NON-SENTENTIAL UTTERANCES IN DIALOGUE: CLASSIFICATION, RESOLUTION AND USE

Raquel Fernández Rovira

A thesis submitted to the University of London
for the degree of Doctor of Philosophy

2006

Department of Computer Science
King's College London

— I would prefer not to.
Herman Melville, *Bartleby*

Abstract

This thesis combines corpus work with symbolic and statistical techniques to offer an account of non-sentential utterances in dialogue. These are utterances that do not have the form of a full sentence according to most traditional grammars, but that nevertheless convey a complete sentential meaning. The approach taken analyses non-sentential utterances as radical context-dependent expressions, and formalises them as functions from the current information state to the type of the next information state. The thesis is grounded in a corpus study that provides evidence about the occurrence of non-sentential utterances in conversation. This is used to put forward a data-driven taxonomy of classes of non-sentential utterances, which are then paired with a formal specification of their resolution requirements formulated in Type Theory with Records. In order to identify the right class, which will determine the appropriate resolution procedure, machine learning methods are employed, which can be used to boost the automatic processing of dialogue. The thesis also proposes a hierarchy of models for dialogue protocols, defined in terms of abstract machine models, that is based on a variety of structural features of dialogue that are also related to properties of non-sentential utterances. This suggest a link between dialogue dynamics on the one hand, and formal language theory and the theory of computation in the other.

Acknowledgements

Writing up this thesis has been a long and painful process—there is no way around it. I recall the whole of my years as a PhD student however as a very enjoyable period, during which I learned tons of stuff and met many interesting and special people. I should first go one step back to Barcelona though and thank Sergi Balari, who introduced me to computational linguistics in the first place, and who gave me all his support to go to London to start an adventure that began in 2001 and has since then been going on. Moving to London was a big turning point in my life. There I met Jonathan Ginzburg and decided to orient my research towards the study of dialogue. Jonathan has been an excellent supervisor, who gave me the freedom to go *from flower to flower* during my early years as a PhD student, and then helped me to make sense of all that mess to put together this PhD thesis. Thanks Jonathan, for your support, your friendliness, and your inspiring ideas.

During the four years and a half I was at King's College London, I had the chance to work and make friends with many nice people. I'd like to thank the guys in the NLP group: Christian Ebert, Leif A. Nielsen, Zoran Macura, and especially Matthew Purver, with whom I had many useful conversations, some of which ended up in joint papers that had an important influence on the work presented in this thesis. I am also very glad to have been able to work with Shalom Lappin, who played a strong role in some of my research topics. While at King's, I had the chance to attend quite a few conferences and workshops, which allowed me to meet people like Staffan Larsson, David Schlangen, and Stina Ericsson, who were always ready to combine the discussion of interesting stuff with some fun. I would also like to thank Robin Cooper for providing many of the ideas on which this work is based, as well as Wilfried Meyer Viol and Stephen Pulman for agreeing to be my examiners.

This thesis has been finished in Berlin. Thanks to Christian for welcoming me in the warmest possible way when I arrived there. I am also very grateful to Manfred Stede and David Schlangen, who took the risk of hiring me as a "postdoc" in Potsdam University when I was still writing up this thesis—thanks for your patience.

Finally, a huge thanks goes to Ulle Endriss, who has helped me in all possible forms all the way from beginning to end—and believe me if I say that towards the end it was not fun at all. During this period I have also been very lucky to have the support of my Good Old Friends from Barcelona. Thanks for not forgetting me. To you all, I promise to be a better company from now on.

Contents

1 Introduction

In the last twenty-five years, formal semantics has experienced a journey from classic approaches whose object of study was the isolated sentence (Montague 1974), to a broader dynamic view of meaning, first concerned with the analysis of discourse (Heim 1982, Groenendijk and Stokhof 1991, Kamp and Reyle 1993) and, more recently, also applied to the study of dialogue (e.g. Piwek 1998, Asher and Lascarides 2003, Ginzburg forthcoming). It is well known that through this process—which has constantly shaken the border between semantics and pragmatics—context dependence and context change have become central notions in the formal study of meaning. Besides the central role of context, the shift towards dialogue as the most natural form of language use brings in some idiosyncratic features that are typically absent from monologue settings. One of these is the pervasive presence of *non-sentential utterances*—like those in bold face in the dialogue below—which are primarily a dialogue phenomenon.

(1) A: How long has this man been living in the churchyard?
B: ***Oh, four or five years.***
A: ***Four or five years?***
B: ***Yes, yes.***
C: I've been there four, and he was there a year before, so it's five years.
A: ***Five years.***
B: There you are, you see.
A: ***Yes.***
[BNC: KRL 2892–2990]

The aim of this thesis is to investigate and analyse non-sentential utterances in dialogue, in order to achieve a better understanding of the forms they can take and the meanings they can convey, while shedding some light on the interplay between grammar and the dynamics of dialogue context. In the remainder of this introductory chapter I shall exemplify the main issues raised by non-sentential utterances and outline the approach taken in this thesis. I conclude with a systematic overview of the thesis, briefly summarising the content of each chapter.

1.1 Non-Sentential Utterances

What are non-sentential utterances? In a broad sense, non-sentential utterances—or NSUs, as I shall be calling them in the remainder of this thesis—are utterances that do not have the form of a full sentence according to most traditional grammars, but that nevertheless convey a complete sentential meaning, usually a proposition or a question. The following are two prototypical examples taken from the transcripts of the British National Corpus (Burnard 2000):

(2) a. A: Who wants Beethoven music?
B: ***Richard and James.***
[BNC: KB8 1024–1025]

b. A: It's Ruth's birthday.
B: ***When?***
[BNC: KBW 13116–13117][1]

One of the most important issues in the analysis of NSUs concerns their resolution, i.e. the recovery of a full clausal meaning from a form which is standardly considered non-clausal. In the first of the examples above, the NSU in bold face is a typical "short answer", which despite having the form of a simple NP would most likely be understood as conveying the proposition *"Richard and James want Beethoven music"*. The NSU in (2b) is an example of what has been called "sluice". Again, despite being realised by a bare *wh*-phrase, the meaning conveyed by the NSU could be paraphrased as the question *"When is Ruth's birthday?"*. Thus, to a large extent, the problem of resolving NSUs amounts to bridging the gap between their non-clausal form and their sentential interpretation.

This makes NSUs particularly interesting in two respects. First, as a challenge for standing theories of grammar, which should take positions with respect to the grammatical status they ascribe to these constructions—should they be considered proper sentences per se, or should they be regarded as *deviations* of more conventional sentences? Second, as the form of NSUs typically is that of a word or phrase with no clausal interpretation, the resolution process is bound to make critical use of contextual information. The intrinsic context-dependence of NSUs then serves to inform theories of context by providing evidence as to what elements these theories should contain. For instance, while most theories of context and context change tend to be formulated exclusively

[1]This notation indicates the name of the file and the sentence numbers in the British National Corpus.

in semantic terms, the constraints governing the well-formedness of some NSUs indicate that context must contain hybrid information. In (3a) and (3b) below, we can see that the grammatical case of the NSU needs to match that of the contextual *wh*-phrase (genitive in the first example and nominative in the second). While (3c) shows that a clarification NSU that intends to convey the question *"Who is Leo?"* must be identical in form to the constituent that is being clarified. A phonologically distinct clarification NSU like *"Your neighbour?"* is also well-formed, but it conveys a different question—roughly *"Is Leo your neighbour?"*.

(3) a. A: Whose bike did you ride?
B: ***Leo's/ #Leo.***

b. A: Who owes the bike you rode?
B: *#**Leo's/ Leo.***

c. A: Leo lent me his bike.
B: ***Leo? / #Your neighbour?***
[Intended meaning: *"Who is Leo?"*]

Thus, a proper analysis of NSUs must be rooted in a theory of context that is rich enough to account for both their semantic interpretation and their structural dependencies.

NSUs are not only a challenging problem for linguistic theories and for theories of context, but also for implemented systems, which in order to handle natural human-computer interactions or to summarise dialogue data need to be able to process and understand NSUs. However, due to their concise form and their highly context-dependent meaning, NSUs are often potentially ambiguous. For instance, B's response in (4a) is readily understood as a short answer. In (4b), on the other hand, the NSU would most likely be interpreted as some sort of correction (or a *helpful rejection*, as I shall call these NSUs later on in the thesis); while the NSU in (4c), depending on whether it is uttered with raising intonation or not, would play the role of either a clarification question or an acknowledgement.

(4) a. A: When are they going to open the new main station?
B: ***Tomorrow***

b. A: They are going to open the station today.
B: ***Tomorrow***

c. A: They are going to open the station tomorrow.
B: ***Tomorrow***

Distinguishing between different kinds of NSUs is not an easy task for a system. Consequently, in order to resolve NSUs appropriately, systems need to be equipped in the first place with the capability of identifying the intended kind of NSU.

1.2 Approach and Contributions

The approach taken in this thesis combines experimental, symbolic and statistical techniques to offer an account of NSUs in dialogue that is empirically founded, theoretically sound, and usable in computational processing.

The empirical work consists of a corpus study that provides evidence about naturally-occurring NSUs. This confirms that NSUs are an important phenomenon in dialogue, making up around 9% of all utterances. The study serves to put forward a data-driven taxonomy of NSU classes that offers a comprehensive inventory of the kinds of NSUs that can be found in conversation. This experimental data is then used in conjunction with machine learning techniques to train an automatic classification model that identifies the right NSU class. The stochastic techniques I employ are well-known, but their application to the automatic classification of NSUs is an original contribution of this thesis. As the NSU class determines the appropriate resolution procedure, the classifier can be used to boost the automatic processing of NSUs.

The theoretical part of this thesis is concerned with the resolution of NSUs. My approach to this issue derives from the one proposed by Ginzburg and Sag (2001), which in turn is based on the theory of context developed by Jonathan Ginzburg (1996, 1999, forthcoming). Ginzburg and Sag (2001) take NSUs to be first-class grammatical constructions whose resolution is achieved by combining the contribution of the NSU phrase with contextual information—concretely, with the current *question under discussion* or QUD, which roughly corresponds to the current conversational topic.

The simplest way of exemplifying this strategy is perhaps to consider a direct short answer to an explicit *wh*-question, like the one shown in (5a).

(5) a. A: Who's making the decisions?
B: ***The fund manager.***
[BNC: JK7 119–120]

b. QUD: $?(x).Make\_decision(x)$
Phrasal content of NSU: fm
Resolution: $?(x).Make\_decision(x)(fm) \equiv Make\_decision(fm)$

In this dialogue, the current QUD corresponds to the *wh*-question "*Who's making the decisions?*". Assuming a representation of questions as lambda abstracts, the resolution of the short answer simply amounts to applying this question to the phrasal content of the NSU, as shown in (5b) in an intuitive notation. Ginzburg and Sag (2001) also consider another contextual element, a *salient sub-utterance* of the current QUD, that provides parallelism conditions for the NSU phrase. In the present example, this would correspond to the *wh*-phrase *"Who"*, and appropriate dependencies like case matching could be established between this and the NSU phrase.

Thus, taking the approach of Ginzburg and collaborators as point of departure, the view I adopt is to see NSUs as context-dependent expressions like for instance deictics—i.e. as functions whose domain determines the contextual coordinates relevant for their resolution, and whose range specifies how these are used to resolve their sentential content and partially determine their form.

While Ginzburg and Sag (2001) couch their analysis in terms of Head-driven Phrase Structure Grammar, to formalise the main classes in the NSU taxonomy I explore the use of Type Theory with Records—an extension of Martin-Löf's type theory with records and record types. In line with the dynamic semantics tradition and inspired, in particular, by recent work by Robin Cooper (2006b) on the formalisation of update rules in computational dialogue modelling, I model NSU types—and utterance types in general—as families of information state types. These are functions whose output is the type of the information state after an utterance has taken place, and whose input is the current information state, which is constrained to contain the contextual coordinates required for interpretation.

As will be shown throughout this thesis, not all NSUs require the same contextual background. One of the factors that distinguishes NSUs in this respect is the distance from their *antecedent*—i.e. the contextual utterance used for resolution. While most NSUs are adjacent to their antecedent, it is well-known that some NSU classes like short answers can appear several turns away. In Ginzburg's theory, this observation motivates the definition of QUD as an ordered *set* of questions under discussion, which makes available potential non-adjacent antecedents.

Another factor that distinguishes different NSU classes, which we have already encountered in example (3), is the nature of the contextual information on which NSUs depend. This is varied, and can range from being purely semantic to including syntactic and phonological material. Yet a related aspect is the strategy by means of which the required contextual information becomes available. For instance, the current question under discussion can arise by several means. Often this question is raised explicitly, like in (5a) above. But it can also be *accommodated*, like in (6), where we can see B's reply as presupposing the question *"Who's making the decisions?"*, which in this case has not explicitly been raised.

(6) A: Is the president making the decisions?
B: ***The fund manager.***

Thus, factors like adjacency *vs.* distance, the nature and extent of the dependencies observed, and the mechanism by means of which antecedents become available allow us to establish a ranking of NSU classes relative to the complexity of the information state and the mechanisms required for resolution.

This is connected to an additional contribution of the thesis, a hierarchy of abstract models for dialogue protocols. Dialogue protocols encode frequently reoccurring sequences of utterance types, like questions being followed by answers, or assertions being either acknowledged, discussed or elaborated upon. As such, they characterise the range of *preferred* or *less-marked* follow-ups in particular dialogue situations, thus reflecting the expectations of dialogue participants given a particular contribution. In this thesis I propose a hierarchy of abstract communication protocols based on the expressive power of well-known machine models from the theory of computation, taking as a starting point protocols based on deterministic finite automata. As the protocols in the hierarchy differ on the information manipulated at each state by an abstract machine model, some correlations can be established between the hierarchy of protocols and the ranking of NSU classes. I believe this opens the door to a novel and interesting synthesis between formal language theory and dialogue semantics.

1.3 Thesis Outline

The structure of the thesis is as follows:

- **Chapter 2: A Corpus-based Taxonomy of NSUs**
  In this chapter I present a taxonomy of NSUs based on the outcomes of corpus work carried out using the dialogue transcripts of the British National Corpus. The taxonomy contains 15 classes, which are described in detail. The chapter also reports on the particulars of two corpus studies, one covering the full range of NSU classes and the other focussing on sluices.

- **Chapter 3: Some Previous Approaches**
  This chapter reviews some existing approaches to NSUs, namely the Higher Order Unification account of ellipsis by Dalrymple et al. (1991) and Pulman (1997); the logic-based approach by Dekker (2003a,b); the minimalist analysis of Merchant (2004); the SDRT-based theory of Schlangen (2003), and the approaches of Ginzburg and Sag (2001) and Ginzburg and Cooper (2004). In the same chapter I also introduce my own previous proposal based on Dynamic Logic.

- **Chapter 4: A Type-theoretical NSU Grammar**
  This chapter defines grammatical types for the main NSU classes in the taxonomy put forward in Chapter 2. These are formalised in terms of Type Theory with Records and are modelled in a way akin to update rules in computational dialogue modelling (Cooper 2006b). I start by introducing the formalism, and then group the different NSU types according to the complexity of the information state and the kind of information required for the resolution process.

