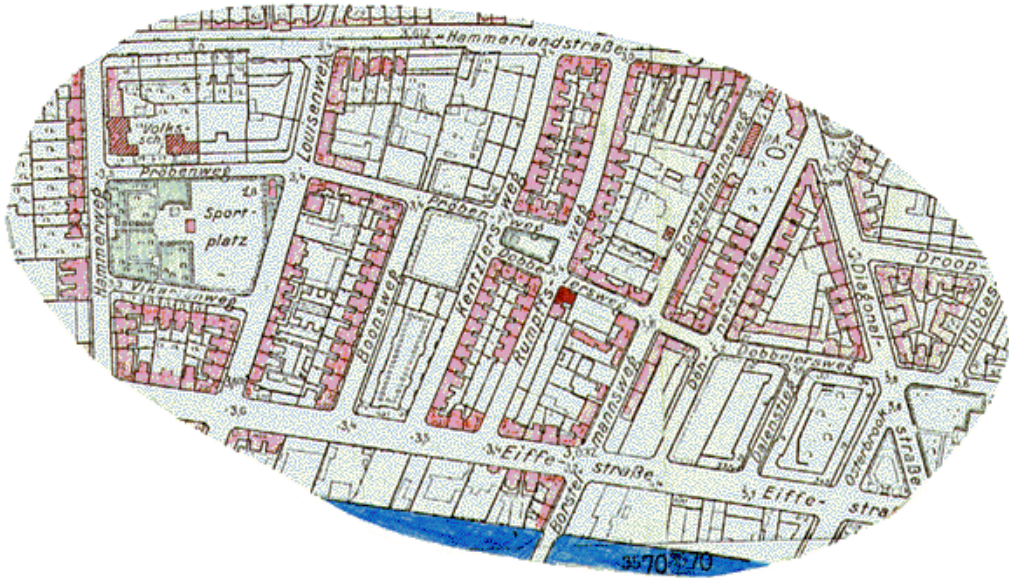


Designing an Anaphora Resolution Algorithm for Route Instructions

Mareile Hillevi Knees



M.Sc. in Cognitive Science and Natural Language

Division of Informatics

University of Edinburgh

2002

Abstract

The ambition of this dissertation is to develop an anaphora resolution algorithm for route instructions which are taken from the Instruction-Based-Learning (IBL)-project. Route instructions describe spatial features of the environment of the route and they also specify temporal information about the actions (e.g. movements, turns) to be performed. Thus, they need to be interpreted with respect to the spatial context reference and domain-specific knowledge. References to the context are not only created by noun phrases (referring to entities) or verb phrases (referring to eventualities), they are also built up by anaphors for example personal or demonstrative pronouns.

In the IBL-data a variety of anaphoric references to the spatial context occur. Since most of the anaphoric references in the data were created by the anaphors '*it*', '*this*', '*that*' and 'This-NP', this dissertation especially deals with these types of anaphors. In order to resolve these anaphors, I developed the Route-Instruction-Anaphora-Resolution-Algorithm (RIAR). Due to the fact the route instructions specify spatial information of the environment, two other anaphors (the spatial anaphors '*here*' and '*there*') frequently occur in the IBL-data. Thus, I also designed the Spatial-Reference-Algorithm (SR). The SR-algorithm is the first algorithm which resolves the spatial anaphors '*here*' and '*there*'.

After analysing the given data and creating a gold standard, the conceptual design of both algorithms is evolved with respect to recent research. The RIAR-algorithm integrates concepts of the most recent algorithm developed by Eckert & Strube (2000) and ideas from a recent work by Byron (2002) which includes semantic filtering in addition to the common salience calculations inspired by the Centering Theory. In order to resolve the spatial anaphors '*here*' and '*there*' the SR-algorithm takes into account that these anaphors refer to positions described by prepositional phrases rather than to entities realized as noun phrases or clauses.

Since neither algorithm is implemented, they are only theoretically tested on a set of unseen data in order to evaluate their performance. The RIAR-algorithm performs quite well. It resolves 89% of all personal pronouns, 60% of all

demonstrative pronouns and 86% of 'This-NP'-anaphors. The SR-algorithm does not perform as well. It only correctly resolves 43% of all 'here/there'-anaphors. This leads to the conclusion that the SR-algorithm needs further refinements. However, the RIAR-algorithm performs better than the Eckert & Strube-algorithm which only correctly resolves 53% of all personal pronouns and 57% of all demonstrative pronouns of the development data. Finally, unsolved problems are discussed and suggestions for further work are given.

Table of Contents

1. Introduction	5
2. Background and Related Work	7
2.1 Route Instructions	7
2.2 Discourse and Anaphoric Reference	8
2.3. Anaphora Resolution	11
2.3.1 <i>Focusing and Centering Theory</i>	11
2.3.2 <i>Eckert & Strube – Algorithm</i>	13
2.3.3 <i>The PHORA-Algorithm</i>	18
2.3.4 <i>The THIS-NP Hypothesis</i>	20
3. RIAR- and SR-Algorithms	23
3.1 Defining the RIAR-Algorithm	23
3.2 Discourse Model and Units	24
3.3 Ontology, Lexicon and Selectional Restrictions	28
3.4 Salience-List, Implicit S-List and Abstract-Object-List	35
3.5 Resolution Rules of the RIAR-Algorithm	37
3.6 Here- and There-Anaphors and the SR-Algorithm	46
4. Evaluation and Discussion	49
4.1 Evaluation of Eckert & Strube’s-Algorithm on the Development Data	49
4.2 Evaluation of the RIAR- and SR-Algorithm on the Development Data	51
4.3 Evaluation of the RIAR- and SR-Algorithm on the Test Data	52
5. Conclusions and Further Work	57
5.1 Summary	57
5.2 Open Issues	58
5.3 Further Work	61
Bibliography	65
Appendices	66

1. Introduction

The aim of the project is to develop an anaphora resolution algorithm for spoken route instructions in English which direct a mobile robot on a route through a modelled world (see Appendix 1). The route instructions used to design the anaphora resolution algorithm are taken from the Instruction-Based-Learning (IBL) corpus. The IBL-project tries to realize the idea that robots can be instructed by language, so that users do not need to program their robot because it can be programmed by language. Since this aim is quite complex, the project is restricted to the domain of route instructions, given as real speech input to a robot which uses vision for navigating in a small modelled world. Thus, the results of the project: *Designing an Anaphora Resolution Algorithm for Route Instructions* might be useful for the IBL-project.

As route instructions describe the spatial environment of the route and as they denote the actions to be performed, they need to be interpreted with respect to the spatial context reference and domain-specific knowledge. References to the context are not only created by noun phrases (referring to entities) or verb phrases (referring to eventualities), they are also built up by anaphors for example personal or demonstrative pronouns. Most of the anaphoric references in the IBL-data are created by the anaphors ‘it’, ‘this’, ‘that’ and ‘This-NP’ which refer to either the linguistic and spatial context mentioned in the route instruction. Thus, the aim of the dissertation is to develop an Route-Instruction-Anaphora-Resolution-Algorithm (RIAR), which resolves these anaphors. Moreover, the spatial anaphors ‘here’ and ‘there’ frequently occur in the IBL-data, since route instructions specify much spatial information of the environment. Therefore, I also designed a Spatial-Reference-Algorithm (SR). The SR-algorithm is the first algorithm which resolves the spatial anaphors ‘here’ and ‘there’.

In order to develop and test the newly designed anaphora resolution algorithms for route instructions (the RIAR-algorithm and the SR-algorithm), the IBL-corpus is split into two sets. The first set is the development data used in order to design the rules for both algorithms. The second set is needed later to test the performance of the algorithms.

To start with, the anaphoric expressions occurring in the IBL-corpus needed to be identified and classified. Since there are many anaphoric references occurring in the IBL-corpus, I had to determine what kind of anaphoric references the algorithm should handle in the framework of this project. Thus, I will shortly elucidate the types of anaphoric reference the algorithms deal with. The RIAR-algorithm is designed to resolve the personal and demonstrative pronouns *it*, *this* and *that* which either refer to individual entities (e.g. *the road*) or abstract entities (e.g. *turn left*). Furthermore, it deals with so-called ‘This-NP’-anaphors (cf. Poesio & Modjeska 2002) which tend to refer to individual entities or to implicit entities introduced by previously mentioned entities. Even though Poesio & Modjeska (2002) only describe cases in which elements or subsets of plural objects implicitly introduce a plural object into the discourse model, the IBL-dataset implies that the reverse also applies, which means that plural objects also implicitly establish their elements or subsets in the discourse model. The other algorithm, the SR-algorithm, resolves references to spatial locations expressed in anaphors like *here* and *there*. The referents of those anaphors are built up by the combination of the verb and the spatial preposition of the previous utterance. Since there are plenty of possible combinations it is quite difficult to define rules for resolving ‘here/there’-anaphors.

To develop the algorithms a gold standard of resolved anaphoric expressions was created. For the conceptual design and for the processing of the algorithms it was useful that the IBL-system is already in charge of an implemented DRT (discourse representation theory) component, so that the algorithms can make use of the existing lexicon and ontology. The third part of this dissertation gives details about the conceptual design and the processing of both algorithms.

Then, the algorithms are tested on the second set of the IBL-corpus (the unseen data). The fourth part of this work contains an evaluation of the algorithms and a discussion of the most common problems observed during the testing phase.

Finally, the conclusion section considers unsolved problems, and suggestions for further work are made.

2. Background and Related Work

As the algorithm is mainly based on theoretical considerations, the dissertation is closely connected with recent research in the field of anaphora resolution. As a consequence, the second part presents findings of some of the most recent research in more detail.

2.1 Route Instructions

Route instructions communicate information between participants with the goal to solve the problem ‘how to come from A to B’. They consist of descriptive elements (e.g. “*next to the green building*”) which give information about spatial relations between entities or which refer to properties of entities. They are also composed of instructive elements (e.g. “*turn left*”) which indicate actions like movements and changes of orientation (e.g. verbs like ‘go’, ‘turn’). In route instructions different segments of the route are described: firstly the starting point with the initial orientation of the navigator, then some intermediate segments are specified and finally the end point is indicated. Usually landmarks and the reorientation of the navigator are designated. Phrases like “*and then*”, “*after that*” are used to segment route instructions into smaller sub-parts. Moreover, decision points are specified so that the navigator follows the correct track in cases of choice. Typically, those points are characterised in their relation to surrounding landmarks (cf. Habel et. al 2002).

Since route instructions are highly connected to the environmental context of the route described in the instruction, the occurrence of anaphoric references to entities in the visual and spatial environment is very high. Thus, they need to be interpreted with respect to the spatial context reference and domain-specific knowledge. References to the context are not only created by noun phrases (referring to entities) or verb phrases (referring to eventualities), they are also built up by anaphors for example personal or demonstrative pronouns.

Due to the fact that they also specify temporal information about the actions to be performed, the same linguistic input might refer to different entities in the environment depending on the actual situation. Thus, the noun phrase ‘*the road*’ used

in the following example refers to two different roads, since the second road is specified by a prepositional phrase. Moreover, due to inference processes we know that after ‘following one road’, you ‘take another one’.

- 1) *follow on the road*
and then take the road on your right

But the following example shows that different linguistic input might refer to the same entity.

- 2) *take **the second road** on your right*
*and follow **it** until you reach the museum*

Thus, in order to resolve anaphors occurring in route instructions, inference methods are needed. But let us first look what is meant by anaphoric reference in general.

2.2 Discourse and Anaphoric Reference

Before focussing on anaphora resolution, I should like to elucidate some theoretical concepts and definitions. Any kind of communication introduces information about objects, situations, events, facts etc. into the addressee’s mental model (*discourse model*). It is part of the context that participants use in understanding the meaning of a text. The discourse model contains representations of the entities that have been referred to in the ongoing discourse (*discourse entities* or *discourse referents*) and it also stores attributes of the discourse entities and information about the relationships in which the entities participate. The meaning of a term may depend on the previous discourse and it may also be affected by the spatio-temporal context which the discourse participants share and of which they are mutually aware. Reference to any entity that has been previously introduced into the discourse is called *anaphora*.

Anaphora resolution is defined as a two-phase process: firstly constructing a discourse model which stores any kind of information from the ongoing discourse; i.e. identifying what is available in a text for anaphoric reference; and secondly merging the anaphor with its referent, i.e. constraining the set of accessible candidates so that only one choice for the given anaphor is left. Most of the existing anaphora resolution algorithms only generate discourse entities for the referents of noun phrases. However, discourse entities represent any kind of participation of a particular

concept in the discourse, so they are not restricted to noun phrases. Following Woods in Webber 1991, they are considered as “*conceptual coathooks*” (Webber 1991, p.111) because they refer to any type of entity correspondent in the real or the hypothetical world. Demonstrative pronouns for example do often not refer to the referent of a previous noun phrase. They usually refer to different kinds of abstract objects such as events, states, concepts, propositions or facts like in the following examples from Byron (2002, p. 81), in which the referent of the discourse deictic demonstrative pronoun differs with respect to its context.

- (1) *Each Fall, penguins migrate to Fiji.*
- a) **That's** where they wait out the winter. (noun phrase: *Fiji*)
 - b) **That's** when it's too cold even for them. (noun phrase: *fall*)
 - c) **That's** why I'm going there next month.
(fact: *each fall, penguins migrate to Fiji*)
 - d) **It** happens just before the eggs hatch. (event: *penguin's migration to Fiji*)

While the demonstrative in 1.a) & b) refer to noun phrase in sentence (1): *Fiji* and *fall*, the demonstrative in 1.c) refers to a fact or proposition stated in the previous sentences (1). The pronoun in 1.d) refers to events mentioned in the previous sentence.

Referents built from constituents other than noun phrases are called *discourse deictic* referents. They are introduced into the discourse model by so called *deictic pronouns* (zero-pronoun, demonstrative pronoun or personal pronoun). If, as in the examples above, the deictic pronoun refers to a section of text previously mentioned, this is called *pure textual deixis*. Whereas, *impure textual deixis* is defined as anaphoric references to something that was expressed by a previous utterance of the discourse but which is not the utterance as a thing. The following example illustrates what is meant by impure textual deixis. Imagine for the following example, that Ann and Beth are searching for their missing dog in a forest and that they face a fork.

- (2) Beth: *I guess Duncan is chasing rabbits.*
Ann: *We knew that he is a hunting dog.*
Beth: *Maybe we should split and you take **this path** and I take **that one**.*

The referent of the anaphors *this path* and *that one* is not explicitly mentioned in the previous utterances but as the discourse participants are facing a fork, they can see the

entities referred to in the actual situation. They, therefore, have no problem finding the right antecedent for the anaphors. This example illustrates the meaning of the term *deixis* which is used by linguists in order to explain phenomena “*which relate utterances to the spatio-temporal coordinates of the act of utterance*” (Lyons 1977, p.636, quoted from Webber 1991, p. 109). The deictic anaphors refer to entities in the spatial environment in which the speakers are situated.

Following Webber (1991) and Eckert & Strube (2000) “*referents of discourse-deictic anaphors do not exist in the discourse model unless anaphorically referred to. For each context there are discourse entities that stand proxy for its propositional content*” (Eckert & Strube 2000, p. 59 and Webber 1991, p. 111). They are not introduced to the discourse model by virtue of the constituent that describes them but rather by virtue of anaphoric reference (cf. *referent coercion* in Eckert & Strube 2000). Thus, deictic pronouns can have the same effect as definite noun phrases. Both add new entities into the discourse model, such as in the following example from Webber (1991, p. 112).

- (3) *I walked up the first house on my list.*
*I noticed that **the side door** was wide open.*

As houses not necessarily have side doors, the listener inserts a new entity for ‘the side door of this house’ into the discourse model. This process is called *accommodation* because “*the use of a singular definite is felt to presuppose that there is already a unique entity in the context with the given description*” (Webber 1991, p.112).

Items in the discourse model which are mentioned in the previous discourse are called *salient*, whereas the most salient entities which attract the attention of the discourse participant at any point in the discourse are called *focused entities*. Various theories define salience and focus with respect to different criteria, but it is generally agreed that items which occur as noun phrases in the sentence are more salient than other items. Due to the occurrence of long-distance anaphora, discourse entities evoked by noun phrases remain in the discourse model for the duration of the

discourse while abstract entities are only available for pronominal reference in the immediately following discourse unit (cf. Byron 2002, p. 81).

2.3. Anaphora Resolution

As the terminology is clear now, I will next describe three commonly known anaphora resolution algorithms. Even though "*the notion of 'topic' or 'discourse focus' is notoriously difficult to formalize*" (Poesio & Modjeska 2002, p. 2), I will firstly give a brief outline of the Centering Theory by Grosz, Joshi & Weinstein (1995) which gives some guidelines for defining the 'topic' or 'focus' of an utterance. As this theory only deals with pronouns which refer to noun phrases, I will later also present two other algorithms which not only resolve pronouns referring to noun phrases but also deictic pronouns referring to abstract objects. Both algorithms (Eckert & Strube's 2000 and the PHORA-algorithm by Byron 2002) use the predication context of the pronoun to predict the antecedent-type of the anaphor (which is either an abstract (clausal) or individual (noun phrase) referent). The PHORA-algorithm is even more advanced. In addition to the common known salience calculations, it also applies semantic filtering in order to identify the correct antecedent of anaphor. After presenting the Centering Theory, I continue with a description of the Eckert & Strube-algorithm (2000). Finally, I will give details about the PHORA-algorithm by Byron (2002).

2.3.1 Focusing and Centering Theory

Some pronouns have more than one possible referent which matches the semantic properties of the anaphor. Thus, there is a potential ambiguity between these referents. But by and large human readers do not notice this ambiguity. This is the reason for the assumption that some entities in the discourse model are more salient than others. In the following example from Sidner (1983, p.371) the pronoun in (5b) can refer to either *the dog* or *the bull*.

- (4) a) *Sandy walked her dog near a bull one day.*
b) *He walked quietly along.*

But Sidner suggests that the chosen referent for a pronoun is always the focused entity in the discourse model except if the semantic properties of the referent and the predication context of the pronoun are incompatible. The focused entity in this example is the noun phrase referring to '*the dog*' since it is the object of the previous sentence, therefore '*the dog*' is the preferred referent of the pronoun rather than '*the bull*'.

But as mentioned above, there are different techniques to determine the 'focus'. The Centering Theory defines several relations between two adjacent utterances. It assumes that each utterance introduces new entities into the discourse. These entities are the so-called *Forward Looking Centers* (CF). They are ranked with respect to different criteria (e.g. grammatical role, information structure, coherence structure etc.). The most highly ranked entity in the CF of the previous utterance (U_{i-1}) is the focused entity in an utterance (U_i) (the so-called *Backward-Looking Center* (CB)). It needs to be said that the Centering Theory does not specify what 'utterances' are, how the entities in the CF are ranked, what kind of entities are introduced into the discourse and how they are created. Thus, researchers using the theory have to define those principles on their own.

Applying salience to resolve pronouns which refer to noun phrases works quite well but it helps to find the correct referent of deictic pronouns. Due to the fact that deictic pronouns tend to have clausal antecedent and according to the observation that they refer to semantically complex non-focused items, the Centering Theory does not succeed in resolving deictic pronouns. Nevertheless, Grosz & Sidner (1986) discuss the concept of a global focus which they characterize as the set of entities being part of the attentional state of the discourse participants. So even if there is only one entity in an utterance 'in focus', the other ones are 'activated' or in Poesio & Modjeska's (2002) words 'sufficiently salient'. They are in the short-term memory because they are mentioned in the discourse or because they are in the physical context (cf. example (3) above). According to Gundel et al's (1993) Givenness Hierarchy (quoted from Poesio & Modjeska 2002) personal pronouns usually refer to

entities ‘in focus’ while demonstrative pronouns more often refer to ‘activated’ entities.

Next, I will illustrate the Eckert & Strube-algorithm (2000) which takes into account differences in the activation status of entities and which not only successfully resolves personal pronouns but also determines the referent of deictic pronouns.

2.3.2 *Eckert & Strube – Algorithm*

The Eckert & Strube-algorithm replaces the *Forward and Backward Looking Centers* by an ordered list of salient entities, the *S-List*. Discourse entities realized as noun phrases are added to the S-List immediately after they are encountered. The ranking constraints on the S-List are based on Prince’s (1981, 1992; quoted from Eckert & Strube 2000) classification. Moreover, Eckert & Strube (2000) distinguish between *hearer-old discourse entities* (OLD), *mediated discourse entities* (MED) and *hearer-new discourse entities* (NEW). All proper names, anaphora, relative pronouns and appositives are categorized as hearer-old entities, while all kinds of *inferable* entities are classified as mediated entities. *Inferables* occur for example in textual ellipses where the definite noun phrase(A)’s referent is completely determined by a link to another noun phrase (B) in the previous utterance and where A and B are not co-referential. (e.g. “*the house ... the door*”). There are also the so-called containing *inferables* (e.g. “*the eggs ... one of them*”) or the anchored brandnew discourse entities which linked entities by possessive or genitive constructions (e.g. “*the king ... his daughter*”; “*the king ... the daughter of the king*”) (see Malvina Nissim: More on Co-reference, Tutorial slide of TNLP2, 2002). All other entities are labelled as new entities. Eckert & Strube presume that OLD or MEDIATED entities are the preferred antecedent of the pronoun. The simplified ranking order of the entities in the S-List is that entities which are classified as OLD precede MEDIATED ones, which precede NEW ones. The first entity in the S-List represents the most salient entity in the discourse model and the preferred antecedent of the anaphor. The E&S-algorithm resolves pronouns by co-indexing them with the highest-ranked entity in the S-List. The classification of entities into different levels of information status

does not reflect the context of the hearer (their knowledge, beliefs, goal, etc.). The aim of this procedure is to indicate what the *common ground* of the actual discourse is and what information the hearer and speaker share. Thus, Eckert & Strube (2000) point out that

“the discourse model is not intended to reflect which entities are familiar to the hearer but rather which entities are salient at that point in the discourse [...] It is similar to Grosz & Sidner’s attentional state, as it is intended to contain representations of entities which are salient to the participants” (Eckert & Strube 2000, p. 55-56).

As Navarretta (2000) indicates centering-based algorithms are usually tested on written text. However, there are some algorithms derived from the Centering Theory tested on multi-party dialogues. In order to apply the algorithm to the new class of data they extended the rules to account for dialogue-specific aspects (e.g. defining utterance boundaries). As Eckert & Strube developed and tested their algorithm on a non-task-oriented dialogue corpora, they assume that *dialogue acts* have an effect on which entities are added and removed from the representation of the attentional state (S-List). Since the data consists of spoken dialogues, the utterance boundaries are defined independently from punctuation. Therefore, Eckert & Strube decide that each main clause plus any subordinated clauses, or a smaller utterance, make up a unit (cf. Eckert & Strube 2000, p. 69). Moreover, they characterize dialogue acts as either *Initiations* (I), which contain semantic content or *Acknowledgements* (A), which ground the preceding initiation. A single initiation and an acknowledgement mutually form a *Synchronising Unit* (SU) (cf. A1 and B1 in the following example from Eckert & Strube 2000, p. 71), while single initiations in longer turns also build synchronising units (cf. A2 in example 6).

