the visual display (the second question above), and looking first at ADAPT, it is clear that the set of apartment icons provides an extremely strong cue for the mutually grounded context: Although users frequently change their desired apartment constraints back and forth as they explore the search space, there is no instance of a user going back and referring verbally to a particular apartment that is no longer displayed on the map. Thus, in our data the objects under discussion are always those that are shown on the map. Similarly, there are no references to previous events (“Go back to the area where we were previously”).

In NICE, the situation is different because of the moving camera and the fact that the set of objects remains constant except when something is put into the machine. Here, users do refer to things currently outside of the visual display, like the fairytale machine and the shelf. Even objects that are no longer physically present in the scene may be referred to, as in utterance N10 above.

4 Contextual interpretation

4.1 The problem

The problem of contextual interpretation can be divided into three subproblems. First, expressions that refer to the context must be recognized in the input. For spoken input, this is not trivial, since state-of-the-art speech recognizers often fail in recognizing short (function) words, such as pronouns. Secondly, there is the issue of finding the set of candidate objects on which the interpretation of the input can be based — that is, computing the right context in which to interpret the utterance. We call this focus management. Thirdly, there is the issue of combining the contextual information with the information conveyed in the utterance to produce the final interpretation. It is well-known that the two last steps can be arbitrarily difficult (see e.g. Hobbs 1978).

As for the first subproblem, we have shown in a previous paper how spoken input exhibiting a large amount of anaphoric and deictic expressions can be efficiently parsed in a limited domain (Boye and Wirén 2003a). As for the third subproblem, our semantic representation is designed so as to let all contextual interpretation be realized by a uniform process of β-reduction in lambda calculus. This representation is described in detail in Section 5. The rest of this section deals mainly with the second subproblem: how to determine the set of objects that, at each moment, constitute the possible targets for interpretation of referring expressions. We call such objects *salient* objects.

[1] In Swedish, “hammer” and “machine” have identical gender, and hence the pronoun agrees grammatically with both of them.

4.2 Apartment domain

To begin with, the ADAPT system must distinguish internally between *intensional* and *extensional* objects. Whenever the user starts over by giving new constraints (as in utterance A1), a new intensional object is created. This object is considered to be salient until a set of concrete apartments is presented to the user (as in utterance A4). These apartments are represented internally as extensional objects and are considered salient as long as their icons are displayed on the map (i.e. until the user has asked for a new set of apartments). Then a new intensional object is created, and the whole cycle is repeated. In general, this kind of distinction must be made by any system in which the dialogue begins by specifying an "ideal" object before matching it with real ones.

As for displayed objects and their relation to the current context, it turns out that the basic mechanism for updating the set of salient object can be made very simple: As discussed in Section 3.2, the set of displayed apartment icons provides both necessary and sufficient information to determine the set of salient apartments. This approach is the same as the one taken in Cheyer and Julia (1995).

To handle implicitly focused items, objects of the domain are represented by means of a type hierarchy, motivated by the characteristics of the domain (this is further described in Section 5). This allows us to handle utterances like A12 below:

A10. **User:** How many rooms does the green apartment have?
A11. **System**: Three rooms.
A12. **User:** What is the monthly fee?

Here, "monthly fee" will be associated with an attribute of the relevant apartment object.

In many cases, referential expressions in the ADAPT domain turn out to be unambiguous (as in utterances A5 and A7 in Section 2.1). In those cases which remain ambiguous, a straightforward recency principle works in the vast majority of cases (that is, preferring type-compatible antecedents that appear at shorter linear distance backwards in the dialogue).

In some easily distinguishable special cases, other rules apply. An example of this is utterance A9, where the desired price should be less than all the previously discussed prices of the apartments in focus.

4.3 Fairy-tale game domain

Looking at the introductory fairy-tale scenario described in Section 2.2, our data so far indicate that it is sufficiently restricted to be amenable to the same basic methods as those used in ADAPT. First, the set of objects that can be referred to is limited and can be kept constant from the point of view of the visual context in which the user's utterance is to be understood. (Although objects disappear from the physical environment when they are put into the machine, they may still be referred to as exemplified by N10 above.) Secondly, the limited amount of moving of the camera also does not require any corresponding shifting of the visual context, as discussed in Section 2.2.