- **Chapter 5: Abstract Models for Dialogue Protocols**
  In this chapter I focus on the formal properties of dialogue protocols, identifying a variety of features that have an impact on the complexity of the dialogue structure. This motivates a hierarchy of protocols based on the expressive power of well-known machine models from formal language theory and the theory of computation. The chapter also highlights correlations between this hierarchy and the different groups of NSU classes identified in Chapter 4.

- **Chapter 6: Automatic Classification of NSUs**
  This chapter addresses the task of automatically classifying NSUs using machine learning techniques. After identifying a collection of features that capture relevant properties for NSU classification, I report the results of two sets of experiments performed using several machine learners. In analogy to Chapter 2, the first experiment covers the full range of NSU classes, while the second one concentrates on sluices.

- **Chapter 7: Conclusions**
  In this final chapter I present my conclusions, briefly summarising the main contributions made and providing an outlook on possible future extensions of the work presented in this thesis.

- **Appendices**
  This thesis has three appendices. Appendix A presents a generalisation of the *PABAK* measure for inter-annotator agreement for any number of categories and coders. Appendix B gives the full details of the formal definitions of Type Theory with Records. Finally, Appendix C contains a grammar fragment based on the representations presented in Chapter 4.

1.4 Publications

This thesis is, in part, based on the following publications:

- Raquel Fernández and Jonathan Ginzburg. Non-sentential utterances: A corpus study. *Traitement automatique des languages. Dialogue*, 43(2):13–42, 2002.
- Raquel Fernández. A dynamic logic formalisation of inquiry-oriented dialogues. In *Proceedings of the 6th CLUK Colloquium*. Edinburgh, 2003.
- Raquel Fernández. A dynamic logic formalisation of the dialogue gameboard. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics. Student Research Workshop*. Budapest, 2003.
- Raquel Fernández and Ulle Endriss. Towards a hierarchy of abstract models for dialogue protocols. In *Proceedings of the 5th International Tbilisi Symposium on Language, Logic and Computation*. Tbilisi, 2003.
- Matthew Purver and Raquel Fernández. Utterances as update instructions. In *Proceedings of the 7th Workshop on the Semantics and Pragmatics of Dialogue (DiaBruck)*. Saarbrücken, 2003.
- Raquel Fernández and Matthew Purver. Information state update: Semantics or pragmatics? In *Proceedings of the 8th Workshop on the Semantics and Pragmatics of Dialogue (Catalog)*. Barcelona, 2004.
- Raquel Fernández, Jonathan Ginzburg, and Shalom Lappin. Classifying ellipsis in dialogue: A machine learning approach. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004)*. Geneva, 2004.
- Raquel Fernández, Jonathan Ginzburg, and Shalom Lappin. Automatic bare sluice disambiguation in dialogue. In *Proceedings of the 6th International Workshop of Computational Semantics (IWCS-6)*. Tilburg, 2005.
- Raquel Fernández, Jonathan Ginzburg, and Shalom Lappin. Using machine learning for non-sentential utterance classification. In *Proceedings of the 6th SIGdial Workshop on Discourse and Dialogue*. Lisbon, 2005.
- Raquel Fernández. The dynamics of utterances: grounding and update in Type Theory with Records. In *Proceedings of the Workshop on Discourse Domains and Information Structure (ESSLLI 2005)*. Edinburgh, 2005.

2 A Corpus-based Taxonomy of Non-Sentential Utterances

In this chapter I introduce a taxonomy of NSUs created on the grounds of an empirical study carried out using the British National Corpus (Burnard 2000). The taxonomy contains 15 classes, which will be described in detail in Section 2.1. After some remarks on some existing classifications of NSUs in Section 2.2, I will present the results of two corpus studies—one covering the full range of NSU classes in Section 2.3, and another one centered around *sluices* (i.e. bare *wh*-phrases) in Section 2.4.

2.1 The Taxonomy

The taxonomy of NSUs I present in this chapter is based on a corpus study performed using a portion of the ∼10 million word dialogue transcripts of the British National Corpus (BNC), a ∼100 million word corpus of current British English compiled by the Oxford University Text Archive (Burnard 2000). I will first say a few words on the methodology and rationale governing the classification, and then proceed to describe the NSU classes in it.

2.1.1 Design and Methodology

The present taxonomy of NSUs emerged after manual examination of ten dialogue transcripts randomly chosen from the BNC. These were exhaustively analysed in order to capture the maximal number of phenomena. In particular, this involved systematic reading of two transcripts of completely unrestricted, free conversation (files KST and KSV), three transcripts of informal interviews (files K68, K69 and JA2), two transcripts of more formal interviews (files K6K and K65), two transcripts of seminars (file JJ7 and JK1), and one transcript of a public county council meeting (file J9T).

From an empirical point of view, the current taxonomy is constrained by the particular instances of NSUs that were encountered in the analysed corpus, although the ontological choices one can take from there are of course rather open territory. In this respect, the kinds of fragments covered by Ginzburg and Sag (2001)—namely short answers, direct and reprise sluices, clarification fragments and propositional lexemes like *"yes"* and *"no"*—serve as the ground of the present classification. The decisions made from this starting point will become clear when I describe in detail the different NSU classes in the next section. But before I turn to this task, a word about *potential* NSUs that were not considered as such is in order.

To start with, I do not consider NSUs fragmentary utterances resulting from interruptions or overlaps. In the BNC utterances split by interruptions are transcribed as different items and are given a different identifier number.[1] In (7), for instance, A's utterance is interrupted by B's cough, which overlaps with part of A's contribution. Even though the fragment *"about this"* is marked up as a different sentence unit in the corpus, I do not consider these fragments proper NSUs, and hence they are not covered by the present taxonomy.

(7) A: Now then <*pause*> I don't know if she $\underline{\text{talked to you}}$
B: $\underline{<\textit{cough}>}$
A: about this.
[BNC: KSN 1621–1622]

More substantially, the current taxonomy does not include greetings and closings like *"Hello"* and *"Bye"*. I regard these utterances as explicit performatives with no associated descriptive content. This means that the main property of NSUs, namely the fact that (part of) their descriptive content has to be resolved contextually, does not hold for these utterances, as the illocutionary relation that constitutes their main predicate does not select for any message argument. The meaning of *"Hello"* can be informally represented as in (8a). This contrasts, for instance, with plain acknowledgements like *"mhm"* and *"aha"*, which I do consider NSUs. The illocutionary relation associated with this kind of utterances selects for a message type—a proposition—which constitutes its descriptive content and which has to be recovered from context.

(8) a. Hello → `Greet`(*speaker, addressee*)

b. Mhm → `Acknowledge`(*speaker, addressee, P*)

[1] More on the BNC sentence mark-up and the transcription of speech overlaps will be said in Section 2.3.4, where I report the results of measuring the distance between NSUs and their antecedents—i.e. the contextual utterances with respect to which NSUs are resolved. I shall use underlining to indicate speech overlap. When underlining is not used, utterances take place sequentially.

2.1.2 Rationale

As mentioned at the onset, the present taxonomy contains 15 different NSU classes. These can be grouped into three main families, roughly corresponding to speech act types: *acknowledgements*, *questions*, and *statements*—the latter being further split into *answers* and *extensions*.[2] I also use a term *completions*, that somehow falls outside of this general scheme—it is concerned with collaborative utterances. My account of the resolution of NSUs, that will be presented in Chapter 4, focuses on *acknowledgements*, *questions*, and *answers*. Table 2.1 shows an overview of the full range of NSU classes included in the taxonomy.

It is now finally time to turn to the presentation of the classes included in the taxonomy. In what follows I describe them in detail and give some corpus examples of each class.

2.1.3 The NSU Classes

2.1.3.1 Plain Acknowledgement

The class plain acknowledgement refers to NSUs like e.g. *"yeah"*, *"mhm"*, *"ok"* that give explicit positive evidence that the previous contribution, typically an assertion, was understood and/or accepted. It is important to take into account that acknowledgements and acceptances are distinct acts: the former involves evidence of understanding, whereas the latter indicates that an assertion is accepted. A form like the one in (9a), for instance, seems to indicate little more than mere understanding, while that in (9b) seems to be closer to an acceptance. However, since often acknowledgements in the form of NSUs simultaneously signal acceptance, the current taxonomy conflates the two acts within the same class.

(9) a. A: I know that they enjoy debating these issues.
B: ***Mhm.***
[BNC: KRW 146–147]

b. A: We should get off and interview Anna.
B: ***Oh yes.***
[BNC: KP4 4079–4080]

[2]Borrowing the terminology employed in the DAMSL markup scheme (Allen and Core 1997), the main distinction between these two groups can be taken to be that answers have a *backward-looking function*, while extensions don't. Also note that *statements* acts as a cover term for different more fine-grained speech acts, like assertions, exclamations etc. Short answers, for instance, are typically assertions, while factual modifiers are perhaps better characterised as exclamations.

Acknowledgements:

- Plain Acknowledgement
- Repeated Acknowledgement

Questions:

- Clarification Ellipsis
- Direct Sluice
- Check Question

Answers:

- Short Answer
- Plain Affirmative Answer
- Repeated Affirmative Answer
- Propositional Modifier
- Plain Rejection
- Helpful Rejection

Extensions:

- Factual Modifier
- Bare Modifier Phrase
- Conjunct

Completions:

- Filler

Table 2.1: Overview of the taxonomy of NSU

2.1.3.2 Repeated Acknowledgement

I distinguish plain acknowledgements from *repeated* acknowledgements that consist of a verbatim repetition or a reformulation of (a constituent of) the antecedent utterance, which as mentioned above is typically an assertion. The following are some examples of this NSU class:

(10) a. A: I'm at a little place called Ellenthorpe.
B: ***Ellenthorpe.***
[BNC: HV0 383–384]

b. A: [. . .] he's <*unclear*> in the theater, his favourite is musical
B: ***Yes, musical.***
[BNC: JSN 189–190]

c. A: She was questioning.
B: ***Questioning yes.***
[BNC: JYM 282–283]

d. A: Oh so if you press enter it'll come down one line.
B: ***Enter.***
That big key on the side.
[BNC: G4K 102–104]

Assuming that there is a continuum of different strengths of positive feedback, a reformulation like that in (10d) may provide stronger evidence of understanding than a verbatim repetition, which in turn is stronger than a plain acknowledgement. However, as pointed out in (Allen and Core 1997), repeated acknowledgements "do not necessarily make any further commitment as to whether the responder agrees with or believes the antecedent".

There is the issue of whether the focus-ground structure of the antecedent affects the felicity of a repeated acknowledgement, as seems to be the case with repeated affirmative answers (see Subsection 2.1.3.8 below). However the tendency seems to be to repeat just the most recent constituent, i.e. the last one. In connection to this, Ginzburg and Cooper (2004) note that a repeated acknowledgement "need not be understood as involving an intention to highlight [the repeated] constituent". They give the following example to illustrate the point:

(11) (Context: B is a waitress in an Edinburgh diner)
A: I'll be having chips and beans and a capuccino.
B: ***And a capuccino, OK.*** (attested example)
[from (Ginzburg and Cooper 2004), p.50]

2.1.3.3 Clarification Ellipsis (CE)

I use the category Clarification Ellipsis (CE) to classify all non-sentential clarification requests (CRs). As has been noted for the whole range of CRs (elliptical and otherwise),[3] non-sentential CRs also present a variety of surface forms. Some of these are illustrated by examples (12a)-(12d) below. In general the class CE covers all NSUs that indicate that some sort of understanding problem with a previous utterance has occurred.

[3] See (Purver 2004b) for an extensive study, but also e.g. (Schlangen 2004), (Rodríguez and Schlangen 2004) and (Larsson 2003).

(12) a. A: Where was this?
B: Oh what we call Waterhall.
A: ***Waterhall?***
B: Yeah, where you come down that hill from the boy's grave.
[BNC: HDH 46–49]

b. A: [. . .] You lift your crane out, so this part would come up.
B: ***The end?***
A: The end would come up and keep your load level on the ground you see.
[BNC: H5H 27–29]

c. A: Yeah cos, my mates were telling me the other day <*pause*> what's Gilly said?
B: ***Who?***
A: Marcus <*name*>.
[BNC: KSR 151–153]

d. A: What about in days gone by?
B: ***What?***
A: What about in days gone by?
[BNC: HDH 350–352]

One of the most common forms of CE is the one exemplified in (12a), where a constituent in the antecedent utterance is echoed with interrogative intonation.[4] This form of clarification NSU is referred to as *elliptical literal reprise* by Ginzburg and Sag (2001), *clarification ellipsis* by Ginzburg and Cooper (2004), and *reprise fragment* by Purver (2004b). Although usually there is a phonological parallelism between this form of CE and the clarified constituent, as shown in (12b) this does not always occur. In (12b) the phrase *"this part"* in A's first utterance is reprised by the bare phrase *"the end"* in B's response. Rodríguez and Schlangen (2004) distinguish between CEs like (12a) and (12b) by classifying the former as *repetition* and the latter as *reformulation*. However, even when the CE is not a verbatim repetition of the clarified phrase, the meaning of the CE and the antecedent constituent are intended to be co-referential. That is, in (12b) the reference of *"this part"* and *"the end"* are intended to be the same.

[4]In fact, the intonation pattern of these kind of fragments is not always unambiguously interrogative. Rodríguez and Schlangen (2004) report that "raising boundary tones [are less common] than expected to clarify reference resolution problems", which are one of the most common understanding problems addressed by fragments like (12a) and (12b). Unfortunately precise results are not given on this front as the authors only present a confusion matrix that shows correlations between raising/falling tones and the source of the understanding problem. No concrete results are reported on the correlation between boundary tones and the other attributes characterising CR forms.

(12c) shows a CE realised by a bare *wh*-phrase, which is used to reprise a constituent in the antecedent utterance. Ginzburg and Sag (2001) refer to this kind of CE as *reprise sluice*. A conventionalised bare *what*-phrase can also be used to clarify a whole utterance, like in (12d), indicating that a complete failure in communication has taken place.

2.1.3.4 Direct Sluice

Not all bare *wh*-phrases are used to request clarification. The present taxonomy distinguishes bare *wh*-phrases like the ones in (12c) and (12d) above from a different kind of non-sentential *wh*-questions, which following Ginzburg and Sag (2001) I refer to as Direct Sluices. These, unlike CE *wh*-questions, are not due to a communication breakdown, but instead ask for further information that was explicitly or implicitly quantified away in the antecedent utterance. The following examples show three instances of this NSU class:

(13) a. A: I know someone who's a good kisser.
B: ***Who?***
[BNC: KP4 511–512]

b. A: Who did you interview?
B: Benjamin.
A: ***When?***
B: Last night.
[BNC: KE0 138–141]

c. A: Anyway Jim so you're off to Australia?
B: Yeah.
A: ***Where?***
B: Er Melbourne.
[BNC: HV0 1015–1018]

As pointed out above, direct sluices query for additional information that has been quantified away. The quantificational context needed for a direct sluice can be made available in a variety of ways. The most obvious one is by means of an utterance that contains an explicit quantifier. This is the case in (13a), where the source utterance contains the quantifier *"someone"* that acts as the antecedent constituent of the direct sluice *"Who?"*. The sluice in (13b), on the other hand, is requesting additional temporal information that was implicitly quantified away in the antecedent utterance by means of tense.[5]

[5] I take the antecedent utterance, which is also a fragment, to convey the proposition *I interviewed Benjamin*. The past tense can be understood as quantifying over all time points or intervals which precede the utterance time.