- (5) A1: *But we actually had some street people picked up last week in Dallas for picking up tin cans.* (I)
 B1: *My gracious.* (A) → SU1
 A2: *For picking up tin cans.* (I) → SU2
 They were going to turn them in, (I) → SU3
 they were going to cash them in. (I)
 B2: *Uh-huh.* (A) → SU4

Synchronising units are used to indicate at which point the S-List needs to be cleaned up. Cleaning up means that after each synchronising unit all discourse entities not referred to again are removed from the S-List.

Eckert & Strube (2000) emphasize that in their spoken-dialogue-corpus less than half of the occurrence of pronouns and demonstratives refer to noun phrases (*individual* anaphors 45.1%), even though this type of anaphoric reference is considered to be the ‘normal’ case. Nonetheless, 22.6% of the anaphora in their corpus have a sentential or verb phrase antecedent (*discourse deictic*) and the remaining third does not have an identifiable linguistic antecedent at all. This latter is classified as *vague* anaphor because these pronouns either refer to inferable entities (19.1%) or they refer to the general discourse topic (13.2%). In the following example from Eckert & Strube (2000, p. 65-66) the pronoun in the final sentence does not refer to any specific entity in the discourse model, but rather to the topic of the discourse: childcare in general.

- (6) A1: *I mean, the baby is like seventeen months and she just screams*
B1: *Uh-huh.*
A2: *Well even if she knows that they’re fixing to get ready to go over there.*
They’re not even there yet –
B2: *Uh-huh.*
A3: *you know*
B3: *Yeah. It’s hard.*

Thus, Eckert & Strube (2000, p. 52) conclude that some pronouns seem to have another function besides anaphoric reference, these pronouns seem to allow

“the speaker to leave certain referents underspecified. In spontaneous spoken language it is simply not necessary for the participants to be able to unambiguously identify a specific referent at all times”

As Eckert & Strube define rules for personal- and discourse deictic-pronoun resolution, they follow the idea of using the predicative context of the anaphors to distinguish between individual and abstract anaphors. As a result, anaphors that cannot refer to individual objects are marked as *I-Incompatible* (*I) and anaphors that cannot refer to abstract objects are labelled as *A-Incompatible* (*A). Eckert & Strube provide a list of certain criteria which the algorithm uses to correctly label the

anaphors. If for example in an equation construction the pronominal referent is equated with an abstract object such as “*this is a suggestion*”, the pronoun will be marked as I-Incompatible or if the pronominal referent in a copula construction whose adjective can only be applied to concrete, individual entities for example “*it is tasty*”, the pronoun will be labelled as A-Incompatible (cf. Eckert & Strube 2000, p. 77). Most of the predicates are not included in these lists because their predicative context allows both concrete and abstract referents in their argument position such as “*you know it*” where the pronoun could either refer to an individual referent (e.g. “*the horse*”) or to an abstract referent (e.g. “*Bush is the president of the USA*”). In those cases, in which the predication context does not constrain the pronoun, the algorithm makes use of the observation that demonstratives prefer abstract referents and personal pronouns prefer individuals. So demonstrative pronouns are resolved to abstract objects and personal pronouns to individual objects.

Eckert & Strube admit that the A- and I-Incompatibility lists are problematic, because the predicates in those lists are by and large preferentially associated with either abstract or individual referents and not categorically. Thus, even though a predicate is listed as A-Incompatible, there might be cases where an individual referent is also acceptable, and vice versa. But besides this problem, the division into *I- and *A-anaphors enhance the performance of the algorithm (Eckert & Strube 2000, p. 77-78).

It is now clear that the predicative context of the anaphor is important to reduce the set of possible referents by determining if its antecedent is either an abstract referent or a concrete, individual entity, but how does the algorithm identifies the correct antecedent? As described above individual anaphors refer to noun phrases. The possible noun phrase referents are stored in the S-List and ranked with respect to their info-status and their discourse position. For resolving discourse deictic anaphors Eckert & Strube (2000) introduce the notion of another list, the A-List, which contains the possible referents of a discourse deictic anaphor. Eckert & Strube (2000) take for granted Webber’s assumption (1991) that referents of discourse deictic anaphors do not exist in the discourse model unless anaphorically referred to which

leads to the insertion of a new entity into the discourse model (*referent coercion* or *ostension*). Thus, the A-List only includes abstract objects previously referred to anaphorically and no abstract objects referred to by each sentence and VP. Discourse deictic anaphors are resolved to the actual element in the A-List. If the A-List is empty, a new abstract object will be created by virtue of the following rules (cf. Eckert & Strube 2000, p. 75):

1. The antecedent is in same Initiation: Put the clause to the left of the clause containing the anaphor in the A-List.
2. The antecedent is in the previous Initiation: Put the rightmost main clause (and subordinated clauses to its right) in the A-List.
3. The antecedent is in the previous Initiation: Put the rightmost complete sentence (if previous I is incomplete sentence) in the A-List.

These rules do not only reflect referent coercion (explained above) but also the rule of linear or hierarchical adjacency between the anaphor and its antecedent. Using Webber's (1991) findings about the notion of adjacency, Eckert & Strube (2000, p. 63) define that the constituent referred to discourse deictically must be linearly or hierarchically adjacent to the anaphor. This discourse structural restriction works as a filter which reduces the set of possible candidates of discourse deictic referents. Like the S-List, the A-List is cleaned up at the end of each synchronising unit. As a consequence of this, referents which are not referred to again are removed.

In the following example from Eckert & Strube (2000, p. 75), the demonstrative pronoun (B1) referring to the preceding verb phrase, adds an event concept entity associated with this verb phrase into the discourse model (due to referent coercion). The subsequent personal pronouns will be resolved and co-indexed to the demonstrative pronoun which is also co-indexed with the abstract object in the A-List.

- (7) A1: ... *and we make it so easy for them [to stay there with welfare that they can get by just signing some papers.]_i*
 B1: *granted, they can do that_i very easily. It_i's easy to do, but look where it_i puts them.*

The overall performance of the algorithm on the spoken-dialogue corpus was quite well. Precision was 66.2 % and Recall 68.2% for individual anaphors, while for

discourse deictic anaphors Precision was 63.6% and Recall 70%. One of the most common classification errors was that discourse deictic or vague anaphors were classified as individuals because there was an individual antecedent available.

2.3.3 The PHORA-Algorithm

In this section, I will describe the PHORA-algorithm developed by Byron (2002) which uses salience calculations as criteria to resolve individual anaphors but which also takes advantage of the semantic constraints on the antecedents given by the predicative context of the anaphor. As discourse deictic anaphors usually refer to less salient abstract entities, semantic constraints can be used to find their referents. Byron makes use of the observation that pronouns which do not refer to the most salient item are typically constrained by their context and that they constrain the anaphor to be incompatible with the more salient referent while indicating the intended referent. Taking semantic constraints into account is useful for pronoun resolution algorithms in order to identify the referent of discourse deictic anaphors. This task is generally judged as being difficult and excluded from most of the pronoun resolution studies until now.

Following Webber (1991) who suggests that each discourse unit (context) has a ‘pseudo-DE’ that ‘stands proxy’ for its propositional context, the first action of the algorithm is to build up the discourse entities and the discourse proxies for the actual discourse unit (DU_n). To resolve a pronoun in the following discourse unit (DU_{n+1}), the algorithm firstly calculates the most general semantic type (T) that satisfies the constraints of the predicative context of the pronoun. Then, the algorithm checks the discourse entities in salience order to find a referent that matches the features of the pronoun. Depending on the type of anaphor (either personal or demonstrative pronoun) the algorithm uses different search orders. Finally, each discourse entity is tested with respect to the type constraint. Every entity that matches the type constraint of the anaphor (i.e. it either has the same semantic type as the anaphor or it is a subtype of it) is a possible referent. In general, there is only one possible referent left after testing the discourse entities.

The discourse model of the PHORA-algorithm contains two kinds of entities: *mentioned entities* and *activated entities*. Mentioned entities are referred to by noun phrases (e.g. proper names, descriptive noun phrases, demonstrative, personal, possessive and reflexive noun phrases). Entities referred to by NPs are inserted into the discourse model as soon as the corresponding NP is interpreted. By default the left-most noun phrase in each clause is the ‘focused’ mentioned entity of the actual discourse unit. All discourse entities have the following set of attributes: input (linguistic constituent), number (singular or plural), type (PERSON or ENGINE), composition (hetero- or homogenous), specificity (individual or kind), interpretation (referent or proxy associated with this discourse entity). Activated entities are proxies for non-noun-phrase-referents like entire sentences, nominals, clauses etc. They are evoked after each clause is interpreted. One clause can trigger multiple proxies. While mentioned entities remain in the discourse model for the complete discourse, activated entities only remain in the discourse model until the interpretation of the next clause is finished. Contrary to Eckert & Strube (2000), discourse deictic reference evokes a mentioned entity which is treated as any other mentioned entity and therefore, remains in the discourse model for the entire discourse.

Semantic type constraints for the pronouns are determined in the beginning of the pronouns’ resolution process. The criteria resemble Eckert & Strube’s list of A- and I-Incompatibility except that the set of types is more complex. Byron’s algorithm does not only distinguish between individuals and abstract objects, the type system is so detailed that equation constructions like “*that’s a good route*” constrain the anaphor as being an entity of the type ROUTE.

Since during pronoun resolution the first referent that matches the semantic type constraints and the agreement features for the pronoun is the preferred referent, the search order of the algorithm plays an important role. Thus, the algorithm employs different search orders for personal and demonstrative pronouns, which reflects the fact that demonstrative pronouns refer more often to activated (non-salient) entities whilst personal pronouns usually refer to mentioned entities (cf.

Byron 2002, p. 84-85). The following example from Byron (2002, p. 85) illustrates the proceeding of the PHORA-algorithm.

- (8) a) *Engine1 goes to Avon to get the oranges.*
b) *So it'll get there at 3 pm.*

The verb phrase in 9.a) is interpreted as action of the type ARRIVE which constrains its theme to be an MOVABLE-OBJECT. As “Avon” is a CITY and therefore, a NON-MOVABLE-OBJECT, the only possible candidates as referent of the personal pronoun “it” are “engine1” and “the oranges”. Due to the search order for personal pronouns, the pronoun is resolved to “engine1”. It is the left-most mentioned entity in the discourse model and it matches the semantic type constraint because it is a MOVABLE-OBJECT.

The general performance of the PHORA-algorithm is good. It correctly resolved 72% of the test pronouns in the dialogue-data which means that both the antecedent and the correct referring function was chosen. The construction of the discourse model is performed automatically by the PHORA software, whereas the semantic resources, like the entries in the semantic type hierarchy and the type constraints on the predicate argument positions, are built up in advance. Testing the PHORA-algorithm on the same data, but with only domain-independent semantics, resulted in 51% precision for correct resolution of the test pronouns. However, even though until now the PHORA-algorithm has only been implemented in a closed-domain system, the methods of the algorithm are not domain-specific. Using semantic resources like WordNet, the PHORA-algorithm could be used and tested on open-domain discourse data, too.

2.3.4 The THIS-NP Hypothesis

There is another paper which is related to the project. The authors, Poesio & Modjeska (2002), discuss a phenomenon occurring in the IBL-corpus. I will only briefly present the main points of this paper. It is a corpus-based investigation about ‘this NPs’ which frequently occur in the IBL-corpus as referring expressions like “*follow that road*”.

According to Poesio & Modjeska (2002) noun phrases with the determiner *this* and *that* (*demonstrative* NPs) are usually classified as deictic anaphors. As mentioned above, this type of anaphor tends to refer to objects in the visual situation (i.e. to objects to which the speaker points in the actual situation). However, these noun phrases also seem to have another function. They tend to refer to discourse entities which are salient but not ‘in focus’ (*pronominal* THIS NPs). Following this assumption, Poesio & Modjeska tested what they call their ‘THIS-NP Hypothesis’. They use two corpora in order to test their hypothesis: the GNOME-corpus which consists of descriptions of museum objects, and a selection of leaflets which provide patients with mandatory information about medicine. Their hypothesis is that “*THIS-NPs are used to refer to entities which are ACTIVE but not IN FOCUS*” (Poesio & Modjeska 2002, p.2). ‘In Focus’ is defined in terms of Centering Theory while the classification of ‘active’ entities is derived from Grosz & Sidner’s concept of global focus which includes the entire set of entities in the discourse model. Thus, ‘active’ entities are non-focused but salient entities. Contrary to Byron (2002), activated entities are not meant to be proxies. Poesio & Modjeska assume that the global focus is composed of a ‘discoursal’ and a ‘visual’ part. While the ‘discoursal’ global focus contains every discourse entity evoked by a construction, the ‘visual’ global focus is a situation-based structure which includes every entity in the visual scene. Moreover, Poesio & Modjeska presume that both inference- and reference-processes can add new entities into the global focus (discourse model). These assumptions lead to a new definition of ‘activeness’. An entity is ‘active’ if it is in the visual situation or if it is salient in the previous utterance. It is also ‘active’ if it can be constructed out of the previous utterance, which means that it is either a plural object whose elements or subsets have been explicitly mentioned in that utterance or that it is an abstract entity (propositions or types) introduced by that utterance. The following example shows what is meant by constructing an entity out of the previous utterance, as “*this yearning*” is joint idiom for the descriptions in the previous utterance (cf. Poesio & Modjeska 2002, p. 4).

(9) *The craftsmen also bent carefully over cheaper metals or glass to create the jewelry that would adorn the arm of the humble servant girl, or the ordinary, insignificant woman, and would accompany her to her final resting place. **This yearning for embellishment this special relationship between a woman and her jewelry** emerges quite clearly here ...*

For most of the test THIS-NP-anaphors Poesio & Modjeska's hypothesis was verified. Nonetheless, there were some cases where the hypothesis was falsified because the antecedent of the anaphor was a focused entity.

3. RIAR- and SR-Algorithms

In this section I will describe the RIAR algorithm developed by analysing pronominal reference and reference to spatial location in a set of development data from the IBL-corpus. To start with, a general scheme of the algorithm is introduced. Following the two phases of anaphora resolution, firstly I will explain how the algorithm identifies what is available in the route instructions for anaphoric reference. This will require an account of how the discourse model is constructed and updated. Secondly, the mechanisms for constraining the set of possible candidates will be illustrated. The algorithm checks several features of the anaphora to define its salience. Moreover, it employs semantic filtering and a certain kind of pre-classification in order to recognize which resolution rule needs to be applied. The use of each rule depends on the type of anaphora and its pre-classification which is determined by selectional restriction rules. Different rules are necessary to resolve the anaphor to its antecedent. Examples from the data will be given, to demonstrate how each resolution rule works in practice.

3.1 Defining the RIAR-Algorithm

The general approach of the algorithm is to resolve unresolved anaphors in a set of units (given as input) so that its output is a set of units with resolved anaphors. For this, a discourse model which corresponds to the given units is constructed and updated after each unit. Then, the properties (constraints) of each unresolved anaphor(A) in unit(U) are generated and the corresponding resolution rule is applied in order to define the correct antecedent of the anaphor. A more formal description of the RIAR-algorithm is expressed in the following resolution scheme (see Appendix

4): *Input:* set of units with unresolved anaphors, discourse model (DM)
 Output: set of units with resolved anaphors

For each unit(U):

- update DM
- for each unresolved anaphor(A) in unit(U):
 - generate properties for anaphor(A)
 - apply rules to anaphor(A)

3.2 Discourse Model and Units

The following sections explain what is meant by the concepts mentioned in the formal description of the algorithm and how the algorithm proceeds in practice is illustrated.

The discourse model is a representation of the mental model which the discourse participants build up during the communication and which contains all the information explicitly or implicitly introduced by the discourse. Some of this information is represented in discourse entities which either stand for individual, concrete objects or for abstract objects. Depending on the structure of the discourse and depending on the type of anaphor, the participation of these entities in anaphoric reference varies. Some entities are more salient than others. Thus, individual entities referred to by noun phrases tend to be ‘in focus’ while abstract entities like eventualities are salient but not ‘in focus’. They are only available for pronominal reference in the subsequent discourse unit while individual entities remain in the discourse model for the duration of the discourse to support long-distance anaphora such as in problem solving conversations. For example questions remain salient in the discourse model until the answer is given. Moreover, particular aspects of the solution are salient even though they are not explicitly mentioned (see Byron 2002, p. 81). This account is relevant for the domain in which the RIAR-algorithm works because in the domain of route instructions a participant’s answer are given to an initial question from the robot. The initial question and the answer in the form of a route instruction constitute the robot’s task which is salient until the questioned destination or goal is reached. Thus, not only the information given in the route instruction is stored in the discourse model but also the origin, the destination, the task and the initial road (the road where the robot stands at the beginning of the task).

A question like *”How do I get from the museum to the post office?”* is represented in the discourse model as:

Discourse Model at unit(1):

origin=museum(sg., BUILDING)

destination=post_office(sg., BUILDING)

task='go from *origin*=museum to *destination*=post office'(sg., EVENTUALITY)

initial_road=road0(sg., PATH)

Unfortunately, the initial questions of the robot are not recorded in the IBL-corpus but the route instructions are systematically catalogued so that the origin and the destination are accessible from the file name under which the route instruction is saved. As anaphoric reference to origin- and initial-road-entities is more likely in the first sentences of the route instructions (cf. examples 1. and 2. below), destination-entities are usually referred to in the final sentence of the instruction while task-entities can be referred to either at the very beginning of the instruction or at the end (cf. examples 3. and 4. below).

1. Origin-reference in First Sentence: *from this building you go along the road*
→ 'this building' refers to *origin*=museum(sg., BUILDING)
2. Initial-road-reference in First Sentence: *follow this road*
→ 'this road' refers to *initial_road*=road0(sg., PATH)
3. a) Task-reference in First Sentence: *to do that is not very difficult*
→ 'that' refers to *task*='go from museum to post_office'(sg., EVENTUALITY)
b) Task-reference in Final Sentence: *I hope doing that was not too difficult for you*
→ 'that' refers to *task*='go from museum to post_office'(sg., EVENTUALITY)
4. Destination-reference in Final Sentence: *it is on your left there*
→ 'it' refers to *destination*=post_office(sg., BUILDING)

Contrary to Byron's algorithm, the discourse model of the RIAR-algorithm does not contain proxies for abstract referents referring to eventualities. It only takes into account all individual or abstract entities referred to by NPs. These discourse entities are either explicitly mentioned in the discourse (i.e. they are referred to by NPs) or they are implicitly introduced by 'This-NP'-anaphoric reference (cf. detail discussion of 'This-NP'-Anaphor in section 3.4). All discourse entities have a unique identifier which is necessary for the implementation of the algorithm. Moreover, they are specified with respect to grammatical and semantic features like number, type, prepositional specification, adjective specification and discourse position. The general structure of discourse entities is represented as: Identifier(NR, TYPE, PP-Spec, ADJ-Spec, Discourse Position). In the discourse model the actual value of these features

is fixed. The identifier is defined as the head noun of the noun phrase plus a digit which records the occurrence of the noun in the ongoing discourse; the number feature takes either plural or singular as value; the type feature contains the hyper type of the noun. As the hyper type is part of the lexical entry of each noun, and it can be retrieved from the lexicon. The features for PP- and ADJ-specifications take into account any relevant information expressed by adjectives or prepositional phrases. The discourse position contains relevant information about the discourse structure. All entities are stored in the discourse model with these specifying features so that, for example, the discourse entity referred to by the following noun phrase: “*the small building on your left*” mentioned at the beginning of a route instruction is represented in the discourse model as:

building1(sg., STRUCTURE_CONSTRUCTION, [in(left_region)], [small], 1.1).

In order to update the discourse model the route instruction is segmented into smaller units. As the IBL-corpus consists of spoken route instructions, utterances are not always made of complete sentences. Moreover, most of the utterances are short imperatives or brief descriptions of the visual scenery in the actual situation. Using sentence construction in order to segment the instruction is therefore inadequate. The algorithm works more efficiently if the updating units are defined with respect to endpoints of actions or with respect to marking points of descriptions mentioned in the route instructions. Thus, updating units are defined as follows: each tensed finite verb with at least one argument or adjunct builds an updating unit, in which the verb either expresses an action or a description and in which the argument or adjunct refers to an end- or marking point. Since in the following example (1) the if-clause is complete, which means it contains a verb and an adjunct, the whole conditional sentence consists of two updating units. Whereas, the temporal sentence in example (2) creates only one updating unit because the first verb has no argument or adjunct (number in square brackets refers to amount of updating units).

- 1) *if you go to the roundabout* [1] *take the second on your left* [2]
- 2) *carry along until you find the car park* [1] *and turn right into it* [2]

Prepositional phrases like *'to the left'* do not count as proper argument or adjunct. Thus, *'go to the left'* does not create an updating unit on its own, because the prepositional phrase does not refer to an endpoint. In contrast, the verb phrases *'turn left'* and *'turn right'* do not need an argument or adjunct to form a unit, since they describe actions which introduce a change of orientation into the visual context. They, therefore refer to an implicit endpoint of the action (the point where the change of orientation happens). Due to this implicit endpoint they constitute complete actions on their own. Nonetheless, relative clauses do not create new updating units because they are specifiers of the corresponding NP. They usually give more detailed descriptions of entities already introduced into the discourse model, but they do not introduce new units.

'And' used as a conjunction between clauses and sentences usually connects two updating units while it is also used as conjunction for coordination of constituents like nouns, noun phrases, verb phrases etc. *'And'* in coordination construction tends to be elliptic. If for example the verb is involved in the coordination a new updating unit is created like in the following example:

- 3) *you have to take right*[1] **and** *then again the first right* [2]

Constructing updating units for elliptic constructions needs to be done by human annotators because elliptic constructions are too difficult to handle since there is no method available for dealing with all types of elliptic constructions. The following example illustrates how tricky some elliptic constructions are:

- 4) *continue over the crossroads* [1]
across the bridge over the lake [2] → **ellipsis**
across straight across the next crossroads between the post office and pizza hut [3]
→ **ellipsis**

The units are not only needed to update the discourse model, moreover they play an important role in constraining the content of the so-called S-List and A-List. These lists contain entities which are available for anaphoric reference at a certain position of the discourse and which therefore, need to be updated as well. But these lists will be described in more detail later in section 3.4.