For these reasons (and in contrast to ADAPT), we do not make use of any mechanism for updating the visual context, but rather keep all objects from the scene constantly in the current context.[2] This includes the objects initially situated in the shelf, the shelf itself, the fairy-tale machine as well as relevant parts and properties of these, like the slots of the machine and the symbolic labellings of each slot.

Clearly, however, this simple strategy will not be tenable in the succeeding scenes of the game (currently under implementation). Here, the changing scenes will require updating of the visual context as the user freely moves about in the large 3D world. We will return to this scenario in a later paper.

5 Representation and implementation

As mentioned above, objects in both the ADAPT and NICE domains are represented by means of a type hierarchy in a standard object-oriented fashion, much the same way one would represent the domain in the Java™ programming language.

[2] This seems to be the approach taken also by Lemon et al (2001), whose system does not use an explicit internal representation of the visual context.

Specifically, this means:

- Every object belongs to exactly one type.
- A type may be a direct subtype of exactly one type.
- An object may have any number of attributes, whose values are objects of the appropriate types.[3]

For instance, in the fairy-tale system, objects that can be moved about belong to the type thing. Things have an attribute position whose values should belong to the type location. So, supposing that hammer and axe are things, and onShelf is a location, the fact that the hammer is lying on the shelf is representable, whereas the fact that the hammer is lying on the axe is not (since the equation hammer.position = onShelf obeys the type constraints whereas hammer.position=axe does not).

A slot (in the fairy-tale machine) is a special kind of location; hence the type slot is a subtype of location. This means that hammer.position can be given values also of type slot.

This object-oriented approach to coding the domain extends also to actions, events, dialogue acts, and so on. For instance, the action of picking up something is represented by an object of type pickUp having two attributes; agent of type (fairy-tale) character, and patient of type thing.

5.1 Representation of user utterances

During execution, user utterances are translated by a parser into typed combinators[4] over the domain model (see further Boye and Wirén 2003a, 2003b). In the NICE system, utterances are translated into expressions of type dialogue_act (request, ask, tell, and so on). As an example, utterance N3 would be translated into

$$\text{request(user, cloddy, pickUp(cloddy, bag))}$$

whereas the utterance "Pick it up" would be translated into

$$\lambda x^{\text{thing}}.\text{request(user, cloddy, pickUp(cloddy, x))}$$

Here, superscripts indicate the types of variables. Thus, the expression above denotes a function taking a thing as the argument, returning the fully instantiated request as the result. Here the domain model is used to infer that the missing information (the object x being picked up) is of type thing.

Resolution of the reference "it" now corresponds to applying[5] the function above to an expression of the appropriate type, e.g.

$$(\lambda x^{\text{thing}}.\text{request(user, cloddy, pickUp(cloddy, x))}\ \text{bag}) \rightarrow \text{request(user, cloddy, pickUp(cloddy, bag))}$$

Thus, the type constraints in the domain model help ruling out undesired interpretations of references in user utterances.

The ADAPT apartment system seeks to translate all user utterances into the form $?x^t(P)$, which can be paraphrased as "Give me x of type t such that P is true". Again, lambda abstractions are used to represent missing information. For instance, utterance A5 would be:

$$\lambda x^{\text{apartment}}\ ?p^{\text{money}}\ (x.\text{price} = p\ \&\ x.\text{color} = \text{green})$$

Supposing that apt1 denotes the apartment the user is referring to in utterance 5, then contextual interpretation of this utterance amounts to applying the functional expression to apt1:

$$(\lambda x^{\text{apartment}}\ ?p^{\text{money}}\ (x.\text{price} = p\ \&\ x.\text{color} = \text{green})\ \text{apt1}) \rightarrow ?p^{\text{money}}\ (\text{apt1.price} = p\ \&\ \text{apt1.color} = \text{green})$$

The resulting expression is paraphrased ``Give me the price of apt1", and can be translated straightforwardly into a database search command.