(13c) shows an interesting example where the location *"Australia"*, which serves as the antecedent constituent of the bare *where*-sluice, is taken as quantifying over a set of possible sub-locations.

The term "sluicing" used to refer to elliptical *wh*-phrases dates back to Ross (1969), who offered the first detailed description of the phenomenon. His discussion is restricted to embedded sluices like the following, considered out of any dialogue context.

(14) They want to hire someone, but they don't know ***who***.

Ginzburg and Sag (2001) take a wider perspective which, besides embedded environments, also takes into account bare *wh*-phrases in dialogue. This allows them to distinguish between two main types of sluices: reprise sluices as the one we have seen above in (12c) for instance, and direct sluices like those in (13a)-(13c). The main differences between these two kinds of sluices they point out are (i) intonational: reprise sluices are accented while direct ones are not; (ii) contextual: direct sluices require the presence in context of a quantified utterance of some sort, while reprise sluices do not; (iii) finally, reprise and direct sluices also differ on whether they can appear in embedded clauses or not. These points are illustrated by the examples below.

(15) a. A: Jo phoned.
B: ***WHO?/#Who?***

b. A: Did anyone phone for me?
B: Yes.
A: Aha. ***Who?/#WHO?***

c. A: Many doctors want to join up.
B: I wonder ***who***.

d. A: Merle saw Mo.
B: #Jo also wonders ***WHO***. (= Jo also wonders who you claim saw Mo.)
[from (Ginzburg and Sag 2001), p.113]

(15a) shows that a direct sluice is not felicitous in the context of a non-quantified utterance. In (15b), since the acknowledgement indicates that A did not have any problem understanding B's utterance, only a direct, non-accented sluice is appropriate. (15c) and (15d) show that direct sluices can appear in embedded environments, while reprise sluices cannot, even though a sentential paraphrase of their intended content seems felicitous as an embedded clause.

Given the variety of interpretations that bare *wh*-phrases can convey, exemplified here by the reprise-CE/direct sluice dichotomy, my empirical investigation includes an additional corpus study focused on this *form* of NSU (which does not correspond to a unique class in the taxonomy), to examine further its possible interpretations and functions, and to provide a basis for its disambiguation. The sluicing corpus study and the results obtained with it are presented in Section 2.4.

2.1.3.5 Check Question

Finally, I turn to the last kind of NSUs used to ask questions. This NSU class, dubbed Check Question, refers to short queries, usually realised by rather conventionalised forms like *"alright?"* and *"okay?"*, that are requests for explicit feedback from the addressee. A couple of randomly picked-up examples are shown in (16).

(16) a. A: [. . .] this dimension is about er where you prefer to focus your attention and where you get your psychological energy from.
Okay?
B: Oh, right.
[BNC: G3Y 162–164]

b. A: So <*pause*> I'm allowed to record you.
Okay?
B: Yes.
[BNC: KSR 5–7]

The antecedent utterance of a check question expresses a proposition that has not yet been explicitly grounded. The check question then denotes a polar question which asks the addressee to give explicit feedback on whether s/he understood/accepted the source proposition.

It is worth noting that the term "check question" is sometimes used for a different phenomenon. Larsson (2003) for instance, uses this term for NSUs like the one in (17), which according to him "indicate understanding but lack of confidence in that understanding":

(17) A: I want to go to Paris.
B: ***To Paris?***
[from (Larsson 2003), p.77]

In (17) B asks for confirmation that his/her understanding of A's utterance was as intended, whereas in (16) A asks for confirmation of B's understanding of A's previous utterance. In the present taxonomy NSUs like (17) fall under the class CE.

2.1.3.6 Short Answer

"Short answer" is a wide cover term commonly used for fragments that occur in the context of a response to a query. In the present taxonomy this denomination is used only to designate non-sentential short answers which are responses to *wh*-questions. Affirmative and negative answers to polar questions are covered by other classes that will be introduced shortly. Short answers are perhaps the type of fragments that have received most attention in the literature (see e.g. Morgan 1973, Barton 1990, Ginzburg and Sag 2001, Schlangen and Lascarides 2002, Merchant 2004) and indeed, as we will see below, they are one of the most frequent NSUs found in corpora. The following are some examples extracted from the BNC:

(18) a. A: Who's this book by?
B: ***Luhmann L U H M A double N.***
[BNC: G4V 132–133]

b. A: What, what are you talking about John?
B: Oh, erm, ***terminal illness.***
[BNC: JK8 177–178]

c. A: Now the triangle adds up to how many degrees?
B: <*pause*> ***Hundred and eighty.***
[BNC: KND 48–49]

d. A: Can you tell me where you got that information from?
B: ***From our wages and salary department.***
[BNC: K6Y 94–95]

As the examples in (18) show, typically short answers have as antecedent a question containing a *wh*-phrase. This, which can be thought of as the antecedent constituent of the fragment, can appear *in situ* (18c) or be part of an embedded clause (18d). However, the contextual background of a short answer does not always explicitly contain a *wh*-phrase. This is often the case when a short answer is used to reply to a CE question which conveys a *wh*-question but, because of its non-sentential nature, does not contain an overt *wh*-phrase. In (19a), B's clarification question, which acts as antecedent of the short answer, could be paraphrased with a *wh*-question close to *"what did you utter after 'vague'?"*.

(19) a. A: [. . .] they're intolerant towards vague ideas and people.
B: Vague and?
A: ***Vague ideas and people.***
[BNC: JJH 65–66]

b. A: Are you right or left handed?
B: ***Right handed.***
[BNC: G3Y 96–97]

Something similar applies to short answers used to respond to alternative questions like (19b). Despite not having an overt *wh*-phrase, (19b) can be analysed as denoting the *wh*-question "What are you, left-handed or right-handed?", that is a question $\lambda X.X(addr)$ where X ranges over the restricted domain provided by the disjunction.[6]

2.1.3.7 Plain Affirmative Answer

I now turn to NSUs that have as contextual background a polar question. The first class I consider are plain affirmative answers like (20a). Again, the question which the NSU is an answer to can be a CE query that despite exhibiting a non-canonical syntactic structure nevertheless conveys a polar question. An example of this is given in (20b), where A's CE can be paraphrased as "Did you just say 'hard'?" or "Did you mean/say that the school was *HARD*?".

(20) a. A: Did, did you know that Spinal Tap was a film before the band came out?
B: ***Yes.***
[BNC: KP4 4153–4154]

b. A: What was the school like?
B: Hard.
A: Hard?
B: ***Yes.***
[BNC: H5G 73–75]

2.1.3.8 Repeated Affirmative Answer

I distinguish plain affirmative answers like the ones above from a different kind of NSUs that also convey a positive answer to a polar question. This class, dubbed Repeated Affirmative Answer, is exemplified in (21):

[6]Some questions with a disjunction combine a reading as alternative (*wh*-)questions with a polar question reading:

(i) A: Did Maggi have coffee or tea?
(= Is it the case that Maggi had one of these two things, coffee or tea?)
B: No, she just had a piece of cake.

The two readings seem to correlate with different intonational patterns of the disjuncts.

(21) a. A: Did you shout very loud?
B: ***Very loud, yes.***
[BNC: JJW 571-572]

b. A: The one three six three goes out through the Sutton on Forest, does it?
B: ***Sutton on Forest, yeah.***
[BNC: J9T 311–312]

Repeated affirmative answers are NSUs that achieve the "affirmative answer" effect by a verbatim repetition or reformulation of a sub-utterance of the source question. Note that the perception that the utterer of the NSU has of the focus-ground structure of the antecedent polar question may affect the felicity of repeated affirmative answers. If a constituent is clearly focussed, then the answer sounds felicitous only if it is a repetition of the focussed constituent:

(22) A: Did you *SHOUT* very loud?
B1: #Very loud, yes.
B2: Shout, yes.

2.1.3.9 Propositional Modifier

The class Propositional Modifier refers to modal adverbs like those in (23), which can function as NSUs conveying a complete message. In their stand-alone uses, such adverbs take as an argument a contextual proposition—either from an assertion under discussion or from a polar question in the context—which they modify.

(23) a. A: They wanted to come back even further didn't they?
B: They did.
A: ***Recently.***
[BNC: K69 22–24]

b. A: I wonder if that would be worth getting?
B: ***Probably not.***
[BNC: H61 81–82]

2.1.3.10 Plain Rejection

This NSU class can be considered the dual of plain affirmative answers. Typically the NSUs that fall under this class are plain negative answers to polar questions like those in (24).

(24) a. A: [. . .] is there a spider in my hair?
B: ***No.***
[BNC: KP4 15443–1544]

b. A: You starving?
B: ***No way.***
[BNC: 152–153]

However, in a way similar to propositional modifiers, I also classify as plain rejections negative responses to assertions, like those in the following examples: [7]

(25) a. A: You're joking.
B: ***No.***
[BNC: J8B 1443–1444]

b. A: I think I left it too long.
B: ***No no.***
[BNC: G43 26–27]

2.1.3.11 Helpful Rejection

When a polar question is answered negatively or when an assertion is rejected, a cooperative speaker will often accompany the rejection with an NSU that provides a contrasting alternative to the rejected proposition (assuming, of course, that such alternative exists and is known to the speaker.) I call this kind of NSUs "Helpful Rejections", after Engdahl et al. (2000) who call them "helpful answers". The following are some examples of instances that fall under this NSU class:

(26) a. A: So there'd be two clerks and two lads?
A: ***No, one clerk.***
[BNC: HDK 1776–177]

b. A: Well I felt sure it was two hundred pounds a, a week.
B: ***No fifty pounds ten pence per person.***
[BNC: K6Y 112–113]

[7] I do not follow the same strategy with affirmative answers though. This is justified because a rejection of a proposition p implies understanding + rejecting p, whereas not all positive responses to assertions that p imply understanding + accepting p. Since the difference between degrees of "positiveness" (e.g. between backchannel, acceptance, agreement, etc.) can often be uncertain, I choose to classify non-sentential positive feedback to assertions with the classes "Plain Acknowledgement/Repeated Acknowledgement" (Sections 2.1.3.1 and 2.1.3.2) distinct from Plain Affirmative Answer/Repeated Affirmative Answer. Negative feedback is of course accounted for by the class CE.

c. A: A bigger pipe has got m more resistance. [. . .]
B: ***Less resistance.***
[BNC: GYR 318–320]

d. A: How much do you think?
B: Three hundred pounds.
C: ***More.***
B: A thousand pounds.
A: ***More.***
[BNC: G4X 44–48]

Note that in fact the "no" can be dispensed with, as illustrated by (26c) and (26d). This seems to be more common when the antecedent is an assertion, although it is also possible with interrogative antecedents, as the following constructed example shows:

(27) A: Would you like some red wine?
B: ***White please.***

As with Repeated Affirmative Answers, the focus-ground structure of the antecedent utterance seems to affect the acceptability of the helpful rejection.

(28) A: Are you *FLYING* to London?
B1: No.
B2: No, (I'm) ***SAILING***.
B3: No, #I'm sailing to *LONDON*.
[from (Engdahl et al. 2000), p.33]

However, when there is no clear focus-ground partition of the antecedent utterance, any constituent can act as antecedent:

(29) A: John is flying to London.
B: (No,) Mary / sailing / to Edinburgh.

2.1.3.12 Factual Modifier

I will now describe those NSU classes that can be thought of as extensions or continuations of the dialogue. These NSUs express statements which go on with the dialogue either by modifying some contextually available entity or by adding information by means of a conjuncted fragment.

The first of these NSU classes is that of factual modifiers, whose members are exclamative factual adjectives that express an attitude of the speaker towards some contextually presupposed fact. A couple of examples of these factual modifiers are given in (30).

(30) a. A: There's your keys.
B: ***Oh great!***
[BNC: KSR 137–138]

b. A: So we we have proper logs? Over there?
B: It's possible.
A: ***Brilliant!***
[BNC: KSV 2991–2994]

2.1.3.13 Bare Modifier Phrase

A related class of NSUs is that of bare modifier phrases, which behave like non-sentential adjuncts modifying a contextual utterance. As the instances in (31) show, they are typically phrases headed by prepositions or adverbs.

(31) a. A: . . . they got men and women in the same dormitory!
B: ***With the same showers!***
[BNC: KST 992–996]

b. A: It's the sickness, it just, you seem to take in every kind of sickness with it.
B: That's right.
A: ***Even with the wee tablet.***
[BNC: H60 33–35]

c. A: [. . .] you got the stables shut up before it got dark at four o'clock.
B: ***Mm in the winter time mm.***
[BNC: HDH 130–131]

2.1.3.14 Conjunct

The last class of NSUs which act as extensions do so by connecting a fragment to the dialogue context by means of a conjunction. Of course the relation between the fragment and the contextual entity to which it gets connected will be determined by the semantic properties of the conjunction involved. The following are a couple of examples of this NSU class:

(32) A: Alistair erm he's, he's made himself coordinator.
B: ***And section engineer.***
[BNC: H48 141–142]

A: If it'd been me and I were gonna put it there, I would have put a window there <*pause*> to look out of.

B: ***Or a door there.***
[BNC: KB1 4419–4420]

2.1.3.15 Filler

Finally the last class in the current taxonomy of NSUs refers to fragments used to complete a previous unfinished utterance. I call this class of NSUs 'Fillers'.[8]

(33) a. A: [. . .] twenty two percent is er <*pause*>
B: ***Maxwell.***
[BNC: G3U 292–293]

b. A: And the second one is a book by
B: ***Beardsmoor*** *[sic]* and he <*unclear*> [. . .]
[BNC: G4V 121–122]

2.2 Other Taxonomies

Although NSUs have not received a major attention in the literature, there are some existing taxonomies of fragments. A well known classification within the Artificial Intelligence tradition is that of Carberry (1990), who categorises NSUs according to the speaker's plans and intentions. From a perhaps more linguistic perspective, there is also the classification offered by Barton (1990), who also resorts heavily to intentions, distinguishing NSUs according to the kind of inference needed for their resolution. My focus on this section, however, will be on a more recent taxonomy of NSUs, namely that presented in (Schlangen 2003), where the interested reader can find short reviews of the other aforementioned classifications.