Furthermore, for the sake of the RIAR-algorithm it is assumed that if the same entity is mentioned several times (either by the same NP or by synonyms), the algorithm is able to recognize that these NPs co-refer. Since ‘*the road*’ in the following example is mentioned in two different units, two discourse entities are introduced which refer to the same entity in the ‘real’ world (they have the same identifier).

take the second road on your left [1] and follow the road [2]

discourse model: **road1**(sg., [PATH], [in(left_region)], [second], **1.1**)
 road1(sg., [PATH], [], [], **2.1**)

Nevertheless, the features for their discourse position and their features for Adj- and PP-specification differ, because both NPs are constituents in different units. Thus, ‘*the road*’ mentioned first is in the left_region of the robot before the robot has performed the action described in unit(1) while the same road is not any longer in the left_region of the robot after performing the action described in unit(1). ‘The same road’ is now in the front_region of the robot. The new discourse position is needed for updating the Saliency-List (explained later in section 3.4).

3.3 Ontology, Lexicon and Selectional Restrictions

The ontology and lexicon used by the algorithm needs to define the type feature of the mentioned entities in the discourse model. But it is even more important for the second phase of the anaphor resolution process in which the possible set of antecedents is filtered with respect to the constraints of the anaphor. The ontology, the lexicon and the selectional restriction rules are necessary to generate the properties of the anaphor (cf. section 1). The properties of the anaphor are the following features: identifier, number, type, PP- and/or ADJ-specification and discourse position (anaphor: X(sg., Type?, PP-Spec?, ADJ-Spec?, Discourse-Position). Since the antecedent is unknown when anaphor-resolution process starts, the identifier is a variable(X) which will be unified with the antecedent after the resolution process is finished. The number feature has always the value singular (sg.) because the RIAR-algorithm only deals with singular anaphors. The discourse-

position is the number of the updating unit and the position within this unit. Thus, if the anaphor occurs after the third noun phrase in unit(2) its discourse-position is defined as '2.4'. The type feature and the PP- and ADJ-specification are denoted by the selectional restriction rules which will now be described in detail.

The selectional restriction rules reflect the predicative context of the anaphor which means they help the algorithm to define whether an anaphor refers to an abstract object (e.g. state, event, concept, proposition, fact) or to an individual, concrete object (cf. A- and I-Incompatibility in Eckert & Strube 2000). Moreover, they serve as semantic filtering mechanisms because they provide type constraints for the set of possible antecedents. Thus, the selectional restriction rules of the RIAR-algorithm are defined with respect to Eckert & Strube's (2000) notion of A- and I-Incompatibility. In addition to this, they are differentiated as semantic filtering rules comparable to the semantic constraints of Byron's PHORA-algorithm (2002). In order to represent the concept of A- and I-Incompatibility and the idea of semantic filtering an ordered ontology is used. The rough organization of the ontology is based on WordNet. Nouns and verbs are connected by 'ISA'-relations (vertical relations between the nodes). Adjectives and adverbs are used to indicate disjointedness between concepts (horizontal relations between nodes). An additional relation defined especially for this domain is the 'PART_OF'-relation which illustrates that certain concepts consist of other concepts (dotted line). The following figure shows that the concepts *path* and *junction* hold an 'ISA'-relation to the concept *way*. It also shows that the concepts *road*, *street*, *branch*, *fork* are in a 'PART_OF'-relation to the concepts *crossing*, *crossroads*, *t-junction*, *junction*, *turning*. Entities in the bottom node of the 'PART_OF'-relation are element_entities while entities in the top node are complex_entities.

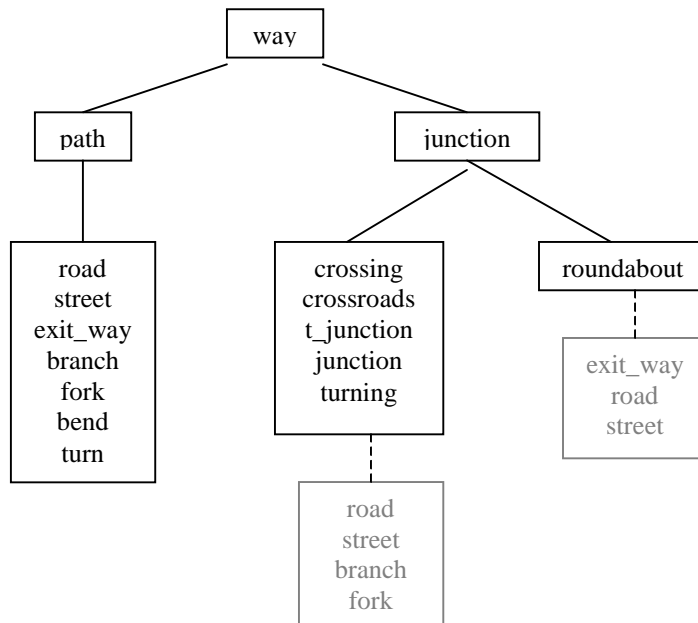


Figure 1: 'ISA'- and 'PART_OF'-relations between concepts in the ontology

As in the ontology the concept *thing* is subdivided into *abstraction* and *entity* the notion of A- and I-Incompatibility is relatively easily to represent. All subtypes of the concept *abstraction* are I-incompatible, while all subtypes of the concept *entity* are A-incompatible. The information about the relational order between the concepts is explicitly stored in the ontology but it is also retrievable from the lexicon since the lexical entry of each word contains the conceptual hyper type of the word. Additionally, the lexical entries store closely related concepts (concepts in the same box) and they keep the information about the class of adjectives which are likely to modify the word. As the domain is quite restricted and since there are not many adjectives, the classes of adjectives are small (cf. adjective classification in the lexicon, Appendix 3) and the number of constraints derived from adjective specification is diminutive. But due to the organization of ontology (horizontal relations) the algorithm is able to indicate inconsistent adjective specification.

- 1) *go to the big building on your right*
- 2) *then it is the next small building on your left*

If, as in the example above ‘*the building*’ mentioned in the route instruction is specified as being big, the algorithm will not resolve the subsequent anaphor to ‘*the big building*’ because due to the semantic filtering the antecedent of the anaphor is constrained to as being small (and in addition to this ‘*it*’ in the final sentence usually refers to the destination). Moreover, because of the PP-specification the antecedent of the anaphor needs to be ‘on the left hand side of the robot’. To filter this information, reasoning about the specification given in the PP is necessary. Even though this inference process is not implemented yet, the algorithm works as if such a reasoning technique exists. This shows that the predicative context of the anaphor provides information necessary to correctly resolve it. For retrieving this information and defining constraints for the antecedent, the RIAR-algorithm uses six selectional restriction rules and two rules dealing with spatial inference processes (see Appendix 2).

The first rule deals with **Equation Constructions** where the pronominal referent is equated either with an abstract object or with a concrete, individual referent. So is it more likely that the anaphor in the x-position in: *x is a suggestion* refers to an abstract object while the anaphor in: *x is a long street* refers to a concrete, individual entity. In equation constructions the anaphor and the complement-NP of the equation construction are entities of the same concept type. Since the anaphor in the previous example occurs in an equation construction the properties of this anaphor are X(sg., [BUILDING], [in(left_region)], [next, small], 2.1).

There are two special rules which define how the algorithm handles **Spatial Inference** given by spatial PPs. The first one controls how spatial relation given in PPs is interpreted. In general, expressions like ‘*on your left/right hand side*’ or abbreviations of this expression like ‘*on your left/right*’ or ‘*on the left/right*’ are interpreted with respect to the actual position of the robot. If such a spatial PP is used to specify an entity, the feature of PP-specification is: [in(left_region)] which means that the entity is located in the left_region of the robot. If the spatial PP is used in a genitive construction like: “*the left hand side of the street*” where the PP specifies the subsequent noun, the feature of PP-specification is filled as follows:

[in(left_region_of(street1))]. The second rule handles cases in which the spatial PP specifies the verb but the specification also allows inference about the properties of the corresponding argument. This happens when the corresponding argument is realised as anaphor like in the following examples:

- 3) *you will see **it** on your left*
- 4) ***it** is on your left there*

As pronouns cannot be specified by PPs, the spatial PP: ‘*on your left*’ modifies the verb. However, it can be inferred through the reasoning processes that the entities referred to by the anaphors are both located in the left_region of the robot. These spatial inference rules are especially defined for the domain of route instructions.

The second rule takes **Adjective Copula Constructions** into account where the adjective modifies either an abstract entity (e.g. *x is true*) or a concrete entity (e.g. *x is expensive*). Depending on the kind of adjective, it also constrains the subject to be a certain kind of entity, for example in the following example:

- 5) *follow the road to the museum*
- 6) *be careful, **it** is quite bendy*

The lexical entry for ‘bendy’ stores the information that ‘bendy’ is likely to specify an entity of the concept: ‘path’ (cf. lexical entry for ‘bendy’ in Appendix 3). By virtue of the constraint (retrievable from the lexicon) the algorithm would not try to resolve the anaphor to ‘*the museum*’.

The third rule checks the lexicon for type constraints for the **Arguments of Verbs** (subject or object). The lexical entry of each verb contains type constraints for the subject and if the verb is transitive it also stores type constraints for the object. The lexical entries for ‘reach’ and ‘take’ show that the subject is restricted to either a ‘human_person’ or a ‘path’. As ‘reach’ belongs to the subclass of REACH-verbs its object is specified as final position of the action (see lexical entry of ‘reach’ in Appendix 3). ‘Take’ is a TURN-verb. Therefore, its object is restricted to be a ‘path’-entity (cf. lexical entry of ‘take’ in Appendix 3). ‘Assume’ is a PROPOSITIONAL ATTITUDE-verb, which is why its complement is defined as a propositional clause (cf. lexical entry of ‘assume’ in Appendix 3). The object of ‘do’ always refers to an

abstract object (eventuality). Thus, an anaphor in the object position of do like in “*do it*” or “*do this*” is classified as I-incompatible and the anaphor will be resolved to the element in the A-List. The classification of vocabulary and the constraints for the argument highly depend on the knowledge about the domain of route instructions. The IBL-lexicon manipulated with respect to this domain knowledge cannot serve as a sensible lexicon in other domains.

The fourth rule is defined especially for certain constructions found in the domain of route instructions. These constructions describe **Movements** in which the **Agent** occurs in the **Object Position** like in the following examples:

- 7) *X will take you to the museum*
- 8) *X will bring you to the museum*
- 9) *X will get you to the museum*
- 10) *X will get you back to the museum*

If the algorithm hits any of these constructions, there are two possibilities for resolving the anaphor. If the verb of the previous unit is not a Verb of Motion (see verb classification in the lexicon, Appendix 3) and if there is also a PATH-entity mentioned, the anaphor will be classified as A-incompatible and it will be resolved to the PATH-entity previously mentioned. Yet, if the verb of the previous unit is a Verb of Motion or if there is no PATH-entity previously mentioned, the anaphor will be classified as I-incompatible and it will be resolved to the element in the A-List (the notion of the A-List will be described in the next section). The constructions which this rule handles are ambiguous. Even for human annotators it seems to be difficult to determine the antecedent of the anaphor in the constructions discussed here. The anaphor in the following example has four possible interpretations.

- 11) *go straight over the next crossroads and follow the road*
- 12) ***that** will take you to the museum*

Firstly, the anaphor can refer to ‘*the road*’, secondly it might refer to the single event: ‘*follow the road*’, thirdly it may refer to the sequence of events: ‘*go straight over the next crossroads and follow the road*’ and finally the anaphor might refer to the whole instruction. The algorithm would resolve the anaphor in this example to the single event: ‘*follow the road*’, but it needs to be said that the rule described above only

reflects a preference for some of the possible interpretations as the antecedent of the anaphor in these constructions cannot be defined exactly.

The fifth rule defines how the algorithm works if the anaphor occurs as **Argument of a Conjunction** like in the following example.

*13) don't turn right there, instead of **that** go straight past the post office*

As conjunctions connect clauses (which refer to event, fact, proposition etc), anaphors occurring in the argument position of a conjunction are classified as I-incompatible and they are resolved to the element in the A-List.

The sixth rule copes with constructions in which the anaphor occurs as **Object of a Spatial PP**. If this is the case, the anaphor generally refers to a concrete, individual object even though the lexical entry of most of the prepositions marks the preposition as ambiguous because they can be combined either with individual or with abstract entities. In the following examples the anaphor is ambiguous because it is the object of an 'into'-, 'in'-, 'before'- or 'after'-PP. The object of these prepositions is either an abstract object as in “*go into/in the middle*” or it is a concrete, individual entity as in “*go into/in the car park*”. ‘After’ and ‘before’ are also ambiguous but in a different way because they are used as prepositions or as conjunctions. If they function as prepositions their object refers to concrete, individual objects while arguments of conjunctions refer to abstract objects. Again, this rule is defined with respect to the domain so that it cannot be generalised and transferred into other domains.

3.4 Salience-List, Implicit S-List and Abstract-Object-List

The Salience-List (S-List) includes all discourse entities which match the constraints of the anaphor actually processed and which are highly salient at this processing point in the discourse. As previously explained, applying selectional restrictions rules and employing inference mechanisms not only defines the properties of the entities stored in the discourse model but also determines the properties of the anaphor. The properties of the anaphor are used as constraints for filtering the entities from the discourse model. Only discourse entities mentioned in the same or in the previous

unit as the anaphor and introduced by NPs are taken into consideration for the S-List, so that the algorithm is able to handle inter- and intrasentential anaphora. This models the attentional state of the discourse participants because entities mentioned earlier in the discourse are not activated highly enough to be an antecedent of the current anaphor. The properties of an anaphor match the properties of a discourse model if the values of all features are consistent. The number feature is inconsistent if the discourse entity is a plural NP (sg. \neq pl.), the type features are inconsistent if the discourse entity and the anaphor are not the same type or if the discourse entity is not a hyper- or sub-type of the anaphor. As mentioned above, the spatial inference rules are needed to ensure that entities located in the `left_region` of the robot and entities located in the `right_region` of the robot cannot co-refer. To check the `Adj-specification` feature the ordering of the ontology is taken into consideration (horizontal relations between nodes). Furthermore, the lexical entries of adjectives store not only related adjectives but also one member of the antonymous class. The `Adj-features` are inconsistent, if the adjective specifying the discourse entity and the anaphor are antonyms.

Eckert & Strube (2000) define the ranking criteria of the S-List according to the info-status of the entities mentioned previously. As described in section 2.3.2 the info-status denotes entities as either discourse ‘new’, ‘mediated’ or ‘old’. The ‘focused’ entity (the preferred antecedent) in Eckert & Strube’s algorithm is defined according to the info-status and the discourse position of the entities. Discourse ‘old’ entities are more highly ranked, than ‘mediated’ or ‘new’.

In contrast to Eckert & Strube, the entities in the S-List in the RIAR-algorithm are primarily ranked according to their type and discourse position. As the route instructions are not conceptually complex, which means that entities are not realised by several semantically related NPs, the RIAR-algorithm works without the notion of the info-status. Instead, the entity with the closest related type to the anaphor is the most highly ranked entity in the S-List (first position). This also ensures that entities not in focus are inserted into the S-List, so that also ‘This-NP’-anaphors can be resolved by means of constructing a S-List.

However, as in Eckert & Strube's algorithm, the discourse position also controls the status of the entities in the S-List. Entities mentioned in the same unit as the anaphor are more highly ranked than entities mentioned in the previous unit. Within one unit the first mentioned entity is more salient than entities mentioned later in the unit. The entity which is in the first position of the S-List is the preferred antecedent of the anaphor. So if the corresponding rule is applied, the anaphor will be resolved to the first entity in the S-List.

Unlike in Eckert & Strube's account, the S-List is not updated and cleaned (which means that all discourse entities from the previous unit which are not realised again in actual unit are removed from the S-List). Eckert & Strube have no discourse model or, in other words, they only model the attentional state of the hearer as the S-List which is a partial discourse model including only highly salient entities. The RIAR-algorithm only constructs a S-List when it hits an unresolved anaphor. The content of the S-List is always deleted after the anaphor is resolved and a completely new S-List is built up for the next unresolved anaphor.

The concept of an **implicit S-List** derives from the observation that some entities are plural objects or complex entities which consists of several elements of subsets (e.g. crossroads which consists of two crossing roads). The 'normal' S-List only contains entities which are conceptually related to the anaphor (hyper- or subtype of the anaphor or the same type as the anaphor). The implicit S-List is built up if a 'This-NP'-anaphor occurs and if the concept of its NP is either an `element_entity` or a `complex_entity`. If the NP-concept is a `element_entity` (e.g. '*this road*'), the implicit S-List contains all entities which have the same concept as the corresponding `complex_entity` of the `element_entity` (e.g. 'crossing', 'junction', 'crossroads' etc.). Conversely, if the NP-concept is a `complex_entity` (e.g. '*this junction*') all entities of the same type as corresponding `element_entities` are put into the S-List (e.g. 'road', 'street', 'branch' etc.). Like the S-List, the implicit S-List only contains entities mentioned in the previous unit or in the same unit as the anaphor.

The RIAR-algorithm resolves discourse deictic anaphors almost in the same way as the Eckert & Strube-algorithm. As soon as the anaphor is marked as I-

incompatible, which means that it refers to an abstract object (event, state, proposition, etc.), a list which contains abstract objects is constructed. The **A-List** only contains a sub-group of the entities classified as *abstractions* in the ontology. *Abstractions* which are realised as noun phrases are elements of the S-List. The A-List only includes *eventualities* like events, states, processes, facts etc. This is contrary to Byron's algorithm which stores so-called 'proxies' for each sentence and each verb phrase and the RIAR-algorithm which inserts only entities referred to anaphorically into the A-List. Therefore, the A-List is usually empty and a new A-List is constructed according to the following rule: if the A-List is empty, create a new A-List by inserting the main verb phrase of the previous unit plus all constituents following this verb phrase. Then, resolve the anaphor to the element in the A-List. The element is removed from the A-List if it is not referred to anaphorically again in the next unit.

Contrary to Eckert & Strube's algorithm which also puts whole sentences and clauses into the A-List, the RIAR-algorithm only inserts the main verb phrases plus further following constituents into the A-List. The utterances in the route instruction domain are shorter and their focus lays on the actions described in most of the units. Thus, discourse deictic anaphors preferentially refer to the main verb phrases. As only a few of the anaphors in the actual data are discourse deictic this assumption needs further investigation.

3.5 Resolution Rules of the RIAR-Algorithm

Now after the description of the first steps of the algorithm (constructing a discourse model with specifies discourse entities; defining updating units; generating the properties of the anaphor and finally creating a S-List, an implicit S-List or an A-List depending on the resolution rule in charge), the resolution rules of the RIAR-algorithm will be explained (see Appendix 4). Depending on the phenomena observed in the data, there are five different types of resolution rules. The first rule deals with anaphors occurring in complex NPs, the second and third rules are special rules which handle anaphoric reference in the first respectively final sentences. The

fourth type of rule is in a large degree similar to Eckert & Strube's resolution rules while the fifth type of rule takes 'This-NP'-anaphors into consideration.

The **Complex-Constituent Rule** handles constructions in which the anaphor occurs in a complex constituent (e.g. nouns which are specified by several PPs or relative clauses) and in which it refers to a previously mentioned element of this constituent like in the following examples:

- 1) [*the lake*]₁ with the bridge on *it*₁
- 2) [*the lake*]₂ which has a bridge on *it*₂

The rule is defined according to syntactic constraints known from Chomsky's *Government and Binding Theory*. Applying the rule requires the anaphor to be a pronoun and either part of a PP which specifies a previously mentioned noun in the same complex constituent or part of a PP in a relative clause (specifying a noun). If the anaphor matches these preconditions, it is resolved to the nearest noun in the syntactic tree which c-commands the PP including the anaphor (cf. *Complex-Constituent Rule*, Appendix 4). The notion of c-command describes the hierarchical relation between constituents in a syntactic tree. Following Haegeman (1991) c-command is defined as: "A c-commands B iff A does not dominate B and every X that dominates A also dominates B" (Haegeman 1991, p.135). The following example is ambiguous because the pronoun could either refer to 'the tree' or to 'the lake'. The rule defined above only finds the second reading (where 'the lake' is the antecedent of the anaphor) not the first one.

- 3) [*the tree*]₁ by [*the lake*]₂ with the flower on *it*_{1/2}

This shows that the rule might fail in correctly resolving all the anaphors because it only expresses a preference of resolving the anaphor in this context. However, PP-attachment, respectively PP-interpretation, is not only difficult to handle for the RIAR-algorithm, this field of work is generally known to be tricky. Moreover, complex constituents only rarely occur in the data.

The **First Sentence Rules** are in control of resolving anaphors which occur in the first sentence of a route instruction. If the anaphor does not refer to an element

mentioned in the first sentence, it is likely that it refers to the origin (cf. example 4), to the initial road (cf. example 5) or to the task (cf. example 6).

- 4) *from **this building** you go along the road* → origin-reference
- 5) *okay, follow **this road*** → initial road-reference
- 6) *well, to do **that** is not very difficult* → task-reference

If the anaphor refers to the origin-entity, its type is constrained to be a ‘building’ (or any hyper type of ‘building’). To ensure that the anaphor does not refer to any other element in the first sentence, the S-List is constructed. Only if the S-List is empty, will the anaphor be resolved to the origin-entity stored in the discourse model. The rule which resolves the anaphor to the initial-road is similar to the origin-rule, except for the type constraint for the anaphor. The anaphor needs to be a ‘path’ (or any hyper type of ‘path’) to be resolved to the initial-road stored in the DM. As the task-entity stored in the discourse model refers to an eventuality, the rule for the task-reference in the first sentence differs from the previous ones. The anaphor is likely to refer to the task (eventuality), if it is marked as being I-incompatible. Usually, the algorithm tries to create an A-List if the anaphor is marked as being I-incompatible but as the A-List of the first sentence is always empty, the only way of dealing with the anaphor is to resolve it to the task stored in the discourse model. If none of the First Sentence Rules succeed, the algorithm applies either the Pronominal Anaphora or the This-NP-Anaphora Rules (explained below) in order to resolve the anaphor.

Similarly to the third First sentence rule, the first **Final Sentence Rule** dealing with task-reference in the final sentence constrains the anaphor to be I-incompatible. The rule defined like this is somewhat problematic because the anaphor could also refer to another event previously mentioned in the route instruction. Thus, it is unclear whether the anaphor in the following example refers to the task-entity (‘go from the origin=X to the destination=Y’) or the event mentioned in the previous utterance (‘go to the museum’).

- 7) *then go to the museum [1]*
*I hope **that** was not too difficult for you [2]* → task-reference

Even though the first Final Sentence Rule defined like this will not correctly resolve all anaphors occurring in final sentences, it expresses the preference that anaphors marked as I-incompatible more often refer to the task (which refers to the sequence of events taking the robot to the destination) than to a single event of the route instruction. This rule describes another domain specific phenomenon.

The second Final sentence rule deals with anaphoric reference to the destination without explicitly mentioning the destination in the utterances preceding the anaphor. This phenomenon occurs quite often in data. For resolving the anaphor mentioned in the final sentence to the destination stored in the discourse model, the algorithm firstly has to check whether the anaphor is a pronoun, as this was the case in all the examples found in the data; replacing the pronoun by a demonstrative sounds odd (cf. examples 8 and 9).