One of the nice features of this representation scheme is that various kinds of anaphora and ellipses can be handled the same way. For example, in utterance A7 it is evident that the user wants to know something about the red apartment, but it is not clear (before consulting the context) exactly what he wants to know. Such

[3] This representation scheme is thus significantly less expressive than e.g. the *partitioned semantic networks* by Hendrix (1975) (used by Grosz, 1977) , which are equivalent to first-order logic.

[4] A *combinator* is a lambda-expression without free variables (see e.g. Hindley and Seldin 1986). For an approach to natural language semantics based on combinators, see Jacobson (1999).

[5] Application of the lambda expression f to the argument a, so-called *β-reduction,* is denoted (f a). Another commonly used notation is f@a.

elliptic utterances are represented using higher-order lambda expressions:

$$\lambda x^{apartment}\ \lambda f^{apartment \rightarrow dialogue\_act}\ (f\ x[x.color = red])$$

Here, x[x.color=red] is a constrained variable, i.e. x can only take values such that x.color=red is true. In this case, contextual interpretation amounts to first applying the above expression to the appropriate apartment (whose icon should be red), and then applying the resulting expression to a function f, expressing what to do with the red apartment. Supposing apt2 denotes the red apartment the user is referring to, then first applying the above expression to apt2 yields:

$$\lambda f^{apartment \rightarrow dialogue\_act}\ (f\ apt2)$$

We will discuss how to find functional antecedents in section 5.2; for now we will just stipulate that the correct antecedent is

$$\lambda y^{apartment}\ ?p^{money}\ (y.price = p)$$

since

$$(\lambda f^{apartment \rightarrow dialogue\_act}\ (f\ apt2)\ \ \lambda y^{apartment}\ ?p^{money}\ (y.price = p)) \rightarrow$$

$$(\lambda y^{apartment}\ ?p^{money}\ (y.price = p))\ \ apt2) \rightarrow$$

$$?p^{money}\ (apt2.price = p)$$

i.e. "How much does apt2 cost?".

5.2 Focus management

To keep track of which objects are potential targets for reference resolution, the ADAPT and NICE systems use several internal data structures.

The *visual context history* is a recency-ordered list of sets of objects, each set corresponding to a visual context. In ADAPT, each set consists of apartments whose icons are shown simultaneously on the map. Each time some icons are added or removed, a new visual context is created and added to the history. The visual context is used for resolving definite NPs like "the green apartment", and metonymies like "King's street".

As previously mentioned, the visual context is kept constant in the first scene of the fairy-tale system.

The *dialogue history* is a recency-ordered list of typed combinators, each combinator representing a (resolved) user utterance or a system utterance. The dialogue history is mostly used to resolve pronouns and ellipses, by searching backwards in the list for a (sub-)expression of compatible type. For instance, consider utterance N13, which is represented as two dialogue acts:

request(user, cloddy, goTo(cloddy, atMachine))

$$\lambda x^{thing}.request(user, cloddy, putDown(cloddy,x,magicSlot))$$

In order to find an argument of type thing, we have to go back to the representation of N11:

request(user, cloddy, pickUp(cloddy, hammer))

Here, the expression hammer is of type thing, and is indeed the expression needed to correctly resolve the reference in N13. Thus, the typing of expressions prevents unwanted resolutions (like resolving "it" by "the machine" in N13).

Resolution of certain kinds of ellipses involves finding a function of the appropriate type. To resolve utterance A7 (as discussed in section 5.1) the system must find a function of type apartment→dialogue_act. This is computed by a technique reminiscent of Dalrymple et al (1991). First abstraction (reverse functional application) from the resolved representation of the preceding user utterance A5 gives us[6]:

$$?p^{money}\ (apt1.price = p\ \&\ apt1.color = green) \rightarrow^{-1}$$

$$(\lambda y^{apartment}\ ?p^{money}\ (y.price = p\ \&\ apt1.color = green)\ \ apt1)$$

By removing the redundant constraint apt1.color = green, we can extract the combinator needed to resolve A7, namely

$$\lambda y^{apartment}\ ?p^{money}\ (y.price = p)$$

[6] In this particular case, there are several possible abstractions; any one or both occurrences of apt1 can be replaced by the variable y. The particular abstraction shown here is preferred by the system, since it does not create a constraint y.color = green. Such a constraint would be inconsistent with the representation of utterance A7; "What about the red one?".