Schlangen's taxonomy of fragments is shaped by the theory of discourse interpretation that the author employs as a framework to explicate the resolution of NSUs, to wit Segmented Discourse Representation Theory or SDRT (Asher 1993, Asher and Lascarides 2003). I will expand on this theory in the next chapter, in Section 3.3.1, where I review in more detail Schlangen's approach to NSU resolution. Here it suffices to say that SDRT focuses on the *rhetorical relations* by means of which different discourse units are connected to one another. These relations are seen as determining the coherence of a discourse or dialogue, and are non-monotonically inferred via a combination of several sources of information, ranging from lexical semantics to world knowledge. With SDRT as background, Schlangen classifies NSUs along two dimensions. The first dimension

[8]The antecedent of a filler is of course fragmentary by definition, although accidentally so. It will therefore not be an NSU, as it does not convey a full message.

concerns the SDRT rhetorical relation in which the NSU stands to its antecedent utterance. The second dimension, which I will address in more detail in the next chapter, has to do with the source of the information needed to resolve the content of the fragment. Here Schlangen distinguishes between NSUs that can be resolved by means of material that is present in the antecedent utterance (what he calls *resolution-via-identity* NSUs) and NSUs that are resolved by means of inferential processes that typically involve world knowledge and plans (*resolution-via-inference* NSUs).

Schlangen identifies 24 rhetorical relations by means of which NSUs can be connected to the previous context. These include, amongst others, different versions of *Question-Answer-Pair*, *Explanation*, *Elaboration*, *Narration* etc. Several relations can hold of a single NSU. For instance, what in our NSU taxonomy are Helpful Rejections, like the already seen example in (34), would presumably require at least two rhetorical relations in Schlangen's system—*Question-Answer-Pair* and *Correction* or *Contrast*.

(34) A: So there'd be two clerks and two lads?
B: ***No, one clerk.***
[BNC: HDK 1776–177]

One limitation of this taxonomy is that overall the notion of metacommunicative interaction is neglected. In particular, those NSUs used to request clarification of a problematic utterance are not identified separately as instances of a distinguished category, but are instead mixed up with other kinds of questions. For instance, the NSU in (35a), which Schlangen describes as asking "for more details about the event described in α [the antecedent]", is classified together with the NSU in (35b), which Schlangen himself paraphrases as "Who is Shmul?". In our taxonomy the first of these NSUs would be classified as a Direct Sluice, while the second one would be an instance of CE. The distinction between direct sluices and clarification NSUs, argued for in Section 2.1.3.4 above, is not captured by Schlangen's taxonomy, where these two NSUs are considered instances of the single relation $Explanation_q$.[9]

(35) a. A: It's a microphone. Recording conversations.
B: ***With who?***

b. A: Did Shmul call?
B: ***Schmul?***

In order to test his taxonomy, Schlangen also conducts a corpus study. Unfortunately however, the empirical investigation only covers the dimension regarding the rhetorical

[9]The examples in (35) are extracted from (Schlangen 2003) p. 31.

relations of NSUs to elements in the context. The *resolution-via-identity/resolution-via-inference* dichotomy is not explored empirically. Thus, despite the fact that this is an interesting distinction, we are left to wonder what percentage of NSU in the data can be resolved with each of the resolution strategies. As for the results of the corpus study he obtains, the rate of NSUs found is higher that ours (19.4% *vs*. 9%) while the coverage of the taxonomy is slightly lower (96.1% *vs*. 98.8%). The results of our corpus investigation of NSUs are presented in detail in the coming section.

2.3 The NSU Corpus Study

In order to know the frequency of the NSU classes identified and to test the coverage of the taxonomy presented in Section 2.1, a corpus study was carried out using the dialogue transcripts of the BNC.[10] In this section I describe the corpus investigation and the results obtained. In Section 2.4,I will present an additional corpus study which, as announced earlier, focuses on a particular form of NSU, namely bare *wh*-phrases or so called *sluices*. These two corpus studies supply the annotated data sets used in Chapter 6 for the machine learning experiments designed to guide the automatic disambiguation of NSU classes.

2.3.1 The Corpus

The present corpus of NSUs includes and extends the sub-corpus used in (Fernández and Ginzburg 2002). It was created by manual annotation of a randomly selected section of 200-speaker-turns from 54 BNC files. Of these files, 29 are transcripts of conversations between two dialogue participants, and 25 files are multi-party transcripts. The total of transcripts used covers a wide variety of domains, from free conversation to meetings, tutorials and training sessions, as well as interviews and transcripts of medical consultations.

The examined sub-corpus contains 14,315 sentences. Sentences in the BNC are identified by the CLAWS segmentation scheme (Garside 1987) and each unit is assigned an identifier number. Within this sub-corpus we found a total of 1299 NSUs. These were labelled according to the taxonomy of NSUs presented in the previous section together with an additional class 'Other' introduced to catch all NSUs that did not fall in any of the classes in the taxonomy. All NSUs that could be classified with the taxonomy classes were additionally tagged with the sentence number of their antecedent utterance.

[10]The portion of the BNC used in this testing phase was of course different from the set of files used to construct the taxonomy (see Section 2.1.1).

2.3.2 Reliability

The labelling of the entire corpus of NSUs was done by the author. To assess the reliability of the taxonomy, a small study with two additional, non-expert annotators was conducted. These annotated a total of 50 randomly selected instances (containing a minimum of 2 instances of each NSU class as labelled by the expert annotator) with the classes in the taxonomy. The agreement obtained by the three annotators is reasonably good, yielding a *kappa* score of 76%. The non-expert annotators were also asked to identify the antecedent sentence of each NSU—i.e. the contextual sentence with respect to which the NSU is resolved. Using the expert annotation as a gold standard, they achieve 96% and 92% accuracy in this task.

2.3.3 Results

A total of 1299 NSUs were found, which make up 9% of the total of sentences in the sub-corpus. These results are in line with the rates reported in other recent corpus studies of fragments: 11.15% in (Fernández and Ginzburg 2002), 10.2% in (Schlangen and Lascarides 2003), 8.2% in (Schlangen 2005). In the following sections I discuss the coverage of the taxonomy, the distribution of NSU classes and the results regarding the distance of the NSUs from their antecedents.

2.3.3.1 Coverage

Of the total of NSUs found—1299 instances—1283 could be classified with one of the classes in our taxonomy. The NSUs not covered by the classification only make up 1.2% (16 instances) of the total of NSUs found. The coverage of the taxonomy is therefore satisfactory. Table 2.2 shows a summary of the results.

Total of sentences		14,314
NSUs	1299	9%
Covered	1283	98.8%
Other	16	1.2%

Table 2.2: Summary of results

The highlighted NSU in example (36) is one of the instances not covered by our taxonomy. The NSU *"Good fun?"*—which presumably resolves to the question *"Was to chase up all the leads afterwards good fun?"*—is not easily captured by any of the NSU categories that denote questions (Direct Sluice, CE, and Check Question), and therefore was classified as Other.

(36) A: I used to go to all the erm exhibitions Olympia and GMEX and all of those.
Erm and then it was my job to chase up all the leads afterwards.
B: Right. ***Good fun?***
[BNC: JEA 212–216]

I should stress however that most of the utterances classified as Other were not entirely comprehensible utterances. In a dialogue fragment like (37), for instance, it is not possible to know what is going on due to the amount of utterances transcribed as *unclear*. The highlighted NSU could only be classified as Other.

(37) A: I'm not quite sure, I think most organisations have a certain amount of sum of money if I can remember from the workshops <*unclear*>.
B: Other than <*unclear*>.
A: <*unclear*>
C: ***Public sector.***
A: That's right.
B: Yeah they get financed.
[BNC: G4X 74–78]

Thus, with a rate of 98.8% coverage, the present taxonomy offers a satisfactory coverage of the data.

2.3.3.2 Distribution

The distribution of NSU classes that emerged after the annotation of the sub-corpus is shown in detail in Tables 2.3 and 2.4. The first of these table shows total numbers and percentages of each NSU class; the second one shows the distribution of the different NSU super-classes or families. By far the most common class can be seen to be Plain Acknowledgement, which together with Repeated Acknowledgement account for more than half of all NSUs found. This is followed in frequency by answers, with Short Answer (14.15%) and Plain Affirmative Answer (8%) being the most common classes within this group. Questions make up 9.6% of all NSUs, from which more than 73% constitute non-sentential clarification requests (i.e. 92 NSUs classified as CE out of 125 question NSUs). Extensions form 3.8% of the total, most of which are Factual Modifiers. Finally Completions make up 1.4% of all NSUs found.

NSU Class	**Total**	**%**
Plain Acknowledgement	599	46.1
Short Answer	188	14.5
Plain Affirmative Answer	105	8.0
Clarification Ellipsis	92	7.0
Repeated Acknowledgement	86	6.6
Plain Rejection	49	3.7
Factual Modifier	27	2.0
Repeated Affirmative Answer	26	2.0
Helpful Rejection	24	1.8
Check Question	22	1.7
Filler	18	1.4
Bare Modifier Phrase	15	1.1
Propositional Modifier	11	0.8
Direct Sluice	11	0.8
Conjunct	10	0.7
Other	16	1.2
Total	**1299**	**100**

Table 2.3: Distribution of NSU classes

NSU Family	**Total**	**%**
Acknowledgements	704	52.7
Answers	403	31.3
Questions	125	9.6
Extensions	48	3.8
Completions	18	1.4

Table 2.4: Distribution of NSU families

2.3.4 NSU-Antecedent Separation Distance

In this section I report the results obtained regarding the distance between the NSUs encountered and their antecedent utterances. A general picture of the results is shown in Table 2.5, where each NSU class is sorted by distance.

NSU Class	**Total**	**D1**	**D2**	**D3**	**D4**	**D5**	**D6**	**D>6**
Plain Acknowledgment	599	582	15	2				
Short Answer	188	105	21	16	6	5	7	28
Plain Affirmative Answer	105	100	5					
Clarification Ellipsis	92	76	13	2	1			
Repeated Acknowledgement	86	80	2	4				
Plain Rejection	49	48	1					
Factual Modifier	27	23	2	1	1			
Repeated Affirmative Answer	26	25	1					
Help Rejection	24	18	5		1			
Check Question	22	15	7					
Filler	18	16	1		1			
Bare Modifier Phrase	15	10	4			1		
Direct Sluice	11	10	1					
Propositional Modifier	11	10	1					
Conjunction Phrase	10	5	4	1				
Total	1283	1123	82	26	10	6	7	28
Percentage	100	87.5	6.4	2	0.8	0.5	0.6	2.2

Table 2.5: Total of NSUs sorted by class and distance

As mentioned above, in order to be able to measure the distance between the NSUs encountered and their antecedents, all classified NSUs were tagged with the sentence number of their antecedent utterance. In the BNC annotation, each sentence unit is assigned a sentence number. By default it is assumed that sentences are non-overlapping and that their numeration indicates temporal sequence. When this is not the case because speakers overlap, the tagging scheme encodes synchronous speech by means of an alignment map used to synchronize points within the transcription. However, even though information about simultaneous speech is available, overlapping sentences are annotated with different sentence numbers. The reported distance is therefore measured in terms of sentence numbers. It should however be noted that taking into account synchronous speech would not change the data reported in Table 2.5 in any significant way, as manual examination of all NSUs at more than distance 3 reveals that the transcription portion between antecedent and NSU does not contain any completely synchronous sentences in such cases.

Sometimes plain acknowledgements are uttered in response to a turn that contains several utterances labelled with different sentence numbers. In these cases I assume that the full turn is being acknowledged and, by convention, take the antecedent utterance to be the last sentence in that turn. Something similar applies to check questions. Typically check questions appear at the end of a turn that may contain several sentences (possibly with an acknowledgement given by the addressee between the end of the turn and the check question). The last sentence uttered by the speaker of the check question is the one labelled as its antecedent.

2.3.4.1 Distance Across Categories

The last row in Table 2.5 shows the distribution of NSU-antecedent separation distances as percentages of the total of NSUs found. This allows us to see that more than 87% of NSUs have a distance of 1 sentence (D1—i.e. the antecedent was the immediately preceding sentence), and that the vast majority (about 96%) have a distance of 3 sentences or less.

One striking result exhibited in Table 2.5 is the uneven distribution of long distance NSUs across categories. With a few exceptions, NSUs that have a distance of 3 sentences or more are exclusively short answers. Not only is the long distance phenomenon almost exclusively restricted to short answers, but the frequency of long distance short answers stands in strong contrast to the other NSUs classes; indeed, over 44% of short answers have more than distance 1, and over 24% have distance 4 or more, like the last answer in the following example (part of which we have already seen in (26d) in connection to helpful rejections):

(38) A: How much do you think?
B: Three hundred pounds.
C: More.
B: A thousand pounds.
A: More.
D: *<inhales> <unclear>*
A: ***Eleven hundred quid apparently, just that.***
[BNC: G4X 44–49]

2.3.4.2 Distance Asymmetry in Dialogue and Multilogue

Although the proportion of NSUs found in dialogue and multilogue is roughly the same (see Table 2.6), when taking into account the distance of NSUs from their antecedent, the proportion of long distance NSUs in multilogue increases radically: the longer the distance, the higher the proportion of NSUs that were found in multilogue. These differences are statistically significant ($\chi^2 = 62.5$, $p \leq 0.001$). In fact, as Table 2.7 shows, NSUs that have a distance of 6 sentences or more appear exclusively in multilogue transcripts.

	NSUs	BNC files
Dialogue	710	29
Multilogue	573	25
Total	1283	54

Table 2.6: Total of NSUs in dialogue and multilogue

Long Distance Short Answers Long distance short answers seem to be primarily an effect of multi-party interaction. Table 2.8 shows the total number of short answers found in dialogue and multilogue respectively, and the proportions sorted by distance over those totals. From this data it emerges that short answers are more common in multilogue than in dialogue—134 (71%) v. 54 (29%). Also, the distance pattern exhibited by these two groups is strikingly different: Only 18% of short answers found in dialogue have a distance of more than 1 sentence, with all of them having a distance of at most 3, like the short answer in (39).

(39) A: [...] cos what's three hundred and sixty divided by seven?
B: I don't know.
A: Yes I don't know either!
B: ***Fifty four point fifty one point four.***
[BNC: KND 197–200]

Group Size As Table 2.8 shows, all short answers at more than distance 3 appear in multilogues. Following (Fay et al. 2000), I distinguish between small groups (those with 3 to 5 participants) and large groups (those with more than 5 participants). The size of the group is determined by the amount of participants that are active when a particular short answer is uttered. I consider active participants those that have made a contribution within a window of 30 turns back from the turn where the short answer was uttered.

Table 2.9 shows the distribution of long distance short answers (distance > 3) in small and large groups respectively. This indicates that long distance short answers

Distance	**D1**	**D2**	**D3**	**D4**	**D5**	**D6**	**D>6**
Dialogue	658 (59%)	38 (46%)	11 (45%)	2 (20%)	1 (14%)	0 (0%)	0 (0%)
Multilogue	465 (41%)	44 (54%)	15 (55%)	8 (80%)	6 (86%)	7 (100%)	28 (100%)

Table 2.7: NSUs in dialogue and multilogue sorted by distance

Short Answers	**Total #**	**D1**	**D2**	**D3**	**D>3**
Dialogue	54	83	9	8	0
Multilogue	134	44	11	8	37

Table 2.8: % over the totals found in dialogue and multilogue

Group Size	**D>3**	**D≤3**	**Total**
≤**5**	20 (22%)	73 (78%)	93
>**5**	26 (63%)	15 (37%)	41

Table 2.9: Long distance short answers in small and large groups

are significantly more frequent in large groups ($\chi^2 = 22.17$, $p \leq 0.001$), though still reasonably common in small groups.