8) *and **it** is on your left there*

9) *and **that** is on your left there*

Secondly, the properties of the anaphor have to match the properties of the destination (usually only the type constraint: ‘building’). Finally, the algorithm has to find out if the destination is mentioned in one of the previous utterances (e.g. if unit(4) is the final unit, units(3 and 2) have to be checked). If all these conditions are fulfilled, the anaphor is resolved to the destination stored in the discourse model. If the anaphor does not match the properties of the destination, the Pronominal Anaphora Rules (see below) are applied. If the destination is mentioned in one of the previous units, the anaphor is resolved to the most recently mentioned destination. Before the next rules are explained, it needs to be said that it is assumed for the First and Final sentence rules that the algorithm is able to recognise the first and final sentence of the route instructions.

The remaining anaphors, either pronominal anaphors (‘it’, ‘this’ and ‘that’), or ‘This-NP’-anaphors (e.g. ‘that road’) are resolved by the **Pronominal Anaphora Rules** and **This-NP-Anaphora Rules** respectively. The Pronominal Anaphora Rules use the concept of the S-List or of the A-List in order to determine the antecedent of the anaphor while the ‘This-NP’-anaphora rules additionally employ the implicit

S-List. If the anaphor refers to an individual entity, it is classified as A-incompatible according to the Selectional Restriction Rules. Then, an S-List is created and the anaphor is resolved to the first element in the S-List. The following example shows how the algorithm successfully applies the first Pronominal Anaphora Rule:

10) Museum – Post office

... u2_GC_MZ_3.wav *after the second road on your right* \\
 u2_GC_MZ_4.wav *there is a building* [3] \\
 u2_GC_MZ_5.wav *and **that** is the post office* [4]

1. update DM: ... road1(sg., PATH, [in(right_region)], [second], 3.1);
 building1(sg., BUILDING, [], [], 3.2)
2. Selectional Restriction: (1) Equation construction → type constraint: BUILDING (*A)
3. properties of anaphor: X(sg. BUILDING, [], [], 4.1)
4. Pronominal Anaphora Rule: **(1) Anaphor is *A**
5. construct S-List: {building1} → resolve ‘that’ = building1

If the S-List is empty, the anaphor is classified as vague and no antecedent is specified like in the following example:

11) Museum – Grand Hotel

u3_GC_ME_1.wav *okay from the beginning **it**'s the second turning on the left* [1] \\

1. update DM: *origin=museum(sg., BUILDING)*
 destination=grand_hotel(sg., HOTEL)
 task='go from the museum to the university'(sg., ABSTRACTION)
 initial_road(sg., PATH)
2. Selectional Restriction: (1) Equation construction → type constraint: JUNCTION (*A)
3. properties of anaphor: X(sg., JUNCTION, [in(left_region)], [second], 1.2)
4. no First Sentence Rule (type of anaphor does not match type of origin or initial_road & anaphor is not *I)
5. Pronominal Anaphora Rule: **(1) Anaphor is *A**
6. empty S-List: {} → classify anaphor as vague

Conversely, if the anaphor refers to an abstract object (eventuality), it is classified as I-incompatible according to the Selectional Restriction Rules. Then, it is resolved to the element in the A-List which will be constructed as soon as the second Pronominal Anaphora Rule is applied which is demonstrated in the following example.

12) Boots (Museum) – Queens Pub

... u14_GA_CL_3.wav *instead of following the road straight on [2]
take the second right [3]*
u14_GA_CL_4.wav *it will feel like you re going only about thirty degrees
to the right [4]*

1. update DM ...
2. Selectional Restriction: (3) Argument of Verb → type constraint: EVENTUALITY (*I)
3. properties of anaphor: X(sg., EVENTUALITY, [], [], 4.1)
4. Pronominal Anaphora Rule: **(2) Anaphor is *I**
5. construct A-List: {*take the second right*} → 'it' = *take the second right*

Again, if the A-List is empty, the anaphor is classified as vague and no antecedent is specified. Finally, if it is unclear whether the anaphor refers to an individual or to an abstract object (eventuality), it is marked as ambiguous. Then, the type of the anaphor is important because, due to the preference that demonstrative pronouns more often refer to abstract objects, while personal pronouns more frequently refer to individual entities, demonstrative-anaphors are resolved to the element in the constructed A-List. Personal pronoun-anaphors are resolved to the first element in the created S-List. The next examples from the data illustrate how the third and fourth Pronominal Anaphora Rule are successfully employed.

12) Example of the 3. Pronominal Anaphora Rule

... u7_GC_CP_7.wav *carry along until you find the car park [5] and turn right into it [6]*

1. update DM: *destination=car_park(sg., PLACE)*
... *car_park1(sg. PLACE, [], [], 5.1)=destination*
2. Selectional Restriction: (6) Object of Spatial PP → type constraint: THING → ambiguous
3. properties of anaphor: X(sg., THING, [], [], 6.1)
4. no Final-Sentence-Rule → destination is mentioned recently
5. Pronominal Anaphora Rule: **(3) Pronoun is ambiguous**
6. construct S-List: {*car_park1*} → resolve 'it' = *car_park1*

13) Example of the 4. Pronominal Anaphora Rule

u2_GC_MY_1.wav *go straight on here*
u2_GC_MY_2.wav *past three roads on your left hand side [1]*
u2_GC_MY_3.wav *and the library is on the left hand side after **that** [2]*

1. update DM ...
2. Selectional Restriction: (10) ‘after’ → ambiguous (unclear whether conjunction (without complete clause) or preposition)
3. properties of anaphor: X(sg., ?, [], [], 2.2)
4. Pronominal Resolution Rule: **(4) DEM is ambiguous**
5. construct A-List: {*go straight on here past three roads on your left hand side*}
6. non-empty A-List → resolve ‘that’ = *go straight on here past three roads on your left hand side*

The last type of resolution rules are the **This-NP-Anaphora Rules** which deal with the phenomenon where ‘This-NP’-anaphors seem to refer to less salient entities in the discourse model. This means that these anaphors either refer to entities mentioned earlier in the discourse or that they refer to entities not explicitly mentioned in the discourse. However, these entities are implicitly introduced into the discourse model by certain activities like TURN-actions or by certain nouns, the so called complex- and element-entities. To handle TURN-actions, it is assumed that elliptic constructions like “*take the second left*” introduce a new discourse entity into the discourse model which stands for the implicit path-entity referred to by the expression ‘*the second left*’. The elliptic expressions are usually combined with the verb ‘take’ which describes a change of orientation in the route instruction. The other TURN-verb ‘turn’ also introduces a new discourse path-entity into the discourse model. Even though the event described by the verb ‘turn’ needs no explicit path-argument, the concept of ‘turning left or right’ involves a path-entity – the path onto which the agent turns. ‘This-NP’-anaphors frequently refer to the implicitly introduced path-entity, so that these entities are explicitly introduced into the discourse model by the anaphoric reference like in the following examples.

14) Example of Turn-Action

u5_GC_EX_2.wav *er if you **turn right** just before derry s [1] *
u5_GC_EX_3.wav *and then continue down **that road** [2] *

1. update DM: ... derrys1(sg., SHOP_BUILDING, [], [], 1.1)
you turn right just before derrys → TURN-Action: create implicit path-entity
path1(sg., PATH, [in(right_region)], [], 1.2)

15) Example of Take-Action

u6_GC_CM_3.wav *so you go over the bridge [2] but instead of carrying straight on [3]
take a right [4] \\
u6_GC_CM_4.wav carry on down **that road** [5] ...*

1. update DM: ... post_office1(sg., BUILDING, [], [], 1.1)
bridge1(sg., TRANSITION, [], [], 2.1)
take a right → ‘take the next road on your right’ (elliptic TAKE)
road1(sg., PATH, [in(right_region)], [next], 4.1)

Poesio & Modjeska (2002) describe this kind of reference to implicitly introduced entities. Not only in the corpus they analysed but also in the IBL-corpus ‘This-NP’-anaphors also refer to elements of previously mentioned complex_entities or vice versa. For this reason, the RIAR-algorithm makes use of the implicit S-List. If the anaphor type is an element_entity, the implicit S-List contains the corresponding complex_entity and vice versa. As the referent of the anaphor might be explicitly mentioned in the previous utterance or it might be introduced implicitly by a TURN-action, the algorithm firstly builds up an S-List. Only if the S-List is empty and if the anaphor type is an element_entity, will the algorithm construct an implicit S-List at the unit the anaphor occurs in. But since the entities referred to by ‘This-NP’-anaphors are not necessarily highly salient, the algorithm creates a new S-List at the previous unit. If this S-List is also empty, a new implicit S-List is constructed. This goes on until an antecedent is found within the S-List or implicit S-List of the two previous units. If an antecedent is found within these units, a new entity of the same type as the anaphor is created at the unit the anaphor occurs in and the anaphor is resolved to this new entity. The new entity is derived from the related entity. However, if no antecedent is found, which is similar to the derived entity, a new entity of the same type as the anaphor is created at the unit the anaphor occurs in. Again, the anaphor is resolved to this new entity but in contrast this entity is additionally marked as *deictic referential to the visual situation*, because it is not previously introduced into the discourse by language but by visual input at the actual situation. If the anaphor type is not an element- or complex-entity, the same procedure is applied with the exception that no implicit S-List is constructed because the entities referred to by This-NP-anaphors are not necessarily highly salient. So the

RIAR-algorithm creates a new S-List at the previous unit and if necessary also at the pre-previous unit. Other pronoun resolution techniques would not find the antecedent of the anaphor if it is not mentioned in the previous utterance because it is not salient and therefore inaccessible for the anaphor-resolution process. The following examples illustrate how these rules work in practice.

16) Example of the 1. This-NP-Anaphor Rule

u5_GC_EG_4.wav *er across straight across the next crossroads
between the post office and pizza hut [4]\ \ → **ellipsis***
u5_GC_EG_5.wav *and you will find tescos a short way along **that road** on the left [5]*

1. update DM:
 - crossroads2(sg.,JUNCTION,[between(post_office1,pizza_hut],[next], 4.1)
 - post_office1(sg., BUILDING, [], [], 4.2)
 - pizza_hut1(sg., SHOP_BUILDING, [], [], 4.3) → **ellipsis**
 - tesco1(sg., SHOP_BUILDING, [], [], 5.1)
2. type constraint for anaphor → PATH
(*on the left* specifies not the road but tesco, inference about spatial specification needed)!
3. properties of anaphor: X(sg., PATH, [], [], 5.2)
4. This-NP-Anaphor Rule: **(1) element_entity**
5. anaphor is element_entity (part_of)
6. empty S-List at unit(5) → construct implicit S-List at unit(5):{crossroads2} (entity in implicit S-List is complex_entity)
7. create new entity for element_entity in DM at unit(5) → road1(sg., PATH,[],[], 5.2)
8. resolve ‘that road’ = road1

17) Example of the 2. This-NP-Anaphor Rule

u22_GB_CL_8.wav *um you walk up few metres [8]
and then you see the huge tall building on your left [9]\ *
u22_GB_CL_9.wav *you have to go round **this building** [10]*

1. update DM: ... metres1(pl., UNIT_OF_MEASUREMENT, [], [few], 8.1)
building1(sg., BUILDING, [in(left_region)], [huge], 9.1)
2. type constraint for anaphor → BUILDING
3. properties of anaphor: X(sg., BUILDING,[],[],10.1)
4. This-NP-Anaphor: **(2) any entity**
5. construct S-List:{building1} → resolve ‘this building’ = building1

Again, the assumptions about implicit entities are domain-specific, but the phenomena described above are generally known as ‘bridging inferences’, which are

defined as “*inferences that are drawn to increase the coherence between the current and preceding parts of a text*” (Eysenck & Keane (2000), p. 528).

3.6 Here- and There-Anaphors and the SR-Algorithm

As the IBL-corpus consists of route instructions, a class of texts which contains plenty of spatial references, this section provides suggestions how some of the phenomena of spatial reference occurring in the IBL-corpus could be handled. As I will show later when the evaluation of the algorithm is given, the SR-algorithm needs further refinements. Due to time restrictions, I could not define new spatial resolution rules which would improve the performance of the algorithm. Therefore, the spatial reference algorithm developed within this project can only be regarded as first sketch of an algorithm. Nonetheless, in the discussion section I will explain how the algorithm can be improved. Since the following algorithm does not perform as efficiently as expected, I will only briefly present the rules of the algorithm (see Appendix 5). The first point to be considered is that all ‘here/there’s used syntactically are excluded. This means that all ‘here/there’s which are in the subject position are not taken into consideration. As ‘here/there’-anaphors usually refer to a place or position specified in the utterance, their antecedent can be inferred from the verb and from the positional or directional PP of the previous utterance. The general approach resembles the RIAR-algorithm; the route instructions are analogously split into units. The inputs of the algorithm are unresolved units and its outputs are resolved units. Like the A-List in the RIAR-algorithm, the SR-algorithm constructs a P-List in which the antecedent-position of the anaphor is presented. If the anaphors occur in a chain, the P-List is filled after the first anaphor is resolved so that the second and third (etc.) anaphors can be resolved to the same antecedent-position as the first one. If the P-List is empty, it is created according to the construction rules. Since the algorithm takes the predicative context of the ‘here/there’-anaphor into account, it needs nine construction rules. As the P-List usually contains spatial PPs, the construction rules determine which preposition is inserted into the P-List. This depends on the verb and the preposition(P1) used in the previous unit. To determine

6. Construction Rule: Transitive Verb in previous unit

Preconditions: There-Anaphor is in unit(U),

Transitive Verb in unit(U-1)

Actions : insert 'position: at & object of Transitive Verb' into P-List

Example: *you'll have two trees on your left and a building on your right* [1] *if you take a left*
there [2] → resolve 'there' = position: at two trees on your left and a building on your right

4. Evaluation and Discussion

In this section, first I will briefly show how the E&S-Algorithm performs on the development data of the IBL-corpus. Then, I will present the evaluation of the RIAR- and the SR-algorithms. Tables of their statistical performance are given which will be explained by discussing the general performance of the algorithms on the development data and on the test data. The discussion focuses on the main problems of the algorithms. As none of the algorithms have been implemented, the tests on the data sets are theoretically done as paper and pencil testing (the run through the unseen data is documented in Appendix 6).

4.1 Evaluation of Eckert & Strube's-Algorithm on the Development Data

Before I begin with the evaluation of the E&S-algorithm, it must be said that the E&S-algorithm does not resolve 'This-NP'-anaphors and 'here/there'-anaphors. But the results of the pronoun-anaphors are comparable to the results attained by the RIAR-algorithm. Following Eckert & Strube's categorisation for personal and demonstrative pronouns, which groups the pronouns depending on the type of antecedent they refer to, there are six anaphora-classes (cf. tables 1 and 2 below). The tables below show that the E&S-algorithm correctly resolves 53% of all personal pronouns and 57% of all demonstrative pronouns. However, mention should be made of the fact that six of the personal pronouns refer to the final destination which is not explicitly introduced into the discourse model. Therefore, the referent for the final destination is not accessible for the E&S-algorithm. Three of the incorrectly resolved demonstrative pronouns which refer to an abstract object are almost correctly resolved because the algorithm resolves them to the clause which includes the correct VP-antecedent. Nonetheless, the VPs in these constructions consist of a modal verb plus the infinitive form of the main verb like in the following example.

- 1) *okay you want to [go to boots]₁ this time [1]
and to do that₁ [2] you've got to get back to the roundabout*

As the A-List of the E&S-algorithm only stores complete clauses, it resolves the demonstrative-anaphor to the whole previous unit which is wrong. Instead, as the

anaphor is the argument of ‘do’, it should be resolved to the main verb of the previous unit. Thus, Eckert & Strube’s rules for ranking the context need some refinement in order to resolve this kind of anaphora. Overall, it cannot be denied that the E&S-algorithm performs reasonably well .

Table 1: **Personal Pronouns: 17**

	<i>Correct</i>	<i>Incorrect</i>	<i>Total</i>
PRO (IND)	7 (50%)	7 (50%)	14 (82%)
PRO (AO)	1 (100%)	/	1 (6%)
PRO (VAGUE)	1 (50%)	1 (50%)	2 (12%)
PRO- TOTAL	9 (53%)	8 (47%)	17

Table 2: Demonstrative Pronouns: 7

	<i>Correct</i>	<i>Incorrect</i>	<i>Total</i>
DEM (IND)	2 (100%)	/	2 (29%)
DEM (AO)	2 (40%)	3 (60%) but nearly correct	5 (71%)
DEM (VAGUE)	/	/	/
DEM – TOTAL	4 (57%)	3 (43%)	7

The performance of the Eckert & Strube-algorithm on their test data was better. For their own data the algorithm correctly resolved 68% of all individual anaphors (in the IBL-data 56%) and 70% of all discourse-deictic anaphors (in the IBL-data 50%). Since Byron’s PHORA-algorithm is designed and implemented using the problem-solving dialogs from the TRAINS93 corpus, testing it on data from the IBL-corpus is not possible without generating the domain-specific characteristics of the IBL-corpus (discussed in this dissertation) into the PHORA-format. Due to time and skill restrictions, I could not run the PHORA-algorithm either theoretically or in practice. But it needs to be said that the PHORA-algorithm correctly resolved 72% of all the anaphors (all personal and demonstrative pronouns) in their test data which is a quite good performance.

4.2 Evaluation of the RIAR- and SR-Algorithm on the Development Data

The performance of the RIAR-algorithm on the development data is excellent. All personal and demonstrative pronouns are correctly resolved. For the evaluation of the RIAR-algorithm Eckert & Strube’s pronoun classification is extended with respect to the additional resolution rules used by the RIAR-algorithm. Thus, how often the personal pronoun ‘it’ refers to the destination, origin, initial road or task and whether it occurs in a complex-constituent is also recorded (see table 1).

Table 1: **Personal Pronoun: 15**

	<i>Correct</i>	<i>Incorrect</i>	<i>Total</i>
PRO (destination)	6 (100%)	/	6 (40%)
PRO (origin)	/	/	/
PRO (initial_road)	/	/	/
PRO (task)	/	/	/
PRO (IND)	4 (100%)	/	4 (27%)
PRO (AO)	1 (100%)	/	1 (7%)
PRO (VAGUE)	2 (100%)	/	2 (13%)
PRO (Complex- Constituent)	2 (100%)	/	2 (13%)
PRO – TOTAL	15 (100%)	0 (0%)	15

Table 2: **Demonstrative Pronoun: 7**

	<u>Correct</u>	<u>Incorrect</u>	<u>Total</u>
DEM (origin)	/	/	/
DEM (initial_road)	/	/	/
DEM (task)	/	/	/
DEM (IND)	2 (100%)	/	2 (29%)
DEM (AO)	5 (100%)	/	5 (71%)
DEM (VAGUE)	/	/	/
DEM – TOTAL	7 (100%)	0 (0%)	7

Moreover, the RIAR-algorithm also finds the correct antecedent of all ‘This-NP’-anaphors. This type of anaphor is also categorized with respect to the types of antecedents referred to. According to the resolution rules the following types of

antecedents are specified: the initial-road, the origin and the task referred to in the first sentence, individuals, individuals implicitly introduced by ‘take’ or ‘turn’, individuals implicitly introduced by a complex- or element-entity and individuals in the visual context which are introduced by the anaphoric-reference (see table 3). These categorisations (for the antecedents of pronouns and ‘This-NP’-anaphors) depend on the knowledge of the domain of route instructions.

Table 3: **This-NP-Anaphor: 18**

	<i>Correct</i>	<i>Incorrect</i>	<i>Total</i>
This-NP (initial_road)	1 (100%)	/	1 (6%)
This-NP (origin)	/	/	/
This-NP (IND)	5 (100%)	/	5 (28%)
This-NP (TAKE-IND)	4 (100%)	/	4 (22%)
This-NP (IND) total	9 (100%)	/	9 (50%)
This-NP (IMPLICIT)	4 (100%)	/	4 (22%)
This-NP (TURN-IND)	4 (100%)	/	4 (22%)
This-NP (IMPLICIT) total	8 (100%)	/	8 (44%)
This- NP (VISUAL)	/	/	/
This-NP - TOTAL	18 (100%)	0 (0%)	18

Nevertheless, the results attained for the development data are not surprising as the resolution rules of the RIAR-algorithm are defined according to the phenomena observed in the development data. Since defining rules for spatial reference is more complicated, the results of the SR-algorithm are good, but not perfect (see table 4).

Table 4: **Here/There-Anaphor: 14**

	<i>Correct</i>	<i>Incorrect</i>	<i>Total</i>
Here	1 (100%)	/	1 (7%)
There	11 (85%)	2 (15%)	13 (93%)
Total	12 (86%)	2 (14%)	14

4.3 Evaluation of the RIAR- and SR-Algorithm on the Test Data

The RIAR-algorithm performs quite well on the test data, at least it correctly resolves 89% of all personal pronouns and 60% of all demonstrative pronouns (see table 1 and

2). Personal pronouns are problematic in that there are two cases in the test data where the ‘destination’-anaphor does not occur in the final sentence, but in the penultimate sentences. As these two cases are the only ones the algorithm gets wrong, a small change in the Final-Sentence-Rule would improve the performance of the algorithm to 100% correctness.

Table 1: **Personal Pronoun: 18**

	<i>Correct</i>	<i>Incorrect</i>	<i>Total</i>
PRO (destination)	8 (100%)	/	8 (44%)
PRO (origin)	/	/	/
PRO (initial_road)	/	/	/
PRO (task)	/	/	/
PRO (IND)	5 (71%)	2 (29%)	7 (39%)
PRO (AO)	/	/	/
PRO (VAGUE)	1 (100%)	/	1 (6%)
PRO (Complex- Constituent)	2 (100%)	/	2 (11%)
PRO – TOTAL	16 (89%)	2 (11%)	18

Table 2: **Demonstrative Pronoun: 5**

	<u>Correct</u>	<u>Incorrect</u>	<u>Total</u>
DEM (origin)	/	/	/
DEM (initial_road)	/	/	/
DEM (task)	/	/	/
DEM (IND)	2 (67%)	1(33%)	3 (60%)
DEM (AO)	1 (100%)	/	1 (20%)
DEM (VAGUE)	/	1(100%)	1 (20%)
DEM – TOTAL	3 (60%)	2 (40%)	5

For the demonstrative pronouns the RIAR-algorithm does not perform very well, but firstly there are not so many demonstrative anaphors in the test data, so that the two mistakes of the algorithm have a big influence on the performance results and secondly the vague anaphor is quite tricky to determine as shown in the following example for the test data.

difficult since I could not find an easy solution. However, it should not be forgotten that the algorithm performs well for this type of anaphora even though those phenomena occur in the test data.