Finally, an *event history* will be added to the NICE system to be able to resolve references like the one in utterance N10.

6 Discussion

This paper describes an approach to contextual reasoning for the interpretation of spoken multimodal dialogue which refrains from general-purpose reasoning and instead uses a much more restricted type system. By combining type information with a recency principle, we obtain a search strategy which is both highly efficient and accurate. By driving interpretation with respect to both the dialogue history and visual context by a process of β-reduction, we obtain a single, uniform machinery for contextual interpretation which is applicable to the resolution of many kinds of underspecified utterances, such as deictical expressions, anaphora and ellipses. Put differently, the search strategy amounts to finding correct arguments for the typed combinators representing user utterances.

Current work is mainly directed towards extending the NICE system to include the subsequent scenario taking place in the virtual fairy-tale world. To this end, the approach described here will have to be generalized. In particular, the visual context will require frequent updating as the user freely moves around in the large 3D world instead of being confined to a single room. There might also be a need for keeping track of the visual context at the time of previous utterances in order to correctly determine antecedents. We expect to report more on these aspects as part of future work.

References

Alshawi, H. (1992) *The Core Language Engine.* The MIT Press.

Bell, L., Boye, J. and Gustafson, J. (2001) Real-time handling of fragmented utterances. *Proc. NAACL workshop on adaptation in spoken dialogue systems.*

Boye, J. and Wirén, M. (2003a) Robust parsing of utterances in negotiative dialogue. *Proc. Eurospeech.*

Boye J. and Wirén, M. (2003b) Negotative spoken-dialogue interfaces to databases. *Proc. Diabruck, 7th workshop on the pragmatics and semantics of dialogue.*

Cheyer, A. and Julia, L. (1995) Multimodal Maps: An Agent-based Approach. *International Conference on Cooperative Multimodal Communication (CMC/95)*, 24–26 May 1995, Eindhoven, The Netherlands.

Dalrymple, M., Shieber, S. and Pereira, F. (1991) Ellipsis and higher-order unification. *Linguistics and Philosophy*, vol. 14, no. 4, pp. 399–452.

Edlund, J. and Nordstrand, M. (2002) Turn-taking Gestures and Hour-Glasses in a Multi-modal Dialogue System. *ISCA workshop on Multimodal Dialogue in Mobile Environments*, Kloster Irsee.

Grosz, B. (1977): The representation and use of focus in a system for understanding dialogs. *Proc. IJCAI*, pp. 67–76.

Gustafson, J., Bell, L., Boye, J., Lindström, A. and Wirén, M. (2004a) The NICE fairy-tale game system. *Proc. SIGDIAL*.

Gustafson, J., Boye, J., Bell, L., Wirén, M., Martin, J.-C., Buisine, S. and Abrilian, S. (2004 b). Collection and analysis of multimodal speech and gesture data in an edutainment application. NICE Deliverable D2.2b. http://www.niceproject.com.

Hendrix, G. (1975) Expanding the utility of semantic networks through partitioning. *Proc. IJCAI*, pp. 115–121.

Hindley, R. and Seldin, J. (1986) *Introduction to combinators and λ-calculus*. Cambridge University Press.

Hobbs, J. (1978) Resolving pronoun references, *Lingua*, vol. 44, pp. 311–338.

Jacobson, P. (1999) Towards a variable-free semantics. *Linguistics and Philosophy* 22, pp. 117–184.

Lemon, O., Bracy, A., Gruenstein A., and Peters, S. (2001) Information states in a multi-modal dialogue system for human-robot conversation. *Proc. Bi-Dialog, 5th workshop on formal semantics and pragmatics of dialogue* , pp. 57–67.

Mitkov, R. (1998) Robust pronoun resolution with limited knowledge. *Proc. COLING/ACL'98*, pp. 869–875.