Large group multilogues in the corpus are all transcripts of tutorials, training sessions or seminars, which exhibit a rather particular structure. The general pattern involves a question being asked by the tutor or session leader, the other participants then taking turns to answer that question. The tutor or leader acts as turn manager. She assigns the turn explicitly usually by addressing the participants by their name without need to repeat the question under discussion. The following example, extracted from the transcript of a training session, is an example of this interaction pattern:

(40) A: Kim, what would you like?
B: Erm, ***waiver of premiums.***
A: Waiver of premiums, crafty. Er, Shirley?
C: ***Flexibility.***
A: Flexibility, oh that was the one you gave me, yes. Er, Janet?
D: ***Unit linking.***
A: Unit linking. Sue?
E: ***Selected period.***
[BNC: JK8 104–114]

Conversations amongst small groups on the other hand have a more unconstrained structure: after a question is asked, the participants tend to answer freely. Answers by different participants can follow one after the other without explicit acknowledgements nor turn management, like in (41a) (taken from a meeting transcript) and (41b) (extracted from a transcript of free conversation):

(41) a. A: Which country is er the biggest single supplier of coffee to this country?
B: ***Kenya.***
C: ***Yemen.***
D: ***Brazil.***
A: It's Uganda.
[BNC: G3U 255-258]

b. A: Are they bigger than me?
B: ***No.***
C: ***Yes.***
D: ***No. Same size really.***
[BNC: KP4 1565–1569]

I will not discuss further the differences between dialogue and multilogue here, which deserve a thesis on their own. For the formulation and discussion of some benchmarks that characterise dialogue and multilogue interaction extracted from the data presented above I refer the reader to (Ginzburg and Fernández 2005a).

2.4 The Sluicing Corpus Study

I now turn to describe a second corpus investigation focused on bare *wh*-phrases or *sluices*, a form of NSU that cuts across two classes in the present taxonomy—Direct Sluice and CE—which potentially indicates that sluices are a highly ambiguous NSU form. The study aims at providing an empirical ground for sluice disambiguation by investigating the different interpretations sluices can convey and establishing possible correlations between these interpretations and particular sluice types. Parts of the work presented in this section were originally published in (Fernández et al. 2004).

The next two sections describe the corpus and the annotation scheme used. The reliability of the annotation is discussed in detail in Subsection 2.4.3. In Section 2.4.4, I report the results obtained.

2.4.1 The Corpus

Because sluices have a well-defined surface form—they are bare *wh*-words—in this case it was possible to use an automatic mechanism to construct the sub-corpus. The sub-corpus of sluices was created using SCoRE (Purver 2001), a tool that allows one to search the BNC using regular expressions.

The dialogue transcripts of the BNC contain 5183 bare sluices (i.e. 5183 sentences consisting of just a *wh*-word). I distinguish between the following classes of bare sluices: *what*, *who*, *when*, *where*, *why*, *how* and *which*. Given that only 15 bare *which* were found, I also considered sluices of the form *which N*. Including *which N*, the corpus contains a total of 5343 sluices, whose distribution is shown in Table 2.10.

what	*why*	*who*	*where*	*which N*	*when*	*how*	*which*	**Total**
3045	1125	491	350	160	107	50	15	**5343**

Table 2.10: Total of sluices in the BNC

Two different samples of sluices extracted from the total found in the dialogue transcripts of the BNC were selected. The samples were created by arbitrarily selecting 50 sluices of each class (15 in the case of *which*). The first sample includes all instances of bare *how* and bare *which*, making up a total of 365 sluices. The second sample contains 50 instances of the remaining classes, making up a total of 300 sluices.

Note that the samples do not reflect the frequency of sluice types found in the full corpus. The inclusion of sufficient instances of the lesser frequent sluice types would have involved selecting much larger samples. Consequently it was decided to abstract over the true frequencies and create balanced samples whose size was manageable enough to make the manual annotation feasible. I will later return to the issue of the true frequencies in Section 2.4.4.

2.4.2 The Annotation Procedure

The annotation procedure consisted in classifying the two samples of sluices according to a set of categories—drawn from the theoretical distinctions introduced by Ginzburg and Sag (2001) and referred to in Section 2.1.3.4—corresponding to different sluice interpretations. The typology reflects the basic direct/reprise divide and incorporates other categories that cover additional readings, including an *unclear* class intended for those cases that cannot easily be classified by any of the other categories. The classification was done independently by 3 different annotators.

The following categories were used to classify the sluices in the first sample of the sub-corpus:

Direct This category corresponds to the NSU class Direct Sluice discussed in Section 2.1.3.4. Sluices conveying a direct reading are not triggered by a communication problem: they query for additional information that was quantified away in the antecedent, which is understood without difficulty. The sluice in (42) is an example of a sluice with direct reading: it asks for additional temporal information that is implicitly quantified away in the antecedent utterance.

(42) A: I'm leaving this school.
B: ***When?***
[BNC: KP3 537–538]

Reprise I have already referred to this reading in Sections 2.1.3.3 and 2.1.3.4. Sluices conveying a reprise reading emerge as a result of an understanding problem and are therefore classified as CE in the NSU taxonomy. They are used to clarify a particular aspect of the antecedent utterance corresponding to one of its constituents, which was not correctly comprehended. In (43) the reprise sluice has as antecedent constituent the pronoun *"he"*, whose reference could not be adequately grounded.

(43) A: What a useless fairy he was.
B: ***Who?***
[BNC: KCT 1752–1753]

Clarification As reprise, this category also corresponds to a sluice reading that falls in the NSU class CE. In this case the sluice is used to request clarification of the entire antecedent utterance, indicating a general breakdown in communication often at the acoustic level. The following is an example of a sluice with a clarification interpretation:

(44) A: Aye and what money did you get on it?
B: ***What?***
A: What money does the government pay you?
[BNC: KDJ 1077–1079]

Unclear I use this category to classify those sluices whose interpretation is difficult to grasp, possibly because the input is too poor to make a decision as to its resolution, as in the following example:

(45) A: *<unclear> <pause>*
B: ***Why?***
[BNC: KCN 5007]

After annotating the first sample, it emerged that there was a distinguished class of sluice readings that did not fall in any of the above categories. Because of this, it was decided to add a new category to the above set. The sluices in the second sample were then classified according to a set of five categories, including the following:

Wh-anaphor This category is used for the reading conveyed by sluices like (46). These are characterised by having as antecedent utterance a (possibly embedded) *wh*-question, whose *wh*-phrase acts as the antecedent of the sluice—hence the denomination "wh-anaphora". The sluice is then resolved to the *wh*-question in the antecedent utterance.

(46) A: We're gonna find poison apple and I know where that one is.
B: ***Where?***
[BNC: KD1 2370–2371]

2.4.3 Reliability

Although the *kappa* coefficient has been the standard measure for evaluating inter-coder agreement in computational linguistics since (Carletta 1996), several authors have identified difficulties related to its interpretation (see e.g. Cicchetti and Feinstein 1990). Recall that the *kappa* score for a set of classifications is computed as

$$\kappa = \frac{P(A) - P(E)}{1 - P(E)}$$

where $P(A)$ is the proportion of actual agreements and $P(E)$ is the proportion of expected agreement by chance. The denominator is the total proportion less the proportion of chance expectation.

There are two main problems that affect κ. These are the **bias** problem, which arises when the coders differ in assessing of the frequency of instantiation for the categories used, and the **prevalence** problem, which occurs when the distributions for categories are skewed (highly unequal instantiation across categories.). The prevalence problem arises because skewing the distributions of categories in the data increases $P(E)$, and the larger the value of $P(E)$, the lower the value of κ.

Even though the bias problem does not seem to affect the present annotation, the *kappa* scores obtained are strongly affected by prevalence. The reason for this will become clear in the next section when I describe the distribution patters for each type of sluice. Therefore, following Di Eugenio and Glass (2004), I will report two measures of reliability for the sluice annotations: κ and an additional measure $PABAK$ (*prevalence-adjusted bias-adjusted kappa*). $PABAK$ was originally introduced by Byrt et al. (1993) to adjust κ to control for differences in prevalence and for bias among observers.[11] The

[11]In fact, (Di Eugenio and Glass 2004) propose to report *three* measures: κ computed according to (Cohen 1960) (κ_{Co}), κ computed according to (Siegel and Castellan 1988) ($\kappa_{C\&S}$), and $PABAK$. κ_{Co} and $\kappa_{C\&S}$ only differ in the computation of $P(E)$. As (Di Eugenio and Glass 2004) point out, $\kappa_{C\&S}$ is not affected by bias (the value of κ_{Co} adjusted for bias turns out to be the same as $\kappa_{C\&S}$). In the present study κ_{Co} and $\kappa_{C\&S}$ are equal, showing that our data is not affected by the bias problem. I therefore report only two measures: κ (computed as $\kappa_{C\&S}$) and $PABAK$, which highlights and corrects the effect of prevalence.

formula to compute $PABAK$ given in (Byrt et al. 1993) is restricted to a dichotomous classification. I use a generalisation of that formula for n categories. A brief description of this generalisation is given in Appendix A.

2.4.3.1 The First Sample

The agreement on the coding of the first sample of sluices was moderate ($\kappa = 52$, $PAPAK = 56$)[12]. There were important differences amongst sluice classes: The lowest agreement was on the annotation for *why* ($\kappa = 29$, $PABAK = 36$), *how* ($\kappa = 32$, $PABAK = 40$) and *what* ($\kappa = 32$, $PABAK = 41$) , which suggests that these categories are highly ambiguous. Examination of the coincidence matrices shows that the largest confusions were between `reprise` and `clarification` in the case of *what*, and between `direct` and `reprise` for *why* and *how*. On the other hand, the agreement on classifying *who* was substantially higher ($\kappa = 71$, $PABAK = 77$), with some disagreements between `direct` and `reprise`. The effect of prevalence is clearly seen in the case of *which N*. Although in the annotation of *which N* the actual agreement was high ($P(A) = 84$), the fact that the coders used basically only two categories, `direct` and `reprise`, choosing one of them (`reprise`) more than 70% of the time, makes the percentage of expected agreement by chance very high ($P(E) = 65$), which in turn lowers the *kappa* coefficient ($\kappa = 55$). Prevalence of this kind is corrected by $PABAK$, which in this case is substantially higher than κ ($PABAK = 79$). See Table 2.11 for a summary of κ and $PABAK$ scores for the annotation of both samples.

	First Sample		**Second Sample**	
	kappa	$PABAK$	*kappa*	$PABAK$
General	52	56	61	67
what	32	41	39	50
why	29	36	52	69
who	71	77	62	84
where	38	54	35	49
when	44	57	62	70
which N	55	79	64	76
which	36	61	—	—
how	32	40	—	—

Table 2.11: Agreement amongst the 3 coders

[12] All values are shown as percentages.

2.4.3.2 The Second Sample

Agreement on the annotation of the 2nd sample was considerably higher, although still not entirely convincing ($\kappa = 61$, $PABAK = 67$). Overall agreement improved in all classes, except for *where* and *who*. The case of *who* provides another example of the prevalence problem. The actual agreement was high ($P(A) = 87$). However, the proportion of chance agreement was also high ($P(E) = 66$) due to the fact that most uses of this sluice were classified as `reprise`. This resulted in a *kappa* much lower than the actual agreement ($\kappa = 62$). Once this score is adjusted for prevalence, we obtain a significantly higher coefficient ($PABAK = 84$).

Agreement on *what* improved slightly ($\kappa = 39$, $PABAK = 50$), and it was substantially higher on *why* ($\kappa = 52$, $PABAK = 69$), *when* ($\kappa = 62$, $PABAK = 70$) and *which N* ($\kappa = 64$, $PABAK = 76$).

2.4.3.3 Discussion

Although the three coders may be considered *experts*, their training and familiarity with the data were not equal. This resulted in systematic differences in their annotations. Two of the coders (coder 1 and coder 2) had worked more extensively with the BNC dialogue transcripts and, crucially, with the definition of the categories to be applied. Leaving coder 3 out of the coder pool increases agreement very significantly (see Table 2.12). The agreement reached by the *more expert* pair of coders was high and stable and, I believe, provides a solid foundation for the current classification. The improvement of scores for the full triple of coders in the second sample also indicates that it is not difficult to increase annotation agreement by relatively light training of coders.

First Sample		**Second Sample**	
kappa	$PABAK$	*kappa*	$PABAK$
70	75	71	80

Table 2.12: Agreement between coder 1 and coder 2

One of the main points of disagreement encountered was the `unclear` category, which was used not only when the transcript was too poor to determine the resolution of the sluice, but also when the interpretation was ambiguous or the coder was uncertain about which category to choose. Disagreement was general amongst the three coders, suggesting that it might be useful to partition this category into `unclear` and `ambiguous`.

Regarding the prevalence problem and the computation of $PABAK$, one could argue that the fact that prevalence lowers *kappa* follows from the *kappa* definition, and that it is not unreasonable that the value of an agreement coefficient should be smaller when the instantiation of the categories is not uniform. However, like Di Eugenio and Glass (2004), I think that this effect undermines the credibility of *kappa* as a metric of agreement for a coding scheme. A distribution pattern that *kappa* treats as poor reliability can be the result of high prevalence and so reflect the nature of the data rather than a defect of the observation method. In our test samples prevalence is a symptom of the fact that different sluice types show clear preferences for particular readings, as we will see in the next section.

This is why I use an additional agreement metric that corrects for prevalence to complement *kappa*.

2.4.4 Results: Distribution Patterns

In this section I report the results obtained from the sluicing corpus study described in previous sections. The study shows that the distribution of readings is significantly different for each class of sluice.

The distribution of interpretations for each class of sluice is shown in Table 2.13. The distributions are presented as row counts and percentages of those instances where at least two annotators agree, labelled taking the majority class and leaving aside the `unclear` cases. Distributions are similar over both samples, suggesting that corpus size is large enough to permit the identification of repeatable patterns. For clarity's sake, in Table 2.13 the results of the two samples are conflated.

Sluice	Direct	Reprise	Clarification	Wh-anaphor
what	7 (9.60%)	17 (23.3%)	48 (65.7%)	1 (1.3%)
why	55 (68.7%)	24 (30.0%)	0 (0%)	1 (1.2%)
who	10 (13.0%)	65 (84.4%)	0 (0%)	2 (2.6%)
where	31 (34.4%)	56 (62.2%)	0 (0%)	3 (3.3%)
when	50 (63.3%)	27 (34.1%)	0 (0%)	2 (2,5%)
which	1 (8.3%)	11 (91.6%)	0 (0%)	0 (0%)
whichN	19 (21.1%)	71 (78.8%)	0 (0%)	0 (0%)
how	23 (79.3%)	3 (10.3%)	3 (10.3%)	0 (0%)

Table 2.13: Distribution patterns

Table 2.13 reveals significant correlations between sluice classes and preferred interpretations ($\chi^2 = 438.53$, $p \leq 0.001$). The most common interpretation for *what* is Clarification, making up more than 65%. *Why* sluices have a tendency to be Direct (68.7%). The sluices with the highest probability of being Reprise are *who* (84.4%), *which* (91.6), *which N* (78.8%) and *where* (62.2%). On the other hand, *when* (63.3%) and *how* (79.3%) have a clear preference for Direct interpretations.