Table 3: **This-NP-Anaphor: 21**

	<i>Correct</i>	<i>Incorrect</i>	<i>Total</i>
This-NP (initial_road)	2 (100%)	/	2 (9,5%)
This-NP (origin)	/	/	/
This-NP (IND)	4 (67%)	2 (33%)	6 (28,5%)
This-NP (TAKE-IND)	7 (33%)	/	7 (33%)
This-NP (IND) total	11 (85%)	2 (15%)	13 (62,5%)
This-NP (IMPLICIT)	/	/	/
This-NP (TURN-IND)	2 (100%)	/	2 (9,5%)
This-NP (IMPLICIT) total	2 (100%)	/	2 (9,5%)
This- NP (VISUAL)	3 (75%)	1 (25%)	4 (18,5%)
This-NP - TOTAL	18 (86%)	3 (14%)	21

Again, the worst performance shows the SR-algorithm. It only resolves 43% of all ‘here/there’-anaphors correctly (see table 4 below). In order to gain a better insight into the problems of the SR-algorithm, I defined several types of ‘here/there’-antecedents.

Table 4: **Here/There-Anaphor: 14**

	<i>Correct</i>	<i>Incorrect</i>	<i>Total</i>
Here	/	/	/
There	5 (83%)	1 (17%)	6 (43%)
Destination-there	/	2 (100%)	2 (14,3%)
From-there	/	1 (100%)	1 (7%)
TV-object & PP	/	2 (100%)	2 (14,3%)
Where	/	1 (100%)	1 (7%)
Vague	1 (50%)	1 (50%)	2 (14,3%)
Total	6 (43%)	8 (57%)	14

Some of the problems are relatively easy to fix. If for example a ‘here/there’-anaphor occurs in one of the final sentences of the route instruction, similarly to the personal

5. Conclusions and Further Work

In this section, I will firstly summarize the main points of the dissertation. Then, a short discussion of unsolved problems follows and finally suggestions for further work are given.

5.1 Summary

Investigating previous work in the field of anaphora resolution was necessary and useful in order to develop the RIAR-algorithm presented in this dissertation. Especially, Eckert & Strube's notion of the A- and S-List (2000), Byron's idea of additional semantic filtering (2002) and Poesio & Modjeska's (2002) analysis of the 'This-NP'-anaphor are taken into account for the conceptual design of the algorithm. The SR-algorithm partially gains from the conceptual design of the RIAR-algorithm since both algorithms use the same unit-segmentation rules and the SR-algorithm also assumes the notion of a list containing the appropriate antecedent of the anaphor. Nonetheless, the SR-algorithm uses a P-List instead of an A- or S-List. As neither algorithm is implemented, their operation is theoretically explained in the main part of the dissertation and modelled for the testing of the algorithms on the unseen data from the IBL-corpus. The results of this testing are documented and discussed in the evaluation section. The outcome of a test of the RIAR-algorithm shows that it performs well on the unseen data since it correctly resolves 89% of all personal pronouns, 60% of all demonstrative pronouns and 86% of all 'This-NP'-anaphors. In comparison, the SR-algorithm does not perform especially well because it only resolves 43% of all 'here/there'-anaphors correctly. As discussed in the evaluation, the SR-algorithm needs further refinements in order to improve performance. But there are also unsolved problems for the RIAR-algorithm which will be examined next.

5.2 Open Issues

Thus, the constructions of the A-List and the organisation of the ontology need to be reconsidered. The concept of A-List used in Eckert & Strube's algorithm and in the RIAR-algorithm is problematic. As mentioned before, the assumption of the RIAR-algorithm that only VPs are inserted into the A-List is unstable because the kind of antecedent really depends on the predicative context of the anaphor. So the anaphor might either refer to a VP or to a fact or proposition expressed in a complete clause or sentence. As described in the background section, discourse-deictic anaphors refer to different types of antecedents, e.g. events, concepts, kinds, situations, propositions, states and facts. Thus, in order to improve the resolution of discourse deictic anaphors, the lexical entries of the verbs should also store information on which argument is likely to refer to which type of abstract object. For example, if a discourse deictic anaphor occurs in the subject position of the transitive verb 'look like', it might refer to an event, or if the anaphor occurs in the object position of the verb 'assume', it refers to a proposition like in the following examples.

- 1) *Even though Tom [laughed]₁, it₁ looked like he was screaming.*
- 2) *Ann thought [that plenty of people would celebrate the Golden Jubilee of the Queen]₁. Paul didn't assume that₁.*

This refinement could improve the performance of both algorithms. But the effect on the performance of the RIAR-algorithm would not be too large, since there are not very many discourse deictic references in the data from the IBL-corpus. In the domain of route instructions, the variety of anaphoric reference is not wide. Thus, anaphoric references to propositions are unlikely to occur in the data. Moreover, Eckert & Strube (2000) point out that events and states cannot easily be distinguished, because states and events are established by complex interactions between lexical information, tense and aspect. Nonetheless, the distinction between events and states is necessary in order to prevent the algorithm from incorrectly resolving the anaphor in the following example from Eckert & Strube (2000, p. 86).

- 3) **[Mary knows French.]₁ That₁ happens frequently.*

Byron's algorithm handles this problem by using a set of referring functions which coerce proxies of the desired type into the discourse model. Depending on the syntactic and semantic analysis of the constituent, the referring functions create kind-, situation-, event-, proposition-entities, etc. which are activated so that discourse-deictic anaphors can refer to them.

Another problem of the A-List is that even if the predicative context is the same, discourse deictic reference seems to vary depending on the type of anaphor (demonstrative or pronoun). A comparison of two examples from recent research literature illustrates this phenomenon. Eckert & Strube (2000) give an example of discourse deictic anaphors occurring in a chain which fits with their context ranking rules, but Byron (2002) gives another interesting example of discourse deictic reference in a chain which does not fit with Eckert & Strube's context ranking rules. But, let us firstly look at Eckert & Strube's example again (2000, p. 75).

- 1) *and we make it so easy for them [to stay there with welfare that they can get by just signing some papers]₁*
- 2) *granted, they can do **that**₁ very easily. **It**₁'s easy to do, but look where **it**₁ puts them.*

In this example the A-List is empty when the algorithm hits the first discourse deictic anaphor ('that'). Thus, it creates a new A-List and resolves the anaphor to the newly constructed element in the A-List (see co-indexed VP and further constituents). For the subsequent discourse deictic pronouns, the algorithm finds the antecedent in the A-List.

Basically, for this example the context ranking rules work but I would like to resume an example from Byron (2002, p. 82) which shows that the referring conduct of demonstrative and personal pronouns differs. Thus, in the following example the referent of the anaphor changes depending on the type of anaphor.

- 3) *John thought about [becoming a bum]₁.*
 - a) ***It**₁ would hurt his mother and **it**₁ would make his father furious.*
 - b) *[**It**₁ would hurt his mother]₂ and **that**₂ would make his father furious.*

Even though the anaphors occur in the same predicative context, the demonstrative pronoun and the personal pronoun do not refer to the same abstract object (see index between anaphor and antecedent), since demonstrative pronouns tend to refer to

entities not in focus. Thus, the personal pronouns in 3.a) refer to the focused entity while the demonstrative in 3.b) refers to an entity not in focus. This is why Byron distinguishes between focused and activated entities, the first being potential referents of personal pronouns and the later being potential antecedents of demonstrative pronouns.

In addition to the problems with the A-List, there is uncertainty about the organisation of the ontology. The concept *thing* is split into *abstractions* and *entities*. Since the ontology is used in order to define whether an anaphor is I- or A-incompatible, this distinction might cause difficulties because anaphors marked as I-incompatible cannot refer to individual entities and vice versa. But as the following example from the data shows, sometimes I-incompatible marked anaphors refer to a concrete, individual entity.

4) Example from IBL-data

u17_GB_MC_8.wav *take this third exit*
 u17_GB_MC_9.wav *just past plymouth university on the right hand side*
 you will find [boots]₁
 u17_GB_MC_10.wav ***this**₁ is your destination*

Due to the Selectional Restriction Rules, the demonstrative pronoun is marked as I-incompatible and therefore, it is resolved to the previous VP:{*find boots*}. Nonetheless, the correct antecedent of this anaphor is '*boots*', a concrete, individual entity. In this context, it must be said that some of the things specified as *abstractions* in the ontology tend to be concrete in a route instruction. Thus, the concept of 'destination', 'middle' or 'end' are somewhat abstract on the word-level (isolated without any context), but as soon as they are used in context, for example in a route instruction, they refer to concrete entities. So 'the end of the road' or 'the middle of the car park' are specified regions or parts of other concrete entities. The concept 'destination' is usually filled with a concrete building-entity. This is the reason that the S-List includes all kinds of entities realized as NPs irrespective of whether they are *abstractions* or *entity*-things according to the ontology. Eckert & Strube (2000) discuss a related problem as they point out that there are exceptions to the correlation between NPs and concrete referents respectively between clauses and abstract

referents. The IBL-ontology shows what Eckert & Strube also observed in their data: “*there are many NPs which refer to abstract entities, and which can therefore function as antecedents for anaphors in so-called A-incompatible verbal contexts*” (Eckert & Strube 2000, p. 85). To illustrate this observation the verb ‘happen’ is used in the following example (see Eckert & Strube) because its subject position refers to an event.

- 5) *One of my friends was injured in [a big accident]₁ near South Bridge. It₁ happened yesterday.*

Eckert & Strube suggest that the capability of the algorithm could be improved by linking it to a lexical database such as WordNet. But as discussed above this would only help if the Selectional Restriction Rules and the classification of the concepts in the ontology are well defined. Finally, it needs to be said that even though the RIAR-algorithm performs quite well, the Selectional Restriction Rules and the ontology of the RIAR-algorithm need further refinements in order to handle these kinds of problems.

5.3 Further Work

In order to receive a general impression of how the RIAR- and SR-algorithms’ performance, especially when the participants do not know that they are creating the route instruction for a robot in a modelled world, I asked a couple of friends (all native speakers of English) to write some new route instructions from places in Edinburgh. There were several conditions for this task. Firstly, the route instructions should be given to a person who does not know Edinburgh and secondly, the participants should try to give their instructions as verbally as possible even though they were written down. Again, I only ran the testing of the algorithms on the new data theoretically. New vocabulary, e.g. ‘stone’ or ‘monument’ is treated as if it is part of the system which means that those words have lexical entries and that their concept is part of the ontology.

Even though I only received nine new route instructions, most of them are longer than the instructions from the IBL-corpus. Thus, the number of personal and

demonstrative pronouns is similar to the amount in the IBL development- and test data, but the amount of ‘This-NP’-anaphors and ‘here/there’-anaphors is much smaller. This might be put down to the fact that the participants of the new task did not look at the modelled world. Instead, they had to build up a model of the ‘real’ world in their head. On the one hand this memorized world might not be as detailed as the modelled world of the robot and on the other hand in the descriptions given by participants facing the modelled world of the robot there often is a tendency to spontaneously assume that the world is also at hand for the addressee. This fits well with the observation that ‘This-NP’-anaphors often refer to unfocused or visual discourse deictic entities which are more salient if the participants look at the modelled world of the robot as if they have to image the ‘real’ world. The following results show again that the RIAR-algorithm performs quite well, on the new data while the SR-algorithm performs rather badly.

Personal Pronoun: 21

	<i>Correct</i>	<i>Incorrect</i>	<i>Total</i>
PRO (IND)	19 (95%)	1 (5%)	20 (95%)
PRO (VAGUE)	/	1 (100%)	1 (5%)
PRO – TOTAL	19 (90%)	2 (10%)	21

Demonstrative Pronoun: 10

	<i>Correct</i>	<i>Incorrect</i>	<i>Total</i>
DEM (IND)	5 (83%)	1 (17%)	6 (60%)
DEM (AO)	1 (33%)	2 (67%)	3 (30%)
DEM (VAGUE)	/	1 (100%)	1 (10%)
DEM – TOTAL	6 (60%)	4 (40%)	10

This-NP-Anaphor: 9

	<i>Correct</i>	<i>Incorrect</i>	<i>Total</i>
This-NP (origin)	1 (100%)	/	1 (11%)
This-NP (IND)	4 (100%)	/	4 (45%)
This-NP (TAKE-IND)	/	/	/
This-NP (IND) total	4 (100%)	/	4 (45%)
This-NP (IMPLICIT)	/	1 (100%)	1 (11%)
This-NP (TURN-IND)	2 (100%)	/	2 (22%)
This-NP (IMPLICIT) total	2 (67%)	1 (33%)	3 (33%)
This- NP (VISUAL)	/	1 (100%)	1 (11%)
This-NP - TOTAL	7 (78%)	2 (22%)	9

Here/There-Anaphor: 5

	<i>Correct</i>	<i>Incorrect</i>	<i>Total</i>
There	1 (50%)	1(50%)	2 (40%)
There (vague)	1 (33%)	2 (67%)	3 (60%)
Total	2 (40%)	3 (60%)	5

In order to reduce the number of rules in the SR-algorithm, one could try to define one general rule. This new rule should be used when the utterance previous to the utterance with the anaphor describes a Verb of Motion. The new rule could be defined as follows: resolve the ‘there’-anaphor to the position where the robot is, after it has performed the activity described in the previous utterance. Applying this rule in the following examples resolves the ‘there’-anaphor either to the position where the robot is after having gone to the post office (see example 1) or to the position where the robot is after having taken a sort of right turn by dixon's.

- 1) *go to the post office and turn left **there***
- 2) *then take a sort of right turn by dixon's, carry on up **there***

But firstly the positions determined by this rule are somewhat vague (which is not too problematic because ‘there’-anaphors tend to be a little vague) and secondly this rule does not work for other verbs or descriptions like ‘*there is a big building*’ or ‘*then*

you will see the museum' . Thus, additional rules still need to be defined. Moreover, this rule is not tested, so it is not clear whether it works successfully.

In addition to this, it might be useful to draw up a temporal ontology for all verbs used in the route instructions, since by virtue of such an ontology every kind of eventuality can be defined (e.g. process, event, state etc.). Then, the 'here/there'-anaphors could be resolved by different rules defined with respect to the type of eventuality mentioned in the previous utterance. The only problem is that constructing a temporal ontology is quite difficult because the type of eventuality depends not only on the semantic of the verb, it moreover relies on the interaction between the verb and its arguments and on the aspect of the construction. Thus, '*he runs*' is a process while '*he runs to the shop*' is an event. The classification of the verbs and the prepositions used in the IBL-lexicon attempts to partly capture these features (see lexical entries of verbs and prepositions with its referring position).

Finally, the RIAR-algorithm (and an improved version of the SR-algorithm) should be implemented as part of the existing DRT resolution component within the framework of the IBL-project. Thus, the results of the project: *Designing an Anaphora Resolution Algorithm for Route Instructions* might be useful for the IBL-project.

Acknowledgements

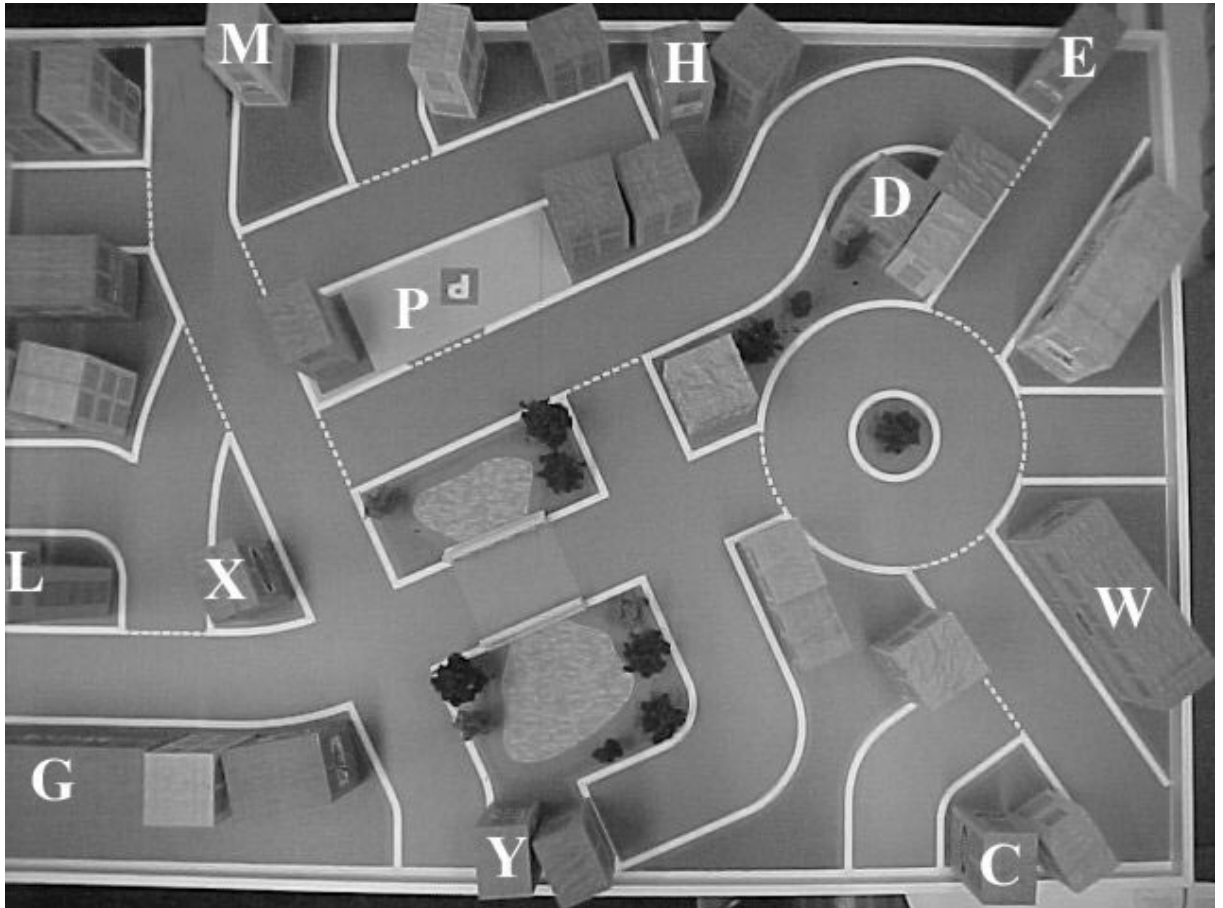
I would like to thank my supervisors Bonnie Webber and Johan Bos for advice and support on this project and for discussing the central issues of this work. My studies at the University of Edinburgh were funded by the Friedrich-Ebert-Stiftung e.V. (and by my parents). Moreover, thanks to all the people who wrote further route instructions and thanks especially to Beth Nuttall who proof-read the dissertation.

Bibliography

- Bugmann, G.; Stanislao, L.; Kyriacou, T.; Klein, E., Bos, J. and Coventry, K. (2001). Using Verbal Instructions for Route Learning: Instruction Analysis. In *Proc. TIMR'01 – Towards Intelligent Mobile Robots*, Manchester.
- Byron, Donna K. (2002). Resolving Pronominal Reference to Abstract Entities. In *Proceedings of ACL 2002*, p. 80-87.
- Eckert, Miriam and Strube, Michael (2000). Dialogue Acts, Synchronising Units and Anaphora Resolution. In *Journal of Semantics*, 17, p. 51 –89.
- Eschenbach, C.; Habel, C.; Kulik, L.; Schmidtke, H. and Tschander, L. (to be published 2002). In C. Freska, C; Brauer, W.; Habel, C. and Wender, K. (eds.) *Spatial Cognition III*. Berlin.
- Eysenck, Michael W. and Keane, Mark T. (2000). *Cognitive Psychology*, East Sussex.
- Grosz, B; Joshi, A. and Weinstein, S. (1995). Centering: A framework for modeling the local coherence of discourse. In *Computational Linguistics 21*, p. 205-225.
- Grosz, B. and Sidner, C. (1986). Attention, Intentions, and the Structure of Discourse. In *Computational Linguistics 12*, p. 175-204.
- Haegeman, Lilian (1991). *Introduction to Government and Binding Theory*, Oxford.
- Navarretta, Costanza (2000). Abstract Anaphora Resolution in Danish. In L. Dybkjaer, K. Hasida and D. Traum (eds.) *Proc. Of the 1st SIGdial Workshop on Discourse and Dialogue*, p. 56-65.
- Nissim, Malvina (unpublished document) *More on Co-reference*, Tutorial slide of TNLP2, 2002).
- Poesio, Massimo and Modjeska, Natalia N. (2002). The THIS-NPs Hypothesis: A Corpus-Based Investigation. In *DAARC' 2002*, September, Lisbon, Portugal.
- Sidner, C. (1983). Focusing in the comprehension of definite anaphora. In M. Brady and R. Berwick (eds.). *Computational Models of Discourse*, p. 363-394.
- Webber, Bonnie Lynn (1991). Structure and Ostension in the Interpretation of Discourse Deixis. In *Language and Cognitive Processes*, 6, p. 107-135.