As explained in Section 2.4.1, the samples used in the corpus study do not reflect the overall frequencies of sluice types found in the BNC. Now, in order to gain a complete perspective on sluice distribution in the full corpus, it is therefore appropriate to combine the percentages in Table 2.13 with the absolute number of sluices contained in the BNC. The number of estimated tokens is displayed in Table 2.14.

what$_{cla}$	2040	whichN$_{rep}$	135
why$_{dir}$	775	when$_{dir}$	90
what$_{rep}$	670	who$_{dir}$	70
who$_{rep}$	410	where$_{dir}$	70
why$_{rep}$	345	how$_{dir}$	45
where$_{rep}$	250	when$_{rep}$	35
what$_{dir}$	240	whichN$_{dir}$	24

Table 2.14: Sluice class frequency (estimated tokens)

For instance, the combination of Tables 2.13 and 2.14 allows us to see that although almost 70% of *why* sluices are Direct, the absolute number of *why* sluices that are Reprise exceeds the total number of *when* sluices by almost 3 to 1. Another interesting pattern revealed by this data is the low frequency of *when* sluices, particularly by comparison with what one might expect to be its close cousin—*where*. Indeed the Direct/Reprise splits are almost mirror images for *when* v. *where*. Explicating the distribution in Table 2.14 is important in order to be able to understand among other issues whether we would expect a similar distribution to occur in a Spanish or Mandarin dialogue corpus; similarly, whether one would expect this distribution to be replicated across different domains.

I will not attempt to provide an explanation for these patterns here. The reader is invited to check a sketch of such an explanation for some of the patterns exhibited in Table 2.14 in (Fernández et al. 2004).

2.5 Summary and Conclusions

In this chapter I have presented a comprehensive taxonomy of NSUs based on corpus work performed using part of the dialogue transcripts of the BNC. The proposed taxonomy of NSUs, which has a coverage of over 98%, distinguishes amongst 15 different classes that can be grouped into acknowledgements, questions, answers, extensions, and completions.

Several results emerge from this empirical study. To start with, the study shows that NSUs are an important phenomenon in dialogue, making up 9% of all utterances in the sub-corpus under investigation. The distribution of NSU classes shows that acknowledgments are the most common kind of NSUs, followed by answers, and next by questions. The study has also provided data concerning the distance between NSUs and their antecedent. Although in general the antecedent is the immediately preceding utterance, there are important differences between the various NSU classes, as well as between dialogue and multilogue. In summary, Short Answer is the only NSU class that exhibits significant long-distance effects, and these seem to be primarily an effect of multi-party interaction. In the last part of the chapter I have presented an additional corpus study focussed on sluices, which has shown that there are significant correlations between interpretation and sluice class.

The NSU and sluicing corpus studies presented in this chapter supply the data for the machine learning experiments undertaken to address the task of automatically disambiguating between NSU classes. These will be reported in Chapter 6. The following two chapters are concerned with the resolution of NSUs. First, I will review some existing approaches, and then in Chapter 4 present my own formalisation of the NSU classes, limiting myself to acknowledgments, questions and answers.

3 Some Previous Approaches

In this chapter I review some existing approaches to the resolution of NSUs. In the literature about ellipsis, it is common to distinguish between different approaches in terms of the level of information at which they assume that the resolution of ellipsis takes place, be it syntax, semantics or pragmatics. As NSUs have more often than not fallen under the denomination "ellipsis", I shall follow this line as well, starting with two accounts formulated exclusively in semantic terms. The first one is based on Higher Order Unification (Dalrymple et al. 1991, Pulman 1997); the second one is the logic-based approach of Paul Dekker (2003a,b). In Section 3.2, I will look into the minimalist account of Merchant (2004), whose proposal mostly relies on syntax. After this, in Section 3.3, I will turn to three approaches that make use of hybrid contextual information and grammatical *constructions*, namely Ginzburg and Sag (2001), Ginzburg and Cooper (2004), and Schlangen (2003). The three of them offer an account of NSUs couched in the formalism of Head-driven Phrase Structure Grammar (HPSG).

I end the current chapter describing my own previous approach in Section 3.4. This is built on the Dynamic Logic formalisation presented in (Fernández 2003a,b) and developed in collaboration with Matthew Purver (Purver and Fernández 2003, Fernández and Purver 2004). As we shall see, this work contains some of the underlying ideas at the core of the proposal that will be presented in the next chapter.

3.1 Semantic-based Approaches

I shall devote this first section to reviewing a couple of approaches that make use of purely semantic information.

3.1.1 Higher Order Unification

The first approach I will look into uses higher order unification (HOU) to resolve the meaning of elliptical expressions. The approach was not developed with the resolution of NSUs in mind, but it rather originated as an account of some intrasentential elliptical constructions, like most notably VP ellipsis. I will nevertheless review its main features here, because its elegant semantic-based formulation can easily be applied to some NSU classes like e.g. repeated acknowledgements and helpful rejections.

The most influential HOU-based account of ellipsis is perhaps that of Dalrymple et al. (1991). The main idea of this account is that the resolution of ellipsis amounts to recovering a property from context such that when it is applied to the meaning of the ellipsis *target*, it resolves the interpretation of the elliptical construction. Typically the property in question derives from a contextual antecedent or *source* clause. The recoverability of a suitable property is tied to the assumption that some sort of parallelism exists between source and target. The to-be-recovered property is then such that, when it is applied to the parallel element in the source, it is equivalent to the interpretation of the full source clause. An example will help clarify things here. Consider an instance of VP ellipsis like the following:

(47) Paul hates Yoko, and George does too.

In this example the source is the complete clause *Paul hates Yoko*, while the target is the elliptical sentence *George does too*, with *Paul* and *George* being parallel elements. Now, assuming that the interpretation of the source clause is $hates(paul, yoko)$ and the interpretation of the parallel element in this clause is $paul$, finding a suitable property P that will resolve the VP ellipsis in the target amounts to finding solutions to the following higher order equation, where P is a variable ranging over properties:

(48) $P(paul) = hates(paul, yoko)$

The obvious solution to this equation, which will make identical the two terms in the expression, is given in (49a).[1] The semantics of the VP ellipsis is then resolved by predicating P of the interpretation of the parallel element in the target, namely $george$, as shown in (49b). The semantics of the whole sentence is then as in (49c).

(49) a. $P = \lambda x.hates(x, yoko)$

b. $P(george) \mapsto \lambda x.hates(x, yoko)(george) \mapsto hates(george, yoko)$

[1]Other less obvious, and less interesting solutions like $\lambda y.(\lambda x.hates(x, yoko))(y)$ are also possible. I ignore them here, as they are all alternative variants of (49a).

c. $hates(paul, yoko) \wedge hates(george, yoko)$

As equations solved by HOU typically yield alternative solutions, the approach can account for ambiguities generated by certain elliptical constructions. In sentence (50a) for instance, the ambiguity between what has been called a strict reading (*Paul likes George's guitar*) and a sloppy reading (*Paul likes Paul's guitar*) of the target clause is explained in terms of the solutions (50c) and (50d), respectively, to the equation in (50b).

(50) a. George likes his guitar, and Paul does too

b. $P(george) = likes(george, guitar\_of(george))$

c. $P = \lambda x.(likes(x, guitar\_of(george))$
$P(paul) \mapsto \lambda x.(likes(x, guitar\_of(george))(paul) \mapsto likes(paul, guitar\_of(george))$

d. $P = \lambda x.(likes(x, guitar\_of(x))$
$P(paul) \mapsto \lambda x.(likes(x, guitar\_of(x))(paul) = likes(paul, guitar\_of(paul))$

This approach relies heavily in correctly establishing the parallelism between source and target. Only when the parallel elements are detected, it is possible to set up the relevant equation and apply its solutions to the target. Dalrymple et al. (1991) do not address this task however, claiming that it is advantageous to keep separate a theory of parallelism in order for it not to be restricted to purely syntactic techniques. Arguably this two-stage strategy boosts the flexibility of the approach in the sense that nothing prevents semantic and/or pragmatic information to be used in determining the relevant contextual property. Examples like the following (from Webber 1978), where the antecedent is not overtly expressed and has to be inferred pragmatically, could therefore potentially be treated.

(51) Irv and Mary want to dance together but Mary can't [*dance with Irv*] since her husband is here.

Despite the potential advantages of this distribution of labour, the fact is that the task of finding candidate parallel elements is not tackled by the authors.

Pulman (1997) offers a slightly different approach. Instead of considering the identification of parallelism and the resolution of ellipsis as two different stages, he regards the process of generating appropriate parallel elements as simultaneous to finding solutions to the higher order equation. He uses a functor $ellipsis$ whose meaning is given by the following conditional:[2]

[2]For simplicity's sake, I formulate the conditional equivalence using only one argument for the $ellipsis$ functor, although it can of course be generalised to multiple arguments without difficulty. See (Pulman 1997), p.17.

(52) $ellipsis(A) \Leftrightarrow P(A)$
if
$antecedent(C)$ and
$P(B) = C$ and
$parallel(A, B)$

According to this conditional equivalence, the elliptical target $ellipsis(A)$ will be interpreted as $P(A)$ provided that some contextual conditions are satisfied. The first condition has to do with the identification of an antecedent or source C. The second condition sets up the higher order equation, decomposing C into a property P and an element B. Finally, the third condition requires elements A in the target and B in the source to be parallel. If these conditions hold, an instantiation for P can be found, which will provide the interpretation of the ellipsis once it is substituted in on the right hand side of the equivalence. In the following example,[3] for instance, the availability of two candidate parallel elements to $bill$ ($john$ and $mary$) generates two solutions (53d) to the equation in (53c), leading to the two possible interpretations of the ellipsis given in (53e):

(53) a. John likes Mary and Bill too.
$ellipsis(bill) \Leftrightarrow P(bill)$

b. $antecedent(likes(john, mary))$

c. $P(B) = likes(john, mary)$

d. $parallel(bill, mary) \mapsto B = mary\ ,\ P = \lambda x.likes(x, mary)$
$parallel(bill, john) \mapsto B = john\ ,\ P = \lambda x.likes(john, x)$

e. $P(bill) \mapsto \lambda x.likes(x, mary)(bill) \mapsto likes(bill, mary)$
$P(bill) \mapsto \lambda x.likes(john, x)(bill) \mapsto likes(john, bill)$

Thus in this approach candidate parallel elements are generated as part of the solution to the equation. Again, however, the problem of deciding what elements count as parallel is not directly addressed. Parallel elements are stipulated rather than computed, which weakens the predictive power of the system.

The computation of parallelism is precisely the focus of attention of some follow-up approaches to HOU, like (Gardent and Kohlhase 1997, Gardent 1999). In order to compute parallel elements, Gardent and colleagues combine HOU with an abductive calculus. This computes the parallelism of two logical forms on the basis of the sortal properties of their subconstituents, defined in a domain-specific sort hierarchy or semantic ontology. I will not go into the details of this approach here, but just note that it does

[3] Adapted from (Pulman 1997) p.18.

seem to be able to predict semantic parallelism, as well as a related notion of contrast or contrastive parallelism.

Turning now to NSUs, it seems clear that HOU combined with a systematic theory of parallelism like the one just mentioned could be applied to some NSU classes, like Repeated Acknowledgements and Helpful Rejections, for instance.[4] Their resolution could be modelled as the recovery of a property from context whose application to the fragment resolves its clausal content. Consider the following example, where (54b) is a Repeated Acknowledgement and (54c) a Helpful Rejection:

(54) a. A: Mark plays football.

b. B: Football.

c. B: (No,) tennis.

To resolve these NSUs, first we need to compute the parallelism between the fragments and the elements in the source. If our semantic ontology is set up correctly, the shortest path in this ontology between *football* in (54b) and *tennis* in (54c) respectively, and the denotations of the constituents in the source will predict that, in both cases, the parallel element is *football*.[5] This will set up the higher order equation in (55a), with solution (55b), and corresponding resolutions in (55c):[6]

(55) a. $P(\mathit{football}) = \mathit{plays}(\mathit{mark}, \mathit{football})$

b. $P = \lambda x.\mathit{plays}(\mathit{mark}, x)$

c. $P(\mathit{football}) \mapsto \lambda x.\mathit{plays}(\mathit{mark}, x)(\mathit{football}) \mapsto \mathit{plays}(\mathit{mark}, \mathit{football})$
$P(\mathit{tennis}) \mapsto \lambda x.\mathit{plays}(\mathit{mark}, x)(\mathit{tennis}) \mapsto \mathit{plays}(\mathit{mark}, \mathit{tennis})$

HOU seems therefore able to assign plausible interpretations to the NSUs in (54). However it becomes less viable when it is faced with helpful rejections like the following:

(56) A: Does Mark play football?
B: Tennis.

[4]The reader will find an approach to non-elliptical corrections in terms of HOU in (Gardent et al. 1996).

[5]Typically, repeated acknowledgements would involve shorter paths (or no path at all if there is identity) than helpful rejections, which by definition involve a contrast and therefore more *distance* in a semantic sort hierarchy.

[6]Note that the interpretation assigned to (54c) should make clear that the NSU counts as a rejection, i.e. that the fragment is associated with some sort of contrastive focus. When an explicit negation is present, this is straightforward—"*No*" can be interpreted as $\neg C$, where C is instantiated to the source $\mathit{plays}(\mathit{mark}, \mathit{football})$.

It is obvious that the equational formulation of the approach requires that the type of the source clause be of the same type as the type of the target. However in the case of helpful rejections to polar questions like (56) the antecedent is of type question, while the NSU denotes a proposition. The same is true for short answers, and vice versa for direct sluices. Hence, this blocks an extension of the approach to this kind of NSUs, as the higher order equation cannot be set up.[7]

In the next section I briefly present a semantic approach that proposes a treatment of short answers.

3.1.2 A Compositional Semantics for Short Answers

In this section I will give a general impression of the logic-based approach to short answers presented in (Dekker 2003a,b).

In Dekker's work questions are built up from formulae ϕ by prefixing an operator $?$ followed by a sequence of variables $\vec{x}$. Intuitively, questions $?\vec{x}\phi$ can be seen as inquisitive expressions asking about those sequences of individuals such that, when they act as values for the variables $\vec{x}$ in ϕ, ϕ is true.

This is a representation of questions similar to the one adopted in the so-called *structured meaning* tradition (e.g. Krifka 1999, Ginzburg 2005) where questions are modelled as lambda abstracts. However Dekker stays in line with *propositional* approaches (Groenendijk and Stokhof 1997) in that he identifies the interpretation of a question with the set of its answers.

That the interpretation of questions is equivalent to answerhood can be seen more precisely in the semantics given for question expressions. He defines a satisfaction relation that holds between a model M, a variable assignment g and a sequence of answers (or individuals) α. For simplicity's sake, (57) shows the satisfaction definition for polar and unary *wh*-questions only, where **1** and **0** denote the truth values *true* and *false*, and Λ denotes the empty sequence

(57) **Satisfaction of polar and unary *wh*-questions**

- $M, g, \mathbf{1} \models ?\phi$ iff $(M, g, \Lambda \models \phi$ iff $M, g, \mathbf{0} \not\models ?\phi)$
- $M, g, \alpha \models ?x\phi$ iff $\alpha = \{d \mid M, g[x/d], \Lambda \models \phi\}$

The first clause in (57) roughly says that an answer to a polar question $?\phi$ is the truth value **1** (*true*) if ϕ is true. The second clause is related to *wh*-questions: α is the set of

[7]To avoid this problem, Pulman (1997) opts for associating interrogative clauses with propositional contents. Although, as suggested by the formal semantic literature on interrogatives (Groenendijk and Stokhof 1997), this move can be regarded as semantically problematic, it allows him to treat constructions like e.g. "*Mary came to the party, but with whom?*", where a proposition and a question are coordinated.

answers to a unary question $?x\phi$ if it contains individuals d such that if variable x in ϕ is substituted by d, ϕ is true.

Dekker employs techniques independently motivated by the phenomenon of topically restricted quantification to analyse short answers. Topically restricted quantification occurs when a quantified term is restricted by a sequence of contextually salient individuals. In the following example, for instance, "*all tennis players*" is restricted to the domain or topic of *Swedish tennis players*.

(58) Swedes are funny. All tennis players look like Björn Borg.

Dekker uses the notation $M, g, \alpha \models_{?\vec{x}\phi} \exists\vec{z}\psi$ to interpret topically restricted quantified expressions, where $?\vec{x}\phi$ is an n-place topic restricting the values of $\vec{z} = z_1 \cdots z_n$ in ψ. The same idea can be used to provide an interpretation of short answers, which is contextually restricted by a salient question. This is mediated by an operator ANS which behaves as follows:

(59) **Topic constituent answer to unary *wh*-questions**

$$M, g, \alpha \models_{?x\phi} ANS(\exists z Pz) \;\text{ iff }\; \alpha = \{d \mid M, g[d/z], \alpha \models Pz \wedge M, g[x/d], \Lambda \models \phi\}$$

This can be better understood with an example. Consider the dialogue in (60a). Assuming that A's question is represented as $?x Phone(x)$ and B's answer as $\exists z Student(z)$, the truth conditions of the short answer are as in (60b). In words, B's answer is true if there is an individual d such that when variables x and z (in the question and the answer, respectively) are substituted by d, both question and answer are true.

(60) a. A: Who phoned?
B: A student.

b. $M, g, \alpha \models_{?xPhone(x)} ANS(\exists z Student(z))$ iff

$$\alpha = \{d \mid M, g[z/d], \Lambda \models Student(z) \wedge M, g[x/d], \Lambda \models Phone(x)\}$$

Thus this approach allows for a compositional interpretation of short answers, which is usually difficult to obtain in propositional approaches to questions like that of Groenendijk and Stokhof (1997). However, it is not clear whether this account could be scaled up to other kinds of NSUs. I will not speculate here about how this could be done, although presumably this would involve defining additional operators on top of ANS.

In any case, the issue that has raised most of the objections to semantic approaches like this one and the HOU-based presented in the previous section is that they cannot account for the structural constraints that seem to govern elliptical constructions as well as NSUs, to which I have already refered to in the introduction. The following examples—(61b) from (Morgan 1973)—show that the well-formedness of the target or the NSU depends on the syntactic properties of the source or the antecedent.

(61) a. Klaus malte einen Menschen und Petra {einen Baum/*einem Baum}
`Klaus painted [a person]`$_{acc}$ `and Petra {[a tree]`$_{acc}$`/*[a tree]`$_{dat}$`}`

b. A: What does John think?
B: That Tricia has given birth to a 7-pound chin. /
*Tricia's given birth to a 7-pound chin. /
*For Tricia to have given birht to a 7-pound chin.

This kind of dependencies cannot be accounted for by HOU nor by a logic-based account which acts exclusively on denotations.[8] The presence of syntactic connectivity has motivated a body of approaches (like e.g. Morgan 1973, Chung et al. 1995, Merchant 1999) that essentially view the resolution of ellipsis as a process of recovering syntactic material from an antecedent clause. In the next section I review a recent account of some classes of NSUs that follows this line.

3.2 Ellipsis as Deletion: a Syntactic Account

Within the generative grammar tradition there is a line of research, which regarding NSUs can be traced back to Morgan (1973)'s analysis of short answers, that views elliptical constructions as involving unpronounced syntactic structure. I will concentrate here on one recent approach within this trend, namely that presented in (Merchant 2004).

Building on previous work (Merchant 2001) on the nature of embedded sluicing and ellipsis in general, Merchant (2004) offers an analysis of short answers from a Minimalist perspective. His account rests on two main ideas: (i) that non-sentential utterances involve ellipsis, i.e., according to the author, deletion of syntactic structure; and (ii) that they involve movement to a peripheral position in the clause's syntactic representation.

The first of these points essentially corresponds to the *ellipsis generation* approach advocated for in (Morgan 1973). This is based on the observation that often elliptical constructions show the same grammatical connectivity as their non-elliptical counterparts. Merchant provides extensive cross-linguistic data to support this. The examples are all in the lines of the phenomena already observed by Morgan (1973, 1989), like case-matching—where the fragment bears the same morphological case that it would bear in a sentential answer, as in (62a) for Russian—and binding and correference effects—as shown in (62b) for the distribution of reflexives, for instance.[9]

[8]Note, however, that Pulman's predicate *parallel* could in principle be used to encode dependencies of varied nature.

[9]The examples are adapted from (Merchant 2004).

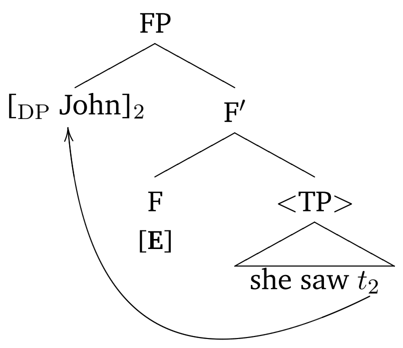
(62) a. A: Komu pomogla Anna? B: Ivanu/*Ivan/*Ivana.
`A: who`$_{dat}$ `helped Anna?` `B: Ivan`$_{dat}$`/*Ivan`$_{nom}$`/*Ivan`$_{acc}$
A: Who did Anna help? *B: Ivan.*

b. A: Who does John$_1$ like?
B: Himself/*Him$_1$.
(*John$_1$ likes himself/*him$_1$.*)

These phenomena seem to support the idea that the same mechanisms operate uniformly in both NSUs and complete sentences, and are taken as evidence that the NSU is generated as a full clause, which is then subjected to a deletion operation. However, as pointed out by Ginzburg (1999), Ginzburg and Sag (2001) and others, there are important divergences between NSUs and their sentential correlates that would invalidate this claim:[10]

(63) A: Who appeared to be the cause of John and Mary's problems?
B: Each other.
**Each other appeared to be the cause of John and Mary's problems.*

Merchant assumes that elliptical structures bear a feature E, which is responsible for all those properties that distinguish them from their non-elliptical counterparts. The idea is that syntactic heads bearing E license the ellipsis of their complements (provided that some feature-feature matching requirements are met). A scheme of his proposal is given in (64).

(64)


The feature E informs the phonological component of the grammar that its complement—which in (64) corresponds to the clause [She saw t_2]—should go unpronounced. As for the semantics, the feature E demands that the elided proposition bears

[10]The data in this respect is varied and often seems to point in different directions. Nevertheless, I take (63)—taken from (Ginzburg and Sag 2001)—to be a clear example of no correlation between the NSU form and its full-sentence counterpart. For some counterexamples I refer the reader to Merchant (2004).

the property of *e-givenness*, which seems to boil down to the requirement that the elided material must have an appropriate antecedent.

According to the scheme in (64), the fragment moves to a specifier position of a functional projection[11] prior to the deletion. This movement is in part introduced to explain certain phenomena that were puzzling for earlier deletion-based approaches like that of Morgan (1973). One such phenomenon is, for instance, the presence of a complementizer in some short answers, which is ungrammatical in the presumed underlying full sentential structure, as shown in (65a). This is a problem for theories that take elliptical constructions and full sentences as being generated by the same mechanisms. However, as frequently noted in previous literature (e.g. Stowell 1981, Bresnan 1994), as well as by Merchant himself, if the complement does not occupy its canonical position like in (65b), the sentential version containing the complementizer is felicitous.

(65) a. A: What are you ashamed of?
B: *(That) I ignored you.
**I'm ashamed of that I ignored you.*

b. *(That) I ignored you, I'm ashamed of.

Merchant takes this parallelism between focused phrases (or clauses) and short answers as ratifying his approach to NSUs as involving movement to a peripheral position. The reader has probably noted however that this argument relies on the theory-specific assumption of generative grammar according to which so-called dislocated phrases like that in (65b) involve movement from their original syntactic position. Of course other analysis of these phrases are possible in movement-free linguistic theories based on surface syntax,[12] and hence this cannot be considered as evidence of movement for NSUs. What these examples do show however is that—not surprisingly—a parallelism should be drawn between short answers and focused constructions, and that it is likely that the same mechanisms that explain the complementizer pattern in one case do so as well for the other one.[13] In any case, movement does not seem to solve the problems posed by examples like (63) above.

Leaving movement aside, it is worth pointing out that Merchant's account differs from previous approaches that view ellipsis as a reconstruction/deletion procedure whereby contextually available syntactic structure is copied from an antecedent clause (like e.g. Chung et al. 1995). In contrast, Merchant's account is somehow more semantic (or pragmatic) in nature. The property of *e-giveness* acts as an anaphoric or presuppositional

[11]Merchant is not precise about this and simply suggests that the functional projection in question could perhaps be a so-called Focus Phrase.

[12]See e.g. the analysis of unbounded dependencies in Chapter 5 of (Ginzburg and Sag 2001).

[13]See (Schlangen 2003) §5.5.1.2–5.5.1.3 for some discussion of this issue.

device, which licenses a deleted proposition if it can be semantically recovered from context.[14] This allows the author to treat fragments which lack a linguistic antecedent, similar in nature to example (51) in the previous section.

(66) a. [*Abby and Ben are at a party. Abby sees an unfamiliar man with Beth, a mutual friend of theirs, and turns to Ben with a puzzled look in her face. Ben says:*]

Some guy she met at the park.

b. [$_{\mathrm{FP}}$ some guy she met at the park$_1$ $<$[$_{\mathrm{TP}}$ he's t_1]$>$]

The structure proposed for the fragment in (66a) is that in (66b), where a demonstrative pronoun and the copula are elided. The assumption is that in this case the context is rich enough to license the *e-giveness* property, as it contributes suitable non-linguistic referents for the anaphoric expressions involved in the elided proposition—in the example below, a salient guy as antecedent of the pronoun *he*, as well as some implicit question that would make the predicate *be* salient. Antecedents are therefore semantic rather than syntactic.

Summarising, in Merchant's system NSUs are not ratified as grammatical constructions per se, but are seen as deviations of full sentences. The author postulates that NSUs are generated as full syntactic sentences, which are deleted after the fragment has moved to a peripheral position. This allows him to account for syntactic connectivity, although it leaves unexplained some divergences between NSUs and their sentential counterparts. Resolution in this system is however essentially semantic, as the entities that act as potential antecedents (and that are denoted by the elided expressions) do not necessarily have to be syntactically present in context. Yet, according to his approach, the context dependent material of an NSU must be fully generated at the syntax/semantics interface. The issue then arises as to how the syntactic structures subjected to deletion are determined. To restrict the space of generation of these syntactic structures Merchant proposes what he calls a "limited ellipsis" analysis. This assumes that only demonstrative pronouns or expletives and the existential predicate can be part of the elided structure of NSUs, as they can generally be taken for granted in most contexts.

This may be true for examples similar to those in (66) above, but it is certainly not enough to account for more complex NSUs, as demonstrated by the examples in (67a) from Ginzburg and Sag (2001) and (67b) from Schlangen (2003), which will be considered in more detail in subsequent sections. In these cases, the linguistic form of the material needed for resolution is not present in the context, and a "limited ellipsis" analysis would clearly be insufficient.

[14]The same notion of *e-giveness* is employed by Merchant (2001) to account for sluices. For some comments on the merits and pitfalls of his analysis see (Ginzburg and Sag 2001) §8.1.7.

(67) a. A: Go home Billie!
B: Why? (= *Why are you ordering me to go?*)

b. A: Why did Peter leave so early?
B: Exams. (= *He has/has to mark/... exams.*)

3.3 Constructionist Approaches

In contrast to sententialist analysis of NSUs exemplified here by Merchant's work, the approaches that will be presented in this section—namely (Ginzburg and Sag 2001, Ginzburg and Cooper 2001, 2004) and (Schlangen 2003)—see NSUs not as deviations of canonical sentences, but as fully grammatical structures that are a proper part of the grammar. Another point shared by these approaches is that they are all couched in the formalism of HPSG. Grammatical analysis in HPSG are characterised by avoiding the need of empty elements and dispensing with movement and deletion operations. In the aspect that interests us here, this is achieved by the use of *constructions* (Sag 1997), phrasal types that may introduce material that is not strictly present in the lexical elements they include. This somehow unorthodox mapping between syntax and semantics has been proved to be useful to analyse different grammatical phenomena like relative and interrogative clauses (Sag 1997, Ginzburg and Sag 2001). In the same line, NSUs can be analysed as sentential constructions whose only constituent is a non-sentential phrase.

Instead of following the chronological order in which the approaches reviewed in this section were developed, I will start by reviewing Schlangen's work, which is partially inspired by (Ginzburg and Sag 2001). I will then turn to the proposals of Ginzburg and collaborators on which my own work is based.

3.3.1 Fragments and SDRT

I have already referred to Schlangen's (2003) work on NSUs in the previous chapter, where I discussed the taxonomy of fragments he adopts. In this section I will concentrate on the strategies employed by the author for representing and resolving NSUs, which combine an underspecified semantics of fragments with SDRT (Asher 1993, Asher and Lascarides 2003).

Inspired by Ginzburg and Sag (2001), Schlangen adopts a constructionist approach to the syntax of NSUs, revamping fragments to the status of sentences by means of HPSG constructions. I forgo giving the HPSG matrices here; it will suffice to say that his constructions encode the following informal rule:[15]

[15]The rule is taken from (Schlangen 2003) §7.3, p. 171.

(68) $S[\mathit{frag}] \rightarrow (ADV)XP$

This rule takes possibly modified phrases ($(ADV)XP$) to the level of sentences. Following Ginzburg and Sag (2001), this is achieved by specifying the root sign of an NSU construction ($S[\mathit{frag}]$ in the informal rule above) as having the same general semantic type of clauses, namely *message*, as well as the same syntactic features of a sentence, like the same categorical type of verbs.