Appendices

APPENDIX 1 MAP OF THE MODELLED WORLD



Landmarks:

M – museum

P – car park

university

G – tesco

H – hospital

L – the queens pub

Y – library

E – the grand hotel

X – post office

C – boots

D – safeway

W –

APPENDIX 2 SELECTIONAL RESTRICTIONS and INFERENCE RULES

1. SELECTIONAL RESTRICTION RULES

(Following Eckert & Strube 2000; Byron 2002)

It & this & that

- exclude → syntactical pronoun / demonstrative adverb (e.g. *the water is not that cold*)
 - A*- & I*-Incompatibility or Ambiguity of X-position based on Ontology (LEXICON)
 - I-Incompatible (*I): Anaphors in the x-position cannot refer to individual, concrete entities
 - A-Incompatible (*A): Anaphors in the x-position cannot refer to abstract entities
 - Restrictions for arguments of verbs and prepositions
 - Type Constraints of X-position based on Ontology (LEXICON)
- (1) **Equating constructions** where a pronominal referent is equated with
 - a) an abstract object → *I, e.g. *x is making it easy, x is a suggestion*
 - b) a concrete individual referent → *A, e.g. *x is a car, x is a nice place to go*
 - BE-Complement (interpreted as equal) → subject and complement-NP are entities of the same type of concept; restricted to hyponym is hypernym e.g. *boots is a building* but not *a building is boots*
- (2) **Copula constructions** whose adjectives can only be applied to
 - a) abstract entities → *I, e.g. *x is true, x is correct, x is right*
 - b) concrete entities → *A, e.g. *x is expensive, x is tasty, x is loud*
 - Copula-Adjective → constrains subject (and vice versa)
- (3) **Arguments of verbs** (check in lexicon & ontology)
 - a) prepositional attitude verbs & arguments of verb which mainly take S'-complements → *I, e.g. *assume x, say x*
 - b) describing physical contact/stimulation, which are generally not used metaphorically → *A, e.g. *eat x, drink x*
 - c) Object of do (*do x*) → *I
- (4) **Movement with Agent in Object Position:** examples from the actual data
 - *x will **take** you to the museum*
 - *x will **bring** you to the museum*
 - *x will **get** you to the museum*
 - *x will **get** you **back** to the museum*
 - a) if path-entity in S-List & no Verb of Motion in previous unit → *A & resolve anaphor to path-entity
 - b) Verb of Motion in previous unit **or** no path-entity in S-List → *I & resolve anaphor to element in A-List
- (5) **Arguments of conjunctions** → *I
 - (Anaphoric referent is equated with 'reason' e.g. *x is because I like her*) → *I
 - (Anaphor occurs in cleft construction with *how, why* e.g. *x is why he's late*) → *I

(6) **Object of spatial PP** is usually a thing → ambiguous: either *A or *I

Examples:

a) *into x* → ambiguous (thing → either *A or *I)

b) *after/before x* → ambiguous (either preposition → *A or conjunction → *I)

2. SPATIAL INFERENCE RULES

(1) **Spatial-PP-Inference:** NP & Position-PP (left, right)

- Positional-PP is interpreted with respect of the actual position of the robot
→ NP-referent is usually an entity (*A)

Expressions for Left- or Right-region (of the robot):

on your left/right hand side = on your left/right = on the left/right hand side = on the left/right

→ in(left_region) of the robot → fluent

→ in(right_region) of the robot → fluent

- different cases: *the left hand side of the street* → in(left_region_of(street1))
the left hand side of the road → in(left_region_of(road1))

(2) **Special Inference/Reasoning for Verb-PP-attachment**

- inference on pronoun in object position:
*you will see it on your left or it is on your left or
it is on your left there*
→ referent of **it** is in(left_region) of the robot & it is usually an entity (*A)

APPENDIX 3 IBL-LEXICON and ONTOLOGY

VERBS

Verbs of Motion

Movement → sub(TYPE), obj(TYPE), pp(TYPE)

- come (IV) → sub(human_person, path), pp(directional)
- double back (IV) → sub(human_person, path), pp(directional)
- get back (IV) → sub(human_person, path), pp(directional)
- go (IV) → sub(human_person, path), pp(directional)
- go (PPCV) → sub(human_person), pp(spatial)
- go (TV) → sub(human_person), obj(route)
- head (IV) → sub(human_person, path), pp(directional)
- travel (IV) → sub(human_person), pp(directional)
- walk (PPCV) → sub(human_person), pp(spatial)
- pass (TV) → sub(human_person, path), obj(thing)

Continue

- carry on (IV) → sub(human_person, path), pp(spatial)
- carry on (PPCV) → sub(human_person, path), pp(spatial)
- continue (along) (IV) → sub(human_person, path), pp(spatial)
- continue (PPCV) → sub(human_person, path), pp(spatial)

- follow on (IV) → sub(human_person, path), pp(spatial)
- follow on (TV) → sub(human_person, path), obj(path, roundabout)
- follow (TV) → sub(human_person), obj(path, roundabout)

Cross

- cross (TV) → sub(human_person, path), obj(path)

Reach

- come **to** (TV) → sub(human_person, path), obj(thing)=final_position
- access (TV) → sub(human_person, way), obj(thing)=final_position
- get **to** (TV) → sub(human_person, path), obj(thing)=final_position
- hit (TV) → sub(human_person, path), obj(thing)=final_position
- meet (TV) → sub(human_person, path), obj(thing)=final_position
- reach (TV) → sub(human_person, path), obj(thing)=final_position

Turn

- turn (PPCV) → sub(human_person, path), pp(spatial or just positional)
- turn (TV) → sub(human_person), obj(path)
- take (TV) → sub(human_person, path), obj(way)

Direction

- bend (IV) → sub(path, human_person), pp(spatial)
- bear (PPCV) → sub(path, human_person), pp(spatial)

Start

- start (IV) → sub(human_person, object), pp(positional)
- exit (TV) → sub(human_person, path), obj(artefact)
- leave (TV) → sub(human_person, path), obj(artefact)

Stop

- park (IV) → sub(human_person), pp(positional)
- stop (IV) → sub(human_person, object), pp(positional)

Verbs of Location

- see (TV) → sub(human_person), obj(thing)
- look for (TV) → sub(human_person), obj(thing)
- find (TV) → sub(human_person), obj(thing)
- face (TV) → sub(entity), obj(thing)
- get (TV) → sub(human_person), obj(thing)
- have (TV) → sub(entity), obj(thing), pp(positional)

Descriptive & other Verbs

- do (TV) → sub(human_person, path), obj(entity)
- do (IV) → sub(human_person, path), comp(eventuality)
- look like (TV) → sub(entity), obj(entity)
- look like (TV) → sub(eventuality), obj(eventuality)
- discuss (TV) → sub(human_person), obj(entity, eventuality)
- mention (TV) → sub(human_person), obj(entity, eventuality)
- recall (TV) → sub(human_person), obj(entity, eventuality)

Propositional Attitude Verb

- assume (PAV) → sub(human_person), prop(eventuality)
- feel like (PAV) → sub(eventuality), prop(eventuality)
- make sure (PAV) → sub(human_person, eventuality), prop(eventuality)
- remember (PAV) → sub(human_person), prop/quest(eventuality, entity)

Modal Verb

- have (TCV) → modal-verb
- need (TCV) → modal-verb
- want (TCV) → modal-verb

PREPOSITIONS

Directional Prepositions preposition & obj(TYPE)

Goal (G)

- in front of (both) → obj(entity) – *go in front of the building*
→ **position: in front of** NP
- in (both) → obj(thing) – *go in the middle of the quadrangle*
- into → obj(thing)
→ **position: in** NP
- onto → obj(way)
→ **on** NP
- till → obj(thing)
- to → obj(thing)
→ **position: at** NP

Partial Goal (PG)

- Partial_Goal-PP Restrictions: **BEHIND** → final position is behind NP

- past-PP – *go past the lake*
- over-PP – *go over the bridge*
- across-PP – *go across the crossroads*
- around/round-PP – *go around the building*
- through-PP – *go through the tunnel*
- behind-PP – *go behind the building*
- behind (both) → obj(entity) – *go behind the university*
- across → obj(way, transition, car park)
- over (both) → obj(way, transition) – *go over the bridge*
- past (both) → obj(entity) – *go past the lake*
- through → obj(thing)

→ **position: behind** NP

- around = round (both) → obj(thing) – *go around the building*

→ **position: at** NP

No-Goal (NG)

- along → obj(path, building, car park)
- down → obj(path)
- on = upon (both) → obj(path) – *go (up)on the street*

- up (both) → obj(path) – *go up the street*
- **position:** along NP

Source → preposition & obj(thing)

- from (starting point) → **position:** from NP
- off (starting point) → **position:** off NP

Direction (D)

- for (direction) → obj(entity)
- towards (direction) → obj(thing)
- in the direction (of) → obj(thing) – *go in the direction of the university*
- **position:** at NP

Comment:

- Directional-PP: *to the left/right* → specifies the direction/orientation of a movement on a path e.g. *go round to the left* → directional PP: *to the left* specifies the adverb: *round* & it specifies the direction of an implicit path

e.g. *walk to the left* → directional PP: *to the left* specifies the direction of an implicit path

Positional Prepositions → preposition & obj(thing)

- about
- after
- around = round (both) – *the museum is around the corner*
- at
- before (both) – *the university is before the museum*
- behind (both) – *the museum is behind the next building*
- between
- by
- in (both) – *the museum is in the next street*
- in front of (both)
- on = upon (both) – *the house is (up)on the top of the mountain*
the museum is on your left
- opposite
- outside
- over (both) – *the bridge is over the lake*
- past (both) – *the museum is past the grand hotel*
- up (both) – *the castle is up the hill*
- **position:** preposition NP

Comment:

- Positional-PP: *on your left/right hand side* = *on your left/right* = *on the left/right* → specify → refer to position of entity
 - a) previous NP-Object → usually entity (*A)
 - b) Subject-NP in Copula-construction → usually entity (*A)
 - c) indirectly Pronoun in Subject-NP position of Copula-construction → pronoun refers usually to entity (*A)

e.g. *go past the lake on your left*

the post office is on your left hand side
it is on your left there

Other Prepositions

- for (intention, purpose) - *a letter for you, clothes for children*
- of
- with
- NP(feature) **of** NP(entity) → specified part of entity → specified entity (feature) e.g. local-regional-info **of** entity
- NP(entity) **with** NP(feature) → entity **with** feature → specified entity (entity)

Preposition modifier

- just
- up
- down
- half way
- a short way

CONJUNCTIONS

(between clauses and sentences) → s'-comp(event-AO)

- after → time of events: s'-comp(event-AO) ambiguous cf. preposition 'after'
- as → time of events: s'-comp(event-AO)
- before → time of events: s'-comp(event-AO)
- as if → description / comparison of events: s'-comp(event-AO)
- as though → description / comparison of events: s'-comp(event-AO)
- as soon as → time of events: s'-comp(event-AO)
- until → time of events: s'-comp(event-AO)
- till → time of events: s'-comp(event-AO)
- without → negation of event: s'-comp(event-AO)
- instead of → negation of event: s'-comp(event-AO)

NOUNS

noun → hyper(TYPE), related(expression), adj(TYPE)

- ❖ entity → concrete, individual object – **IND**
- ❖ abstraction → abstract object – **AO**

- abstraction → hyper(thing)
- amount → hyper(measure)
- angle → hyper(shape), related(corner, line), adj(sharp)
- area → hyper(region)
- artefact → hyper(object)
- beginning → hyper(location), related(back_location, destination, front_location)
- bend → hyper(path), related(turn, road, street, exit_way, branch, fork, bend), adj(sharp)
- bible → hyper(book)
- book → hyper(artefact)
- boots → hyper(shop_building), related(derrys, dixons, pc_world,

- bottom → hyper(region), related(end, quarter, side)
- branch → hyper(path), related(road, street, exit_way, fork, bend, turn)
- bridge → hyper(transition)
- building → hyper(structure_construction)
- car → hyper(vehicle)
- car_park → hyper(place), related(park_area, parking_lot, parking_place, parking_space)
- center → hyper(area), related(middle, quadrangle)
- convertible → hyper(car)
- corner → hyper(shape), related(angle, line), adj(sharp)
- crossroads → hyper(junction), related(crossing, t-junction, junction, turning), consist_of(road, street, branch, fork)
- crossing → hyper(junction), related(crossroads, t-junction, junction, turning), consist_of(road, street, branch, fork)
- definite_quantity → hyper(measure)
- degree → hyper(unit_of_measurement), related(meter, yard)
- derrys → hyper(shop_building), related(boots, dixons, pc_world, pizza_hut, tesco)
- destination → hyper(location), related(back_location, beginning, front_location), adj(ORDINATION)
- direction → hyper(route), adj(DIRECTION)
- dixons → hyper(shop_building), related(boots, derrys, pc_world, pizza_hut, tesco)
- door → hyper(structure_construction)
- end → hyper(region), related(bottom, quarter, side)
- exit_way → hyper(path), related(street, road, branch, fork, bend, turn)
- entity → hyper(thing), adj(SIZE, LOCATION, COLOUR, ORDINATION)
- fork → hyper(path), related(bend, branch, turn, road, street, exit_way)
- hospital → hyper(building), related(house, library, museum, post_office, skyscraper, train_station)
- hotel → hyper(building)
- house → hyper(building), related(hospital, library, museum, post_office, skyscraper, train_station)
- human_person → hyper(organism)
- idea → hyper(abstraction)
- junction → hyper(junction)
- lake → hyper(water)
- library → hyper(building), related(house, hospital, museum, post_office, skyscraper, train_station)
- line → hyper(shape), related(corner, angle), adj(COLUOR, dotted, broken, complete)
- location → hyper(abstraction)

- marking_point → hyper(point)
- measure → hyper(relation)
- meter → hyper(unit_of_measurement), related(degree, yard)
- middle → hyper(area), related(center, quadrangle)
- minute → hyper(time), related(moment)
- moment → hyper(time), related(minute)
- multiple → hyper(-)
- museum → hyper(building), related(house, library, hospital, post_office, skyscraper, train_station)
- object → hyper(entity)
- organism → hyper(entity)
- park_area → hyper(place), related(car_park, parking_lot, parking_place, parking_space)
- parking_lot → hyper(place), related(car_park, park_area, parking_place, parking_space)
- parking_place → hyper(place), related(car_park, parking_lot, park_area, parking_space)
- parking_space → hyper(place), related(car_park, parking_lot, parking_place, park_area)
- path → hyper(way), adj(DISTANCE, bendy)
- participant → hyper(human_person)
- pc_world → hyper(shop_building), related(boots, derrys, dixon, pizza_hut, tesco)
- pizza_hut → hyper(shop_building), related(boots, derrys, dixon, pc_world, tesco)
- place → hyper(structure_construction)
- plant → hyper(organism)
- point → hyper(location)
- pond → hyper(lake), related(pool)
- position → hyper(point)
- post_office → hyper(building), related(house, library, hospital, skyscraper, train_station)
- pub → hyper(building)
- quadrangle → hyper(area), related(center, middle)
- region → hyper(location), adj(LOCATION)
- relation → hyper(idea)
- road → hyper(path), related(street, exit_way, branch, fork, bend,turn)
- roundabout → hyper(junction), consist_of(exit_way, street, road)
- route → hyper(location), adj(DISTANCE)
- safeway(s) → hyper(shop_building),related(boots, derrys, dixon, pc_world, pizza_hut, tesco)
- set → hyper(multiple)
- shape → hyper(idea), adj(LOCATION)
- shop_building → hyper(building)
- side → hyper(region)
- singleton → hyper(-)

- street → hyper(path), related(road, exit_way, branch, bend, turn, fork)
- structure_construction → hyper(artefact)
- substance_matter → hyper(object)
- system → hyper(participant), related(user)
- tesco(s) → hyper(shop_building), related(boots, derrys, dixons, pc_world, pizza_hut)
- the_grand_hotel → hyper(hotel)
- the_queens_pub → hyper(pub)
- the_university_of_plymouth → hyper(university)
- thing → hyper singleton)
- train_station → hyper(building), related(house, library, hospital, post_office, skyscraper)
- tree → hyper(plant)
- turn → hyper(path), related(road, street, exit_way, branch, bend, fork), adj(sharp)
- turning junction) → hyper(path), related(crossing, crossroads, t-junction, consist_of(road, street, branch, fork)
- t-junction → hyper(junction), related(crossing, crossroads, turning, junction), consist_of(street, road, branch, fork)
- unit_of_measurement → hyper(definite_quantity)
- university → hyper(building)
- vehicle → hyper(transport)
- water → hyper(substance_matter)
- way → hyper(artefact)
- yard → hyper(unit_of_measurement), related(meter, degree)

ADJECTIVES

Adjectives & Adverbs → adj(SET), antonym(SET)

ORDINATION → entity (IND -*A) & destination (AO -*I)

- adj(actual=current), antonym(previous, immediate, final)
- adj(previous), antonym(actual, immediate, final)
- adj(next=immediate=following), antonym(actual, final, previous)
- adj(final), antonym(next, previous, actual)
- adj(first), antonym(final)
- adj(second), antonym(second...)

DIRECTION → direction (AO -*I)

- adj(anticlockwise=counterclockwise), antonym(clockwise)
- adj(clockwise), antonym(anticlockwise)

LOCATION → entity(IND - *A), shape(AO -*I), region(AO -*I)

- adj(diagonal)
- adj(opposite)
- adj(left=left hand), antonym(right)
- adj(right=right hand), antonym(right)

SHAPE

- adj(bendy) → path(IND -*A)
- adj(complete), antonym(dotted, broken) → line(AO -*I)
- adj(dotted), antonym(complete) → line(AO -*I)
- adj(broken), antonym(complete) → line(AO -*I)
- adj(sharp) → bend(IND -*A), turn(IND -*A), turning (IND -*A), angle(AO -*I), corner(AO -*I)

COLOUR → entity(IND -*A), line(AO -*I)

- adj(beige), antonym(grey, white)
- adj(grey) antonym(beige, white)
- adj(white), antonym(beige, grey)

SIZE → entity(IND -*A)

- adj(big=huge=tall), antonym(little)
- adj(little=slight=small), antonym(big)

DISTANCE → entity(IND -*A), route(AO -*I)

- adj(far), antonym(short)
- adj(short), antonym(far)
- adj(slow)

REST → not specified whether they refer to IND or AO: if PRO → *A & if DEM → *I

- adj(basic)
- adj(direct)
- adj(exact)
- adj(existent)
- adj(improbable)
- adj(insensible =invisible=obvious)
- adj(main)
- adj(plausible)
- adj(same)
- adj(sensible)
- adj(unlikely)
- adj(unreasonable)
- adj(usual)

APPENDIX 4 RIAR-ALGORITHM and RIAR-RESOLUTION RULES

Input: set of units with unresolved anaphors, discourse model (DM)

Output: set of units with resolved anaphors

For each unit(U):

- update DM
- for each unresolved anaphor(A) in unit(U):
 - generate properties for anaphor(A)
 - apply rules to anaphor(A)

Update Discourse Model

For each unit(U):

- generate properties for discourse entity in unit(U)
- insert discourse entity into DM at unit(U)

GENERATE PROPERTIES OF ANAPHOR(A)

1. Anaphor(A) is personal or demonstrative pronoun with constraints (inference)

Precondition: Anaphor(A) is personal or demonstrative pronoun

Anaphor(A) is specified by adjective and/or PP

Action: apply selectional restrictions on anaphor(A) → type constraint for anaphor(A),
take adjective and/or PP as additional constraints for anaphor(A)

(cf. Selectional Restrictions and Ontology)

2. Anaphor(A) is personal or demonstrative pronoun without constraints

Precondition: Anaphor(A) is personal or demonstrative pronoun

Action: apply selectional restrictions on anaphor(A) → type constraint for anaphor(A)

3. Anaphor(A) is This-NP-anaphor

Precondition: Anaphor(A) is This-NP-anaphor(N)

Action: check type of NP in anaphor in ontology

→ type of NP is type constraint for anaphor(A)

APPLY RESOLUTION RULES TO ANAPHOR

1. Complex-Constituent Rule

1. Anaphor is part of Complex-constituent (PRO)

Preconditions: Anaphor(A) in unit(U) is pronoun(P),

Pronoun(P) is part of PP which specifies noun **or**

Pronoun(P) is part of PP in relative clause

Action: resolve pronoun(P) to nearest noun which c-commands PP which includes pronoun(P)

Examples:

- *the lake with the bridge on it* → refers to 'head'-NP of complex-NP
- *the tree by the lake with the flower on it* → refers to 'noun' in PP 'by the lake'
- *the turning which has a bridge on it* → refers to 'head'-NP of relative-clause

C-Command

A c-commands B iff A does not dominate B and every X that dominates A also dominates B.
(Lilian Haegeman (1991). *Introduction to Government and Binding Theory*. Cambridge: p.135)

2. First Sentence Rules

1. Anaphor refers to origin (PRO, DEM, DET)

Preconditions: anaphor(A) is in unit(1),

type of anaphor(A) is BUILDING

Actions: construct S-List at unit(A) → resolve anaphor(A) to first element of S-List (**cf. S-List**)

if empty S-List → resolve anaphor(A) to origin(X) in DM

2. Anaphor refers to start-road (PRO, DEM, DET)

Preconditions: anaphor(A) is in unit(1),

type of anaphor(A) is PATH

Actions: construct S-List at unit(1) → resolve anaphor(A) to first element of S-List

if empty S-List → resolve anaphor(A) to start_road(X) in DM

3. Anaphor refers to task (PRO, DEM)

Preconditions: anaphor(A) is in unit(1),

anaphor(A) is *I

Action: resolve anaphor(A) to task(X) in DM

3. Final Sentence Rule

1. Anaphor refers to task (DEM, PRO)

Preconditions: anaphor(A) is in final_unit(U),

anaphor(A) is *I,

Action: resolve anaphor(A) to task(X) in DM

2. Anaphor refers to destination (PRO)

Preconditions: anaphor(A) is pronoun(P) in final_unit(U),

Pronoun(P) is *A,

properties of pronoun(P) match properties of destination(X),

destination(X) is not mentioned in unit(U) **or** in unit(U-1) **or** in unit(U-2)

Action: resolve pronoun(P) to destination(X) in DM

4. Pronominal Anaphora Rules

1. Anaphor is *A (PRO, DEM)

Precondition: anaphor(A) in unit(U) is *A

Actions: construct S-List at unit(U) → resolve anaphor(A) to first element of S-List

if empty S-List → classify anaphor(A) as vague

2. Anaphor is *I (DEM, PRO)

Precondition: anaphor(A) in unit(U) is *I

Actions: construct A-List at unit(U) → resolve anaphor(A) to element in A-List (**cf. A-List**)

if empty A-List → classify anaphor(A) as vague

3. Pronoun is ambiguous (PRO)

Preconditions: anaphor(A) is pronoun(P) in unit(U),

Pronoun(P) is ambiguous

Actions: construct S-List at unit(U) → resolve pronoun(P) to first element in S-List,

if empty S-List → construct A-List **and** resolve pronoun(P) to element in A-List,

else classify pronoun(P) as vague

4. Demonstrative is ambiguous (DEM)

Preconditions: anaphor(A) is demonstrative(D) in unit(U),

Demonstrative(D) is ambiguous

Actions: construct A-List at unit(U) → resolve demonstrative(D) to element A-List,
if empty A-List → construct S-List **and**
 resolve demonstrative(D) to first element S-List,
else classify demonstrative(D) as vague

5. This-NP-Anaphora Rules

1. Anaphor is element-entity or complex-entity

Preconditions: Anaphor(A) in unit(U) is This-NP(N),
 This-NP(N) is element_entity or complex_entity

Actions:

construct S-List → resolve This-NP(N) to first element of S-List,
if empty S-List at unit(U) → construct implicit S-List at unit(U),
if non-empty implicit S-List at unit(U) → create new entity(E) of same
type as element_entity at unit(U) **and** resolve This-NP(N) to new entity(E),
elsif construct S-List at unit(U-1) → resolve This-NP(N) to element of S-List,
if empty S-List at unit(U-1) → construct implicit S-List at unit(U-1),
if non-empty implicit S-List at unit(U-1) → create new entity(E) of same type
as element_entity at unit(U) **and** resolve This-NP(N) to new entity(E),
elsif construct S-List at unit(U-2) → resolve This-NP(N) to element of S-List,
if empty S-List at unit(U-2) → construct implicit S-List at unit(U-2),
if non-empty implicit S-List at unit(U-2) → create new entity(E) of same
type
as element_entity at unit(U) **and** resolve This-NP(N) to new entity(E),
else create new entity(E) of same type as anaphor & resolve This-NP(N) to new
entity(E) & mark new entity(E) as deictic reference to visual situation

Implicit S-List

The implicit S-List includes the complex_entity which is related to the element_entity and vice versa.

2. Anaphor is This-NP

Precondition: Anaphor(A) in unit(U) is This-NP(N),

Actions: construct S-List → resolve This-NP(N) to first element of S-List,
if empty S-List at unit(U) → construct S-List at unit(U-1),
if non empty S-List at unit(U-1) → resolve This-NP(N) to first element of S-List,
if empty S-List at unit(U-1) → construct S-List at unit(U-2),
if non empty S-List at unit(U-2) → resolve This-NP(N) to first element of S-List,
if empty S-List at unit(U-2) → construct S-List at unit(U-3),
if non empty S-List at unit(U-3) → resolve This-NP(N) to first element of S-List,
else create new entity(E) of same type as anaphor & resolve This-NP(N) to new
entity(E) & mark new entity(E) as deictic reference to visual situation

APPENDIX 5 SR-ALGORITHM and SR-RESOLUTION RULES

- exclude syntactical 'there' → there in subject position
- there usually refers to place/position so its antecedent can be inferred from verb and/or positional or directional PP of previous unit

Input: set of unresolved units

Output: set of resolved units

For each unresolved There-anaphor(T) in unit(U):

- check P-List at unit(U),
- if empty P-List at unit(U) → construct P-List at unit(U),
- **if** non-empty P-List → resolve There-anaphor to element in P-List'
else classify There-anaphor(T) as vague

CONSTRUCT P-LIST (Position)

1. 'from There-Anaphor' in First sentence

Preconditions: There-Anaphor is in unit(1),

There-Anaphor is part of from-PP

Actions : insert 'origin(X) from DM' into P-List

Example: *from **there** carry on to the end of the road* → resolve 'there' = origin=museum

2. There-Anaphor in First sentence

Precondition: There-Anaphor is in unit(1),

Actions : insert 'position: from & origin(X) from DM' into P-List

Example: *go straight on **here*** → resolve 'here' = position: from origin=museum

3. Intransitive Verb in previous unit

Preconditions: There-Anaphor is in unit(U),

Intransitive Verb in unit(U-1)

Actions : check type of preposition in ontology,

if positional PP(P) in unit(U-1) → insert 'position: positional PP(P)' into P-List,

elsif directional PP(P) in unit(U-1) → insert 'position:' (from ontology)

& 'object of directional PP(P)' into P-List

Example: *go straight to the bottom of the road almost / and it's on your left **there*** → resolve 'there' = position: at the bottom of the road almost

4. Copula-Construction in previous unit

Preconditions: There-Anaphor is in unit(U),

Copula-Construction in unit(U-1)

Actions : check type of preposition in ontology,

if positional PP(*on the left or on the right*) is the only in unit(U-1)

→ insert 'position: at & subject of copula-construction' into P-List

elsif positional PP(P) in unit(U-1) → insert 'position: positional PP(P)' into P-List,

elsif directional PP(P) in unit(U-1) → insert 'position:' (from ontology)

& 'object of directional PP(P)' into P-List

Examples:

- *the museum is on the left / turn left there* → resolve ‘there’ = position: at the museum
- *the museum is behind the post office / and turn left **there*** → resolve ‘there’ = position: behind the post office

5. There-construction in previous unit

Preconditions: There-Anaphor is in unit(U),
There-Construction in unit(U-1)

Actions : check type of preposition in ontology,
if positional PP(*on the left or on the right*) is the only in unit(U-1)
→ insert ‘position: at & object of there-construction’ into P-List
elsif positional PP(P) in unit(U-1) → insert ‘position: positional PP(P)’ into P-List,
elsif directional PP(P) in unit(U-1) → insert ‘position:’ (from ontology)
& ‘object of directional PP(P)’ into P-List

6. Transitive Verb in previous unit

Preconditions: There-Anaphor is in unit(U),
Transitive Verb in unit(U-1)

Actions : insert ‘position: at & object of Transitive Verb’ into P-List

Example: *you’ll have two trees on your left and a building on your right / if you take a left **there*** → resolve ‘there’ = position: at two trees on your left and a building on your right

7. PASS or CROSS in previous unit(U)

Preconditions: There-Anaphor is in unit(U),
Cross- or Pass-Verb in unit(U-1)

Actions : insert ‘position: behind & object of Cross- or Pass-Verb’ into P-List

Example: *you’ll pass another road on the left and the car park is on the right from **there*** → resolve ‘there’ = position: behind another road on the left

8. Topicalization of PP in same unit

Preconditions: There-Anaphor is in unit(U),
Positional PP(P) is topicalized in unit(U),
There-Anaphor(T) is object in unit(U)

Actions : insert ‘position: topicalized Positional PP(P)’ into P-List

Example: *at the end of the road the museum is **there*** → resolve ‘there’ = position: at the end of the road

9. Temporal-conditional clause in previous unit

Preconditions: There-Anaphor is in unit(U),
unit(U-1) is a temporal-conditional clause

Actions : insert ‘position: unit(U-1)’ into P-List

Example: *when you see the university of Plymouth turn down the road **there*** → resolve ‘there’ = position: when you see the university of Plymouth

APPENDIX 6 TESTING THE ALGORITHM ON UNSEEN DATA

1) Museum – Boots

u3_GC_MC_1.wav okay from the university

u3_GC_MC_2.wav er you turn right [1] and **it** s on your left [2]

1. update DM: *origin*=museum(sg., BUILDING)
destination=boots(sg., SHOP_BUILDING)
task='go from museum to boots'(sg., EVENTUALITY)
start_road(sg., PATH)
university1(sg., BUILDING, [], [], 1.1)
you turn right → TURN-Action: create implicit path-entity
path(sg., PATH, [in(right_region)], [], 1.2)
2. Selectional Restriction: (9) inference from Verb-PP
3. Selectional Restriction: (1b) equating construction → type constraint: ENTITY (*A)
4. properties of anaphor: X(sg., ENTITY, [in(left_region)], [], 2.1)
5. **Final-Sentence-Rule**
6. properties of anaphor match with properties of destination
7. destination is not mentioned in unit(2/1) → resolve 'it' = *destination*=boots
correct

2) Museum – Safeway

u3_GC_MD_1.wav okay from the grand hotel you need to turn round [1]

u3_GC_MD_2.wav and go back down the road [2] and **it** s on the left [3]

1. update DM: *origin*=museum(sg., BUILDING)
destination=safeway(sg., SHOP_BUILDING)
task='go from museum to safeway'(sg., EVENTUALITY)
start_road(sg., PATH)
grand_hotel1(sg., HOTEL, [], [], 1.1)
road1(sg., PATH, [], [], 2.1)
2. Selectional Restriction: (9) inference from Verb-PP
3. Selectional Restriction: (1b) equating construction → type constraint: ENTITY (*A)
4. properties of anaphor: X(sg., ENTITY, [in(left_region)], [], 3.1)
5. **Final-Sentence-Rule**
6. properties of anaphor match with properties of destination
7. destination is not mentioned in unit(3/2/1) → resolve 'it' = *destination*=safeway
correct

3) Museum – University

u3_GC_MW_6.wav er forward and then left [4]

u3_GC_MW_7.wav erm then you go left round the roundabout [5]

u3_GC_MW_8.wav and then forward

u3_GC_MW_9.wav and then you re **there** [6]

1. **Here/There-Rule: (3) IV in previous unit**
2. type of preposition: 'round' → position: behind
3. construct P-List: {position: behind *the roundabout*} → resolve 'there' =

position: behind *the roundabout***incorrect** → FINAL SENTENCE-RULE

4)Museum – Post office

u3_GC_MZ_1.wav okay from the beginning you go forward [1] and then **it** s on your left about three quarters of the way down the road [2]

1. update DM: *origin*=museum(sg., BUILDING)
destination=post_office(sg., BUILDING)
task='go from museum to post_office'(sg., EVENTUALITY)
start_road(sg., PATH)
beginning1(sg., LOCATION, [],[],1.1)
2. Selectional Restriction: (9) inference from Verb-PP
3. Selectional Restriction: (1b) equating construction → type constraint: ENTITY (*A)
4. properties of anaphor: X(sg., ENTITY, [in(left_region)], [], 2.1)
5. **Final-Sentence-Rule**
6. properties of anaphor match with properties of destination
7. destination is not mentioned in unit(2/1) → resolve 'it' = *destination*=post_office
correct

5) Grand Hotel – Boots

u4_GC_EC_1.wav yes well go past the train station again to the university [1]

u4_GC_EC_2.wav past the university [2] **ellipsis**

a) u4_GC_EC_3.wav after you pass the university [3]you ll see pc world [4] you need to take **that right** [5]

b) u4_GC_EC_4.wav and boots is on **that road** [6]

- a) 1. update DM: ... university1(sg., BUILDING, [],[],2.1)
university1(sg. BUILDING, [],[], 3.1)
pc_world1(sg., SHOP_BUILDING, [],[],4.1)
2. **elliptic**: *that right* → 'the next road on your right hand side'
3. type constraint for anaphor → PATH
4. properties of anaphor: X(sg., PATH, [in(right_region)], [], 5.1)
5. **This-NP-Anaphor: (1) element_entity**
6. empty S-List at unit(5) → construct implicit S-List at unit(5)
7. empty implicit S-List at unit(5) → construct S-List at unit(4)
8. empty S-List at unit(4) → construct implicit S-List at unit(4)
9. empty implicit S-List at unit(4) → construct S-List at unit(3)
10. empty S-List at unit(3) → construct implicit S-List at unit(3)
11. empty implicit S-List at unit(3) → create new entity of same type as
element_entity **and** classify new entity as deictic referential to visual situation
correct
- b) 1. update DM: ... university1(sg., BUILDING, [],[],2.1)
university1(sg. BUILDING, [],[], 3.1)
pc_world1(sg., SHOP_BUILDING, [],[],4.1)
road1(sg., PATH, [in(right_region)],[], 5.1) → **deictic referential to visual situation**
boots1(sg., SHOP_BUILDING, [],[], 6.1)
2. type constraint for anaphor → PATH

3. properties of anaphor: X(sg., PATH, [],[],6.2)
4. **This-NP-Anaphor: (1) element_entity**
5. construct S-List:{road1} → resolve ‘that road’ = road1 **correct**

6) Grand Hotel – Tesco

u4_GC_EG_3.wav *this time* you ll be going past pizza hut which is on your left [2]
 u4_GC_EG_4.wav and tescos the second building from **there** [3]

1. **Here/There-Rule: (3) IV in previous unit**
2. type of preposition: ‘past’ → position: behind
3. construct P-List:{position: behind *pizza hut*} → resolve ‘there’ = position: behind *pizza hut* **incorrect** → FROM-RULE

7) Grand Hotel - Hospital

u4_GC_EH_3.wav you take the right after the car park [3]
 u4_GC_EH_4.wav and then another right again [4] → **ellipsis**
 u4_GC_EH_5.wav and you ll be moving towards the hospital on the end of **that road** [5]

1. update DM: ... *take the right* → ‘take the road on your right’ (elliptic TAKE)
 road1(sg., PATH, [in(right_region)], [],3.1)
 car_park1(sg., PLACE, [],[],3.2)
 then another right again → ‘take the next road on your right again’ (**Ellipse**)
 road2(sg., PATH, [in(right_region)], [],4.1)
 hospital1(sg., BUILDING, [],[], 5.1)
 end1(sg., REGION, [of(roadX)], [], 5.2)
2. type constraint for anaphor → PATH
3. properties of anaphor: X(sg., PATH, [],[], 5.3)
4. **This-NP-Anaphor: (1) element_entity**
5. construct S-List at unit(5):{road2} → resolve ‘that road’ = road2 **correct**

8) Grand Hotel – Post Office

u4_GC_EX_6.wav erm yeah exit the roundabout at the third exit [4]
 u4_GC_EX_7.wav move forward over the bridge [5] and you ll be able to see the post office from **there** [6]

1. **Here/There-Rule: (3) IV in previous unit**
2. type of preposition: ‘over’ → position: behind
3. construct P-List:{position: behind *the bridge*} → resolve ‘there’ = position: behind *the bridge* **correct** → NO FROM-RULE

9) Grand Hotel – Hospital

u5_GC_EH_1.wav er when you arrive at the car park [1] if you cross the car park [2] and turn right [3]
 u5_GC_EH_2.wav you turn right [3] → **repetition**
 u5_GC_EH_3.wav and you will find the hospital a short way along **that road** in front of you [4]

1. update DM:... car_park1(sg., PLACE, [],[],1.1)
 car_park1(sg., PLACE, [],[], 2.1)

1. **Here/There-Rule: (3) IV in previous unit**
2. type of preposition: 'past' → position: behind
3. construct P-List: {position: behind *the t-junction*} → resolve 'there' = position: behind *the t-junction* **correct**

13) Hospital – Boots

u8_GC_HC_4.wav er go forwards to the roundabout [4] and take the second turning off [5]
 u8_GC_HC_5.wav er take the first turning on your right [6]
 u8_GC_HC_6.wav and **it** s the first building on the left [7]

1. update DM: *origin*=hospital(sg., BUILDING)
 destination=boots(sg., SHOP-BUILDING)
 ... turning1(sg., JUNCTION, [in(right_region)], [first], 6.1)
2. Selectional Restrictions: (1) equation construction → type constraint: BUILDING (*A)
3. properties of anaphor: X(sg., BUILDING, [in(left_region)], [first], 7.1)
4. **Final-Sentence-Rule**
5. properties of anaphor match with properties of destination
6. destination is not mentioned in unit(7/6/5) → resolve 'it' = *destination*=boots
correct

14) Hospital – Grand Hotel

u8_GC_HE_2.wav er keep going till the end of the end of the road [2]
 u8_GC_HE_3.wav turn left [3]
 u8_GC_HE_4.wav and **it** s the first building on the left [4]

1. update DM: *origin*=hospital(sg., BUILDING)
 destination=grand_hotel(sg., HOTEL)
 ... end1(sg., REGION, [of(road1)], [], 2.1)
 road1(sg., PATH, [], [], 2.2)
 turn left → TURN-Action: create implicit path-entity
 path1(sg., PATH, [in(left_region)], [], 3.1)
2. Selectional Restrictions: (1) equation construction → type constraint: BUILDING (*A)
3. properties of anaphor: X(sg., BUILDING, [in(left_region)], [first], 4.1)
4. **Final-Sentence-Rule**
5. properties of anaphor match with properties of destination
6. destination is not mentioned in unit(4/3/2) → resolve 'it' = *destination*=grand_hotel
correct

15) Hospital – Tesco

u8_GC_HG_3.wav turn right [3] when you reach a crossroads [4]
 u8_GC_HG_4.wav and head down the crossroads [5] and **it** s the third building on your left [6]

1. update DM: *origin*=hospital(sg., BUILDING)
 destination=tesco(sg., SHOP_BUILDING)
 ... *turn left* → TURN-Action: create implicit path-entity

- path1(sg., PATH, [in(left_region)], [], 3.1)
 crossroads1(sg., JUNCTION, [], [], 4.1)
 crossroads1(sg., JUNCTION, [], [], 5.1)
2. Selectional Restrictions: (1) equation construction → type constraint: BUILDING (*A)
 3. properties of anaphor: X(sg., BUILDING, [in(left_region)], [third], 6.1)
 4. **Final-Sentence-Rule**
 5. properties of anaphor match with properties of destination
 6. destination is not mentioned in unit(4/3/2) → resolve ‘it’ = *destination=tesco*
correct

16) Hospital – University

u8_GC_HW_1.wav er yup from the grand hotel head towards the roundabout [1]

u8_GC_HW_2.wav take the second turning off the roundabout [2]

u8_GC_HW_3.wav and **it** s the first building on the left [3]

1. update DM: *origin=hospital*(sg., BUILDING)
destination=university(sg., BUILDING)
 ...
grand_hotel1(sg., HOTEL, [], [], 1.1)
roundabout1(sg., JUNCTION, [], [], 1.2)
turning1(sg., JUNCTION, [], [second], 2.1)
roundabout1(sg., JUNCTION, [], [], 2.2)
2. Selectional Restrictions: (1) equation construction → type constraint: BUILDING (*A)
3. properties of anaphor: X(sg., BUILDING, [in(left_region)], [first], 3.1)
4. **Final-Sentence-Rule**
5. properties of anaphor match with properties of destination
6. destination is not mentioned in unit(3/2/1) → resolve ‘it’ = *destination=university*
correct

17) Hospital – Queens Pub

u9_GC_HL_4.wav er then er take a sort of right turn by dixons [3]

a) u9_GC_HL_5.wav erm carry on up **there** [4] and then take a right [5]

b) u9_GC_HL_6.wav **that** should take you to the queens pub [6]

- a)
 1. **Here/There-Rule: (TV) in previous unit**
 2. construct P-List: {position: at a sort of right turn} → resolve ‘there’ = position: at a sort of right turn
incorrect → SPECIAL RULE FOR TAKE or SPECIAL CONDITIONS for spatially specified object of TV
- b)
 1. update DM: ... *turn1*(sg., PATH, [in(right_region)], [], 3.1)
take a right → ‘take the next road on your right’ (elliptic TAKE)
road1(sg., PATH, [in(right_region)], [], 5.1)
 2. Selectional Restrictions: (4) Movement with agent as object: b) verb of motion in previous unit → type constraint: EVENTUALITY (*I)
 3. properties of anaphor: X(sg. EVENTUALITY, [], [], 6.1)
 4. **(2) Anaphor is *I**
 5. construct A-List: {*take a right*} → resolve ‘that’ = *take a right*
correct → but ambiguous: 1) take a right, 2) whole instruction & 3) road1

2. Selectional Restrictions: (7) Spatial PP → type constraint: ENTITY (*A)
3. properties of anaphor: X(sg., ENTITY, [],[], 2.1)
4. **(1) Anaphor is *A**
5. construct S-List:{roundabout1} → resolve ‘it’ = roundabout1 **correct**

- b)**
1. update DM: ... roundabout1(sg., JUNCTION, [],[],1.1)
 roundabout1(sg., JUNCTION, [],[], 2.1)
 2. Selectional Restrictions: (1) equation construction → type constraint: PATH (*A)
 3. properties of anaphor: X(sg., PATH, [in(right_region)], [second], 3.1)
 4. **(1) Anaphor is *A**
 5. employ S-List → classify ‘it’ as **vaguecorrect**

22) Grand Hotel – Post Office

u11_GA_EX_2.wav start going around the roundabout [3] and take your first right [4]

u11_GA_EX_3.wav and continue straight

u11_GA_EX_4.wav go across the bridge [5]

a) u11_GA_EX_5.wav and **it** should be right in front of you to the right [6]

b) u11_GA_EX_6.wav and **that** s where the post office is [7]

- a)**
1. update DM:... *take your first right* → ‘take the first road on your right’ (ellip.TAKE)

road1(sg., PATH, [in(right_region)], [first], 4.1)

bridge1(sg., TRANSITION, [],[], 5.1)

2. Selectional Restriction: (9) inference from Verb-PP

3. Selectional Restriction: (1b) equating construction → type constraint: ENTITY (*A)

4. properties of anaphor: X(sg., ENTITY, [in(in_front_region), in(right_region)], [],6.1)

5. **(1) Anaphor is *A**

6. construct S-List:{bridge1} → resolve ‘it’ = bridge1 **incorrect**
DESTINATION not in final sentence, but at the end

- b)**
1. update DM:... *take your first right* → ‘take the first road on your right’ (ellip.TAKE)

road1(sg., PATH, [in(right_region)], [first], 4.1)

bridge1(sg., TRANSITION, [],[], 5.1)

bridge1(sg., TRANSITION, [],[],6.1)

2. Selectional Restrictions: → ambiguous

3. properties of anaphor: X(sg., ?, [],[],7.1)

4. **(4) DEM is ambiguous**

5. construct A-List:{*be right in front of you to the right*} → resolve ‘that’ = *be right*

in

front of you to the right **incorrect** SPECIAL WHERE-RULE

23) Grand Hotel – Post Office

u12_GA_EX_1.wav go along the road [1] and take the first right [2]

a) u12_GA_EX_2.wav carry all carry along all the way down **this road** [3]

b) u12_GA_EX_3.wav until you reach the end of **it** [4]

- a) 1. update DM: ... road1(sg., PATH, [],[], 1.1)
take the first right → ‘take the first road on your right’ (elliptic TAKE)
road2(sg., PATH, [in(right_region)], [first], 2.1)
2. type constraint for anaphor → PATH
3. properties of anaphor: X(sg., PATH, [],[], 3.1)
4. **This-NP-Anaphor: (1) element_entity**
5. construct S-List at unit(3):{road2} → resolve ‘this road’ = road2 **correct**
- b) 1. update DM: ... road1(sg., PATH, [],[],1.1)
take the first right → ‘take the first road on your right’ (elliptic TAKE)
road2(sg., PATH, [in(right_region)], [first], 2.1)
road2(sg., PATH, [],[], 3.1)
end1(sg., REGION, [of(X)],[],4.1)
2. Selectional Restriction: nothing for ‘of’-preposition
3. **Complex NP-Rule: (1) complex NP**
4. resolve ‘it’ = road2 **correct** OF-Rule

24) Hospital – Boots

u15_GA_HC_1.wav okay go to the end of the road [1] and turn left [2]

a) u15_GA_HC_2.wav and erm and then carry on down **that road** [3]

u15_GA_HC_3.wav and then turn take your second left [4] where the trees are on the corner [5]

b) &c) u15_GA_HC_4.wav and then go over **that bridge there** [6]

u15_GA_HC_5.wav carry on straight until you meet the roundabout [7]

u15_GA_HC_6.wav take your first exit on the right at the roundabout [8]

d) u15_GA_HC_7.wav and erm and then walk down **that road** with pc world on your right [9]

e) u15_GA_HC_8.wav you take **that road** on the right [10] and then you reach boots which is on your left [11]

- a) 1. update DM: ... end1(sg., REGION, [of(road1)],[],1.1)
road1[sg., PATH, [],[], 1.2)
turn left → TURN-Action: create implicit path-entity in DM
path1(sg., PATH, [in(left_region)],[],2.1)
2. type constraint for anaphor → PATH
3. properties of anaphor: X(sg., PATH, [],[], 3.1)
4. **This-NP-Anaphor: (1) element_entity**
5. construct S-List at unit(3):{path1} → resolve ‘that road’ = path1 **correct**
- b) 1. update DM: ... end1(sg., REGION, [of(road1)],[],1.1)
road1[sg., PATH, [],[], 1.2)
turn left → TURN-Action: create implicit path-entity in DM
path1(sg., PATH, [in(left_region)],[],2.1)
path1=road2(sg., PATH, [],[], 3.1)
take your second left → ‘take the second road on your left’ (elliptic TAKE)
road3(sg., PATH, [in(left_region)], [second], 4.1)
trees1(pl., PLANT, [],[], 5.1)
corner1(sg., SHAPE, [],[], 5.2)
2. type constraint for anaphor → TRANSITION

26) Hospital – University

a) u15_GA_HW_2.wav and then you take the next left [4] and walk down **that road** [5] as if you re doubling back on yourself [6]

u15_GA_HW_3.wav and then you reach the bend in the road [7] where safeways is [8]

b) u15_GA_HW_4.wav continue down **that bend** [9]

c) & d) u15_GA_HW_5.wav take the next take **that turn right there** [10] until you reach the roundabout [11]

e) u15_GA_HW_6.wav and then you take the second left off the roundabout [12] and the university is at **that exit** [13]

- a)
1. update DM: ... *take the next left* → 'take the next road on your left' (elliptic TAKE)
road1(sg., PATH, [in(left_region)], [next], 4.1)
 2. type constraint for anaphor → PATH
 3. properties of anaphor: X(sg., PATH, [], [], 5.1)
 4. **This-NP-Anaphor: (1) element_entity**
 5. construct S-List at unit(5): {road1} → resolve 'that road' = road1 **correct**
- b)
1. update DM: ... bend1(sg., PATH, [], [], 7.1)
road1(sg., PATH, [], [], 7.2)
safeway1(sg., SHOP_BUILDING, [], [], 8.1)
 2. type constraint of anaphor → PATH
 3. properties of anaphor: X(sg., PATH, [], [], 9.1)
 4. **This-NP-Anaphor: (2) normal**
 5. empty S-List at unit(9) → construct S-List at unit(8): {bend1, road1}
 6. resolve 'that bend' = bend1 **correct**
- c)
1. update DM: ... bend1(sg., PATH, [], [], 7.1)
road1(sg., PATH, [], [], 7.2)
safeway1(sg., SHOP_BUILDING, [], [], 8.1)
bend1(sg., PATH, [], [], 9.1)
 2. type constraint for anaphor → PATH
 3. properties of anaphor: X(sg. PATH, [], [], 10.1)
 4. **This-NP-Anaphor: (2) normal**
 5. construct S-List at unit(10): {bend1} → resolve 'that turn' = bend1 **incorrect**
tricky
- d)
1. **Here/There-Rule: (3) IV in previous unit**
 2. type of preposition: 'down' → position: along
 3. construct P-List: {position: along *that bend*}
→ resolve 'there' = position: along *that bend* **incorrect**
→ VISUAL DEICTIC (vague)
- e)
1. update DM: ... roundabout1(sg., JUNCTION, [], [], 11.1)
take the second left → 'take the second road on your left' (elliptic TAKE)
road2(sg., PATH, [in(left_region)], [second], 12.1)

roundabout1(sg., JUNCTION, [], [], 12.2)
university1(sg., BUILDING, [], [], 13.1)
 2. type constraint for anaphor → PATH

3. properties of anaphor: X(sg., PATH, [],[], 13.2)
4. **This-NP-Anaphor: (1) element_entity**
5. construct S-List at unit(13):{road2} → resolve ‘that exit’ = road2 **correct**

27) Museum – Safeway

u17_GB_MD_1.wav the destination is safeway [1] so we continue straight down **this road** [2]

1. update DM: *origin*=museum(sg., BUILDING)
destination=safeway(sg., SHOP_BUILDING)
task=‘go from museum to safeway’(sg., EVENTUALITY)
start_road(sg., PATH)
destination1(sg., LOCATION, [],[],1.1)
destination=safeway(sg., SHOP_BUILDING, [],[], 1.2)
2. type constraint for anaphor → PATH
3. properties of anaphor: X(sg., PATH, [],[],2.1)
4. **First-Sentence-Rule: (2) Start-Road**
5. empty S-List → resolve ‘this road’ = *start_road*(sg., PATH) in DM **correct**

28) Museum – University

u17_GB_MW_1.wav go straight down **this road** past the post office as before

1. update DM: *origin*=museum(sg., BUILDING)
destination=hospital(sg., BUILDING)
task=‘go from museum to hospital’(sg., EVENTUALITY)
start_road(sg., PATH)
2. type constraint for anaphor → PATH
3. properties of anaphor: X(sg., PATH, [],[],1.1)
4. **First-Sentence-Rule: (2) Start-Road**
5. empty S-List → resolve ‘this road’ = *start_road*(sg., PATH) in DM **correct**

29) Museum – Boots

u17_GB_MC_6.wav go round the roundabout [7] and you will take the third exit off [8]

u17_GB_MC_7.wav the third exit will be for plymouth university [9]

a) u17_GB_MC_8.wav take **this third exit** [10]

u17_GB_MC_9.wav just past plymouth university on the right hand side you will find boots [11]

b) u17_GB_MC_10.wav **this** is your destination [12]

- a) 1. update DM: ... roundabout1(sg., JUNCTION, [],[], 7.1)
exit1(sg., PATH, [],[third],8.1)
exit1(sg., PATH, [],[third], 9.1)
Plymouth_university(sg., UNIVERSITY, [],[],9.2)
2. type constraint for anaphor → PATH
3. properties of anaphor: X(sg., PATH, [],[third], 10.1)
4. **This-NP-Anaphor: (1) element_entity**
5. construct S-List at unit(10):{exit1} → resolve ‘this third exit’ = exit1
correct

- b) 1. update DM: ... exit1(sg., PATH, [], [third], 10.1)
Plymouth_university(sg., UNIVERSITY, [in(right_region)], [], 11.1)
boots1(sg., SHOP_BUILDING, [], [], 11.2)
2. Selectional Restrictions: (1) equation construction → type constraint: LOCATION
- (*I)
3. (2) **Anaphor is *I**
4. construct A-List: {find boots} → resolve 'this' = find boots **incorrect**
→ CHANGE in ONTOLOGY for destination

30) Grand Hotel – Hospital

u19_GB_EH_1.wav okay you want to go to the hospital [1]

a) u19_GB_EH_2.wav and to get **there** [2] you want to do the first right down the road [3]
from where you are [4] to go past derry s and the grand hotel [5]

b) u19_GB_EH_3.wav and then you want to work your way round the bendy road **there** [6]

...

u19_GB_EH_8.wav er you come to another junction [11] and the first on the right

c) u19_GB_EH_9.wav you want to er turn into [12] and then go all the way down **that road**

u19_GB_EH_10.wav past the car park again from the other side [13]

- a) 1. **Here/There-Rule: (3) IV in previous unit**
2. type of preposition: 'to' → position: at
3. construct P-List: {position: at the hospital} → resolve 'there' = position: at the hospital **incorrect** → VERB & PREPOSITION do not match
- b) 1. **Here/There-Rule: (3) IV in previous unit**
2. type of preposition: 'past' → position: behind
3. construct P-List: {position: behind derry's and the grand hotel} → resolve 'there' =
position: behind derry's and the grand hotel **correct**
- c) 1. update DM: ... junction1(sg., JUNCTION, [], [], 11.1)
turn into the first on the right → 'turn into the first road on the right' (elliptic TURN
road2(sg., PATH, [in(right_region)], [first], 12.1)
2. type constraint for anaphor → PATH
3. properties of anaphor: X(sg., PATH, [], [], 13.1)
4. **This-NP-Anaphor: (1) element_entity**
5. construct S-List at unit(13): {road2} → resolve 'that road' = road2 **correct**

31) Grand Hotel – University

u19_GB_EW_3.wav and then you want to take the second exit on the left again [3]

a) & b) u19_GB_EW_4.wav and if you can see **it** [4] you should be right **there** [5]

u19_GB_EW_5.wav just go down the road [6]

c) u19_GB_EW_6.wav and **it** will be on the left big grey building [7]

- a) 1. update DM: ... exit1(sg., PATH, [in(left_region)], [second], 3.1)
2. Selectional Restrictions: (3) Argument of Verb → type constraint: ENTITY (*I)
3. properties of anaphor: X(sg., ENTITY, [], [], 4.1)
4. (1) **Anaphor is *I**
5. construct S-List: {exit1} → resolve 'it' = exit1 **incorrect** → tricky,

turn right → TURN-Action: create implicit path-entity
 path1(sg., PATH, [in(right_region)],[],7.1)
 street1(sg., PATH, [],[main], 7.2)

2. Selectional Restrictions: (3) Argument of Verb → type constraint: PATH or roundabout (*A)
3. properties of anaphor:
 - a) X(sg., PATH, [],[],8.1)
 - b) X(sg., roundabout, [],[],8.1)
4. **(1) Anaphor is *A**
5. construct S-List: {path1, street1} → resolve 'it' = path1 **correct**
 BUT TRICKY, when introduce new path-entity

- b)**
1. update DM: ... path1(sg., PATH, [],[], 8.1)
 end1(sg., REGION, [],[],8.2)
 building1(sg., BUILDING, [in(right_region)], [big, tall],9.1)
 2. Selectional Restrictions:(1)equation construction
 → type constraint: BUILDING & MUSEUM (*A)
 3. properties of anaphor: X(sg., BUILDING & MUSEUM, [],[],10.1)
 4. **(1) Anaphor is *A**
 5. construct S-List: {building1} → resolve 'that' = building1 **correct**

35) Hospital – University

u23_GB_HW_2.wav you recall on a previous trip to the grand hotel we passed over a pond over a bridge [2]

a) u23_GB_HW_3.wav if we start at **that bridge** [3]

u23_GB_HW_4.wav go forward across the crossroads [4]

u23_GB_HW_5.wav onto a roundabout [5] → **ellipsis**

u23_GB_HW_6.wav um entering the roundabout [6] plymouth university will be the second exit [7]

b) u23_GB_HW_7.wav and **it** s a three storey building on your right [8]

- a)**
1. update DM: ... trip1(sg., JOURNEY, [],[previous],2.1)
 grand_hotel(sg., HOTEL, [],[],2.2)
 pond1(sg., LAKE, [],[],2.3)
 bridge1(sg., TRANSITION,[],[],2.4)
 2. type constraint for anaphor → TRANSITION
 3. properties of anaphor: X(sg., TRANSITION, [],[], 3.1)
 4. **This-NP-Anaphor: (2) normal**
 5. construct S-List at unit(3): {bridge1} → resolve 'that bride' = bridge1 **correct**

- b)**
1. update DM: ... *destination*=university(sg., BUILDING)
 roundabout1(sg., JUNCTION, [],[], 5.1)
 Plymouth_university1(sg., UNIVERSITY, [],[], 7.1)
 exit1(sg., PATH, [],[second], 7.2)
 2. Selectional Restrictions:(1)equation construction → type constraint: BUILDING (*A)
 3. properties of anaphor: X(sg., BUILDING, [in(right_region)], [three storey], 8.1)
 4. no **Final Sentence Rule**
 5. **(1) Anaphor is *A**

6. construct S-List:{Plymouth_university1} → resolve 'it' = Plymouth_university1
correct

36) Hospital – Tesco

u24_GB_HG_1.wav forward to the white dotted line [1]

u24_GB_HG_2.wav turn left [2]

u24_GB_HG_3.wav forward to the crossroads [3]

u24_GB_HG_4.wav turn left [4]

u24_GB_HG_5.wav forward across the crossroads to the roundabout [5]

u24_GB_HG_6.wav go two hundred and seventy degrees around the roundabout clockwise [6]

u24_GB_HG_7.wav exit at er **that angle** turning left [7]

1. update DM: ... *turn left* → TURN-Action: create implicit path-entity
path2(sg., PATH, [in(left_region)],[],4.1)
crossroads2(sg., JUNCTION, [],[],5.1)
roundabout1(sg., JUNCTION, [],[], 5.2)
degrees1(pl., UNIT_OF_MEASUREMENT, [],[270], 6.1)
roundabout1(sg., JUNCTION, [],[], 6.2)
2. type constraint for anaphor → SHAPE
3. properties of anaphor: X(sg., SHAPE, [],[],7.1)
4. **This-NP-Anaphor: normal**
5. empty S-List at unit(7) → construct S-List at unit(6)
6. empty S-List at unit(6) → construct S-List at unit(5)
7. empty S-List at unit(5) → create new angle_entity in DM:
angle1(sg., SHAPE, [],[],7.1) & mark it as **deictic referential to visual situation**
correct

APPENDIX 7 TESTING THE ALGORITHM ON NEW INSTRUCTIONS

ROUTE INSTRUCTIONS from Edinburgh

1.

a) From my flat to Waverley

*Turn left outside the front door, then left again at the end of the street, **that's** Drummond Street. (RIAR: that = the street; **incorrect** / GS: that= turn-path-entity; **correct**) Go along **it** to the end and turn right onto the main road. (RIAR: it=Drummond Street; **correct**) Then you continue down the road until **it** crosses the Royal Mile.(RIAR: it = the road; **correct**) Turn left onto the Royal Mile then first right down Cockburn Street (**it's** spelt C O C K burn). (RIAR: it = first road on your right; **incorrect** / GS: it = Cockburn; **correct**) **It** goes round a couple of corners and there's a roundabout at the bottom, (RIAR: it = Cockburn Street; **correct**) go straight on and the train station is just on the right **there**, (RIAR: there = vague; **correct**) you can see the entrance from the bottom of Cockburn Street.*

b) Main Library to Waverley

*Come out of the door and down the steps and turn left past the garden part of George Square. Then at the corner of the garden turn right up the cobbles. At the top of **that side** of the square you can turn left in between two buildings (RIAR: that side = visual discourse deictic; **incorrect** / GS: that side = element-entity of square, **correct**) and **that path** (cars can't go down there) brings you out on Middle Meadow Walk (RIAR: 1) that path = implicit turn-path; **correct** / 2) there = position: in between two buildings; **correct**). Turn right and go up **it** to the top. (RIAR: it = implicit turn-entity; **correct**) Cross the road at the crossing and go straight on down Forrest Road. You come out on George IV Bridge with Greyfriar's Bobby pub on the left. Keep going down George IV Bridge, **it** crosses the Royal Mile (RIAR: it = George IV Bridge; **correct**) and then bends to the left. There's the Bank of Scotland Head Office on the right **there**. (RIAR: there = position: at the left; **incorrect** / GS: there = after George IV Bridge bended to the left; **correct**) Just after **that** (RIAR: that = Bank of Scotland Head Office; **correct**) turn right down Market Street and then left at the roundabout at the bottom and the train station's on the right.*

2.

a) From my flat to Waverley

*Go out the front door – turn left. At the corner left again. Go straight 2 blocks until you reach a busy street (South Bridge). Turn right. *Go straight past the Royal Mile, then downhill until you come to a large intersection where you can't go straight anymore. Turn left and in about ½ block (past the Balmoral Hotel) you'll see a sign for Waverly on the left.*

b) Main Library to Waverley

*Go out the front door of the library – turn right. Take 1st possible left (follow the edge of the park) – go straight for about 2 blocks – turn right after parking lot on right. Go 1 short block, cross the street, go another short block and the road divides around a small park. Continue straight past the park (go on either side). Then turn left on the busy street (South Bridge) and continue as above *.*

3.

a) From my flat to Waverly Station

*If you go right to the top of the street and er turn to your left then you go right down to the bottom of the road, you'll go down a hill right next to the meadows just keep on going till you can't really go any further, don't turn off **that road** or anything (RIAR: that road = the road (in pre-previous utterance); **correct**) – kind of half way down you'll sort of meet a sort of crossroads, with lots of like traffic lights, don't TURN left, like to your left but just bear left, like the road you're on sort of bends to the left, and **that's** Lothian road (RIAR: that = the road; **correct**) and you kinna just go past loads of shops on the way down.*

*When you get to the end of the road, you'll see right in front of you like erm across the main road you'll see erm 'Ryan's bar' **it** has like a big blue canvas top over (RIAR: it = Ryan's bar; **correct**) like **this glass fronted bit**, (RIAR: this glass fronted bit = a big blue canvas top, **correct**) erm and **that's** like the start of Prince's street.(RIAR: that = has like a big blue canvas top over like this glass fronted bit; **incorrect** / GS: that = vague, **correct**) Waverly's actually right down the other end so if you just turn on to your right and just keep walking right to the other end. You'll go past, like there's all Prince's street gardens down the side and stuff – on your right and then like you'll hit there's erm there's like a big stone building with erm pillars and stuff, **it's** covered in scaffolding at the moment (RIAR: it = a big stone building with pillars and stuff; **correct**) and stuff so ya can't miss **it** (RIAR: it = the big stone building with pillars and stuff; **correct**) (but **it's** the national art gallery (RIAR: it = the big stone building with pillars and stuff; **correct**) so if you go right past **that** (RIAR: that = the big stone building with pillars and stuff; **correct**) and you'll see a right big black sort of stone thing **it's** like a monument (RIAR: it = a right big black sort of stone thing; **correct**) and **it's** right high (RIAR: it = the right big black sort of stone thing; **correct**), **that's** the Scott monument (RIAR: it = the right big black sort of stone thing; **correct**) and erm just after **that** (RIAR: it = the right big black sort of stone thing; **correct**) there's erm Waverly bridge so if you turn right down the bridge erm Waverly station's actually on the left hand side erm it like goes down the road to **it** off the bridge (RIAR: it = Waverly station; **correct**) and there always loads of taxis and stuff coming out so you just walk down **there** (RIAR: there = position: at loads of taxis and stuff coming out; **incorrect** / GS: there = vague; **correct**) and **that's** it!*

b) From the main Edinburgh library to Waverly station

*Right you erm you come out the er library and turn to your erm left and you'll hit a like erm like a crossroads where the bridge hits the Royal Mile you just want to go, you just want to keep goin' right over the royal mile and then like down the other side of **it** (RIAR: it = the royal mile; **correct**) the road like slopes down a bit and in front of you you should see like er kinna posh old stone building, **it's** actually the Bank of Scotland. (RIAR: it = posh old stone building; **correct**) If you cross over the road that your on so you're standing in front of the building and then turn left - you're actually you're actually facing the building and then go to your left and then just walk down the hill a bit until you meet a like a erm road goin off to your right **it's** not that far down (RIAR: it = a road; **correct**) **it's** just like a little way (RIAR: it = the road; **correct**) and you wanna you wanna go down **that road**,(RIAR: that road = the road; **correct**) and **it** kinna slopes down as well (RIAR: it = the road; **correct**) and you'll hit erm a a mini roundabout thing and you can go like erm like right left straight on or right and you wanna go to your left and **that's** like you onto Waverly bridge (RIAR: that = go to your left; **correct**) the station's on the opposite side of the road so erm yeah and I think I'm not that sure but I think **it's** like the first like turning off erm off the bridge down the right side (RIAR: it = the station; **incorrect** / GS: it = vague; **correct**) I mean there's loads of like taxis*

*an stuff coming up from the station so you just kinna go down **there** (RIAR: there = position: from the station; **incorrect** / GS: there = to the station; **correct**) and **that's** you (???)*

4.

From Buccleuch Place to Waverly

*Turn out of Buccleuch Place – left onto the main road, no, actually, don't do **that!** (RIAR: that = turn left onto the main road; **correct**) Go down Buccleuch Place to the very end where you see a park – not **that** either! (RIAR: that = see a park; **incorrect** / GS: that = go down BP to the very end where you see a park; **correct**)*

*Half way along Buccleuch Place there are some steps – go up the steps and go straight along – you'll see a car park on your right – carry straight on – walk along the edge of Bristo square – you'll see a Northwest bank on your right and people skateboarding on your left – when you reach the main road, cross over and keep walking in the direction you've been going – down either Bristo Place or Forrest Hill onto a road called George IV Bridge. Go right along **this road** to the traffic lights (RIAR: this road = a road; **correct**) – cross over – (you'll see a pub on the left hand corner of the cross-roads) – carry on in the same direction a little till you see a big building (a bank, I think) in front of you – turn right onto the road in front of **it** (RIAR: it = a big building; **correct**) and you'll see some steps going down the hill. They're called the new steps. Go down them. Turn right at the bottom – and then left at the mini-roundabout – Waverly station is on your right.*

5.

From Buccleuch Place to Waverly

*Turn right out of **this building** (RIAR: this building = origin=lab_in_BP; **correct**). Turn left at the end of the road. Follow **this road** past a church on the right (RIAR: this road = the road; **correct**), through two sets of traffic lights, with a large car park on the left between them.*

*Follow the road as **it** curves to the left (RIAR: it = the road; **correct**) and straightens out again, until your get to a crossroads with a set of traffic lights, with Doctors pub on the corner across the road to your right. Take the road to the right, and follow **it** to an intersection of three roads (RIAR: it = the road; **correct**). Turn left down a wide road that goes slightly uphill.*

Turn right at its end. Turn left down a steep road that curves down to the left just before a large crossroads. Go straight across the roundabout at its bottom, and then take the first right.

6.

From Buccleuch Place to Waverly

*Go downstairs and turn right out of the door. Cross over the road at the lights and go through the tunnel on your right. Go past the Indian restaurant and turn left the music shop onto Clerk Street. Keep going straight on. You'll pass **this big building** made of glass on your left (RIAR: this big building = the music shop; **incorrect** / GS: this big building = visual discourse deictic; **correct**). Go across South Bridge (which doesn't look like a bridge). Keep on going, over North Bridge. Turn left at the Balmoval onto Princess Street and the station is just down some steps on your left.*

Personal Pronoun: 21

	Correct	Incorrect	Total
PRO (destination)	/	/	/
PRO (origin)	/	/	/
PRO (initial_road)	/	/	/
PRO (task)	/	/	/
PRO (IND)	19 (95%)	1 (5%)	20 (95%)
PRO (AO)	/	/	/
PRO (VAGUE)	/	1 (100%)	1 (5%)
PRO (Complex-NP)	/	/	/
PRO – TOTAL	19 (90%)	2 (10%)	21

Demonstrative Pronoun: 10

	Correct	Incorrect	Total
DEM (origin)	/	/	/
DEM (initial_road)	/	/	/
DEM (task)	/	/	/
DEM (IND)	5 (83%)	1 (17%)	6 (60%)
DEM (AO)	1 (33%)	2 (67%)	3 (30%)
DEM (VAGUE)	/	1 (100%)	1 (10%)
DEM – TOTAL	6 (60%)	4 (40%)	10

This-NP-Anaphor: 9

	Correct	Incorrect	Total
This-NP (initial_road)	/	/	/
This-NP (origin)	1 (100%)	/	1 (11%)
This-NP (IND)	4 (100%)	/	4 (45%)
This-NP (TAKE-IND)	/	/	/
This-NP (IND) total	4 (100%)	/	4 (45%)
This-NP (IMPLICIT)	/	1 (100%)	1 (11%)
This-NP (TURN-IND)	2 (100%)	/	2 (22%)
This-NP (IMPLICIT) total	2 (67%)	1 (33%)	3 (33%)
This-NP (VISUAL)	/	1 (100%)	1 (11%)
This-NP - TOTAL	7 (78%)	2 (22%)	9

Here/There-Anaphor: 5

	Correct	Incorrect	Total
There	1 (50%)	1(50%)	2 (40%)
There (vague)	1 (33%)	2 (67%)	3 (60%)
Total	2 (40%)	3 (60%)	5