The semantic representations built up by these syntactic constructions are underspecified descriptions of logical forms. Because Schlangen uses the English Resource Grammar (Flickinger et al. 2000), these underspecified representations are encoded in the formalism of Minimal Recursion Semantics (MRS) (Copestake et al. 1999), which is in fact used as a short-hand for a more detailed underspecification language employed in (Asher and Lascarides 2003). We do not need to go into the details of these formalisms, as they are not significant for our purposes here. More important is the general idea behind the use of underspecification for NSUs' semantics. This is indeed quite simple and elegant: NSUs are seen as denoting an *unknown* (i.e. underspecified) message or relation (corresponding to a proposition, a question or a request), which is crucially restricted to involve in some way or other the entity denoted by the phrase used in the NSU. For example, a declarative NSU consisting of the NP *"Peter"* will denote a proposition (and therefore an eventuality in a Neo-Davidsonian approach), which involves the entity denoted by the phrase, namely 'Peter' in this case. This can be formulated as follows, where P denotes an unknown relation in which x (the denotation of "*Peter*") plays some role:

(69) $\exists e \exists x \; P(e,x) \wedge \mathit{named}(x, \mathrm{Peter}) \wedge P = ?$

This underspecified representation describes a potentially infinite number of logical forms, one for each proposition that the NSU can denote. The role of the resolution procedure will then be to resolve this underspecification by choosing the pragmatically preferred reading given the context. And here is where SDRT comes into play.

Originated as an extension of DRT (Kamp and Reyle 1993), SDRT focuses on how DRSs corresponding to semantic representation of sentences are combined into larger semantic representations of discourses. While in classical DRT this is assumed to be a simple operation of merging DRSs, in SDRT this is done by introducing rhetorical relations, which act as the means to connect new sentences with the previous discourse. These rhetorical relations are computed by means of a non-monotonic logic that, besides lexical semantics and other sources of information, critically uses defeasible pragmatic knowledge. The pragmatically preferred interpretation of a sentence in a given context is computed by means of a constraint *update* that has access to different kinds of information and that, given a new sentence β, describes all possible ways in which β can connect to the context. This is then filtered by a principle called *Maximise Discourse Coherence*, which is formulated as a coherence ordering that e.g. prefers a maximal number of rhetorical relations between two items.

As already explained in Section 2.2 of the previous chapter, Schlangen distinguishes between *resolution-via-identity* and *resolution-via-inference* NSUs. The *unknown* predicate of a *res-via-id* NSU is resolved to a relation explicitly present in its antecedent. If the fragment plays an argumental role in this relation, some syntactic requirements must be met.

In order to account for the syntactic connectivity exhibited by the *res-via-id* NSUs, Schlangen modifies the constraint *update* so that it is sensitive to some syntactic information. This is done by adding a relation *G-parallel* (for *generalised parallel*),[16] which holds when a fragment is semantically and structurally very similar to its antecedent (i.e. it is a *resolution-via-identity* fragment). The idea is then that *G-parallel* triggers a syntactic constraint that must be satisfied as well. This syntactic constraint is formulated as follows:

(70) `Syntactic Constraint on G-Parallel`
syn-constr$(\alpha, \beta) \leftrightarrow$

a) There is a partial isomorphism between the DRS-structure of K_α and that of K_β, and

b) no argument of a predicate in K_β has a syntactic category-specification different from what the *arg-st*-specification of that predicate demands

Here α is the potential antecedent of an NSU β, while K_α and K_β are the DRSs representing the content of the antecedent and the NSU, respectively. Clause (a) thus has to do with the required similarity between NSU and antecedent. Clause (b), on the other hand, makes sure that the argumental specifications of all predicates in the resolved content of the fragment are met. From the infinite set of resolutions described by the underspecified meaning of the fragment, those that trigger *G-parallel* will be validated only if *syn-constr* holds. This of course requires that some syntactic specifications are *visible* or accessible to the logic computing the rhetorical relations, and this is what Schlangen assumes is the case, thereby modifying standard versions of SDRT and blurring its modularity.

With this syntactic constraint in place, Schlangen can account for the connectivity effects of (61) as well as for the optionality of prepositions in short answers like (71). If the NSU is a PP, this is taken as satisfying the subcategorisation requirements of the verb *"rely"*, while if it is an NP it satisfies the requirements imposed by the preposition *"on"*.

[16]*G-parallel* is a variant of the SDRT relation *Parallel*, which is inferred when its arguments are semantically very close.

(71) A: Who can we rely on?
B: (On) Sandy.

Thus in (71) the fact that *G-parallel* holds and that *syn-constr* is satisfied will trigger the rhetorical relation *Question Answer Pair* (*QAP*). Sometimes however, there are no well-formed representations that satisfy *G-parallel*, like in (72), where the semantic type of the answer does not match that of the *wh*-phrase. In such cases inference is needed.

(72) A: Why did Peter leave so early?
B: Exams.

As the name indicates, resolving *resolution-via-inference* fragments involves inferring some material not explicitly present in the antecedent utterance. Indeed, what licenses an inference to *QAP* in (72) is a rather complicated inferential process: it involves assuming that the representation of A's question includes a presupposition p and a question *'Why p?'*, which are connected by the relation *Explanation*$_q$,[17] and inferring that the fragment is connected to p by *Explanation*. If these inferences are validated, then it is possible to infer that the NSU is an answer to *'Why p?'*. Although Schlangen is not explicit on this point, presumably in these cases the *unknown* relation is left underspecified.

One of the problems of this approach is that, as it is formulated, it can only account for the syntactic properties of NSUs that derive from the subcategorisation requirements of the resolved *unknown* predicate, whenever this is also present in the antecedent clause—i.e. whenever the NSUs is *res-via-id*. Consider, however, the following example, where the interpretation of the NSU *"Her?"* can be roughly paraphrased as one of the options in (73b–73d).

(73) a. A: Leo saw her.
B: Her? / #She?

b. *Who are you referring to with your utterance "her"?*

c. *Did you just say "her"?*

d. *You saw HER, of all people?*

Presumably *G-parallel* would not hold here, especially in cases (73b) and (73c), where the predicate that resolves the *unknown* relation is not explicitly present in the antecedent. Hence these NSUs would not fall under the denomination *resolution-via-identity* and *syn-constr* would not apply. Even if it did, however, the form of the NSU

[17]The relation *Explanation*$_q$ holds between a proposition p and a question q if all answers to q are explanations of p.

in (73) cannot be explicated in terms of the argumental role it plays within the resolved relation. As pointed out by Ginzburg and Cooper (2004), NSUs like (73a) with readings (73b) and (73c) are *utterance anaphoric*—they refer to a constituent whose content has not been understood to ask about that content. And they do so by means of a phrase that must be identical (categorically and phonologically) to the to-be-clarified constituent.

In order to explain the constraints on utterance anaphoric NSUs within Schlangen's framework one could either extend *G-parallel* and the notion of *res-via-id* NSUs, or alternatively introduce structural constraints for *res-via-inf* NSUs. Any of these ways forward, however would obscure—if not invalidate—the distinction between *res-via-id* and *res-via-inf* NSUs, and certainly blur even further the boundaries of the different modules of the theory.

3.3.2 Ginzburg and Sag (2001)

The approach of Ginzburg and Sag consists in analysing NSUs by means of different construction types, whose sentential content is derived not only from their constituent lexical items, but via combination with context. The basic structure in HPSG is the *sign*, a combination of semantic, syntactic and contextual information modelled as a feature structure and represented as an attribute value matrix. The CONTEXT attribute is modified to integrate dialogical notions like *questions under discussion* (QUD). In particular, the authors assume that CONTEXT contains two additional features:

- Maximal Question Under Discussion MAX-QUD, whose value is a semantic object of type *question* and represents the most prominent issue currently under discussion, and
- Salient Utterance SAL-UTT, which takes as its value sets of elements of type *sign* and represents a focal constituent in the utterance that has given rise to MAX-QUD.

The instantiation of MAX-QUD follows the basic ideas of Ginzburg's theory of context as presented for instance in (Ginzburg 1996), where both asking a question q and asserting a proposition p raise a question under discussion. In the former case this is just q, while in the latter it is the polar question *whether p*. Ginzburg and Sag view questions as λ-abstracts over propositions. They simulate this in HPSG by a type *question* which includes two features, PARAM(ETER)S and PROP(OSITION), where the value of PARAMS represents those elements that are abstracted away from the propositional core, i.e. those variables bound by the λ operator. For instance, a *wh*-question like *"Who left?"*, which could be formalised as the λ-abstract $\lambda x.Leave(x)$, in Ginzburg and Sag's HPSG would be represented by the structure in (74). Polar questions on the other hand are seen as empty abstracts, which are represented with an empty value for the feature PARAMS.

(74) $$\left[\begin{array}{ll} \textit{question} & \\ \text{PARAMS} & \left\{ \left[\begin{array}{ll} \text{INDEX} & \boxed{1} \\ \text{REST} & \left\{ \textit{person}(\boxed{1}) \right\} \end{array}\right] \right\} \\ \text{PROP} & \left[\begin{array}{ll} \textit{proposition} & \\ \text{SIT} & s \\ \text{SOA} & \left[\begin{array}{ll} \text{QUANTS} & \langle\,\rangle \\ \text{NUCL} & \left[\begin{array}{ll} \textit{leave-rel} & \\ \text{LEAVER} & \boxed{1} \end{array}\right] \end{array}\right] \end{array}\right] \end{array}\right]$$

The feature SAL-UTT has a function akin to the *parallel element* in the HOU approaches we have seen in Section 3.1.1. But, since SAL-UTT is of type *sign*, it offers access to information beyond semantics, thereby making it easy to define constraints involving different kinds of linguistic material. This is used by Ginzburg and Sag to account for syntactic connectivity effects that could not be captured by a purely semantic approach like HOU. Moreover the authors offer a recipe for computing the value of SAL-UTT: this corresponds to the (sub)utterance associated with the role bearing widest scope within MAX-QUD.

Let us see how this works in more detail. The various fragments Ginzburg and Sag analyse[18] are subtypes of the construction type *headed-fragment-phrase*, governed by the constraint in (75).

(75) *hd-frag-ph*:

$$\left[\begin{array}{ll} \text{HEAD} & \left[\begin{array}{l} v \\ \text{VFORM}\ \textit{fin} \end{array}\right] \\ \text{CTXT}|\text{SAL-UTT} & \left\{ \left[\begin{array}{ll} \text{CAT} & \boxed{1} \\ \text{CONT}|\text{INDEX} & \boxed{2} \end{array}\right] \right\} \\ \text{HD-DTR} & \left[\begin{array}{ll} \text{CAT} & \boxed{1} \left[\text{HEAD}\ \ \textit{nominal}\right] \\ \text{CONT}|\text{INDEX} & \boxed{2} \end{array}\right] \end{array}\right]$$

This constraint has two significant effects. First, it takes a nominal daughter[19] to a clausal phrase. This is done by constraining the mother of the phrase to be of the same category

[18] I concentrate on their analysis of short answers and direct and reprise sluices here, and leave clarification fragments for Section 3.3.3.

[19] Note that they limit their approach to NPs and case-marking PPs, i.e. phrases whose head is of type *nominal*.

as finite verbs, which will allow such phrases to serve as stand-alone clauses, i.e. to be embedded as the daughter of *root*-clauses,[20] and also to function as the complement of verbs that select for finite sentential clauses. Second, the constraint ensures that the category of the head daughter is identical to that specified by the contextually provided SAL-UTT, and coindexes the two of them.

Short answers are analysed by means of the type *declarative-fragment-clause* (*decl-frag-cl*), a subtype of *hd-frag-ph*. In addition to inheriting the information in (75), this latter type is governed by the constraint in (76):

(76) *decl-frag-cl*:

$$
\begin{bmatrix}
\text{HEAD} & \begin{bmatrix}\text{IC} & +\end{bmatrix} \\
\text{CONT} & \begin{bmatrix}\textit{proposition} & \\ \text{SIT} & \boxed{2} \\ \text{SOA} & \begin{bmatrix}\text{QUANTS} & \texttt{order}(\boxed{\Sigma_3}) \oplus \boxed{A} \\ \text{NUCL} & \boxed{5}\end{bmatrix}\end{bmatrix} \\
\text{STORE} & \boxed{\Sigma_1} \\
\text{MAX-QUD} & \begin{bmatrix}\textit{question} & \\ \text{PARAMS} & \textit{neset} \\ \text{PROP} & \begin{bmatrix}\textit{proposition} & \\ \text{SIT} & \boxed{2} \\ \text{SOA} & \begin{bmatrix}\text{QUANTS} & \boxed{A} \\ \text{NUCL} & \boxed{5}\end{bmatrix}\end{bmatrix}\end{bmatrix} \\
\text{HD-DTR} & \begin{bmatrix}\text{STORE} & \boxed{\Sigma_3} \cup \boxed{\Sigma_1}\ \textit{set(param)}\end{bmatrix}
\end{bmatrix}
$$

The content of this phrasal type is a proposition, which is constructed for the most part from MAX-QUD. This provides the concerned situation and the nucleus, whereas if the fragment is (or contains) a quantifier, that quantifier must outscope any quantifiers already present in the contextually salient question.

The combined effect of (75) and (76) is better appreciated with an example. (77) shows a (somehow simplified) representation of the NSU *"John"* as a short answer to the question *"Who left?"*. The content of this question provides the value of MAX-QUD, which is the contextual source from which the propositional content of the short answer is derived. The index of the NP *"John"* is unified with that of the SAL-UTT, which corresponds

[20]The type *root-clause* corresponds to the start symbol of Ginzburg and Sag's grammar. This is the type of utterances and its content is an *illocutionary proposition*. More will be said about this in Section 3.3.3.

to the sign representing the *wh*-phrase *"who"*. Note that the content of this sign is in the PARAMS set of MAX-QUD, which in turn identifies *"John"*'s index with the LEAVER role in MAX-QUD. Finally the category of the NP *"John"* is identified with the CAT value of the SAL-UTT. As in this case the sign in SAL-UTT (i.e. the *wh*-phrase *"who"*) is an argument of the verb *"leave"*, its category is constrained by the subcategorisation requirements of the predicate in question. Therefore the categorical identity between SAL-UTT and the head daughter NP enforces that the latter also meets these requirements.

This explains why the syntactic form of the fragment NP is similar to the the form it would exhibit in a full sentence, without need of postulating any underlying syntactic structure. Furthermore, the fact that SAL-UTT is a sign, opens the door to further parallelisms that go beyond the subcategorisation requirements of predicates, thus offering a more flexible account than that of Schlangen (2003).

(77)
$$\begin{bmatrix}
\text{HEAD} & \begin{bmatrix}\text{IC} & + \\ v & \end{bmatrix} \\
\text{CONT} & \begin{bmatrix} \textit{proposition} & \\ \text{SIT} & \boxed{2} \\ \text{SOA} & \begin{bmatrix}\text{QUANTS} & \langle\ \rangle \\ \text{NUCL} & \boxed{3}\begin{bmatrix}\textit{leave-rel} & \\ \text{LEAVER} & \boxed{1}\end{bmatrix}\end{bmatrix}\end{bmatrix} \\
\text{STORE} & \{\} \\
\text{MAX-QUD} & \begin{bmatrix} \textit{question} & \\ \text{PARAMS} & \left\{\begin{bmatrix}\text{INDEX} & \boxed{1} \\ \text{REST} & \left\{\textit{person}(\boxed{1})\right\}\end{bmatrix}\right\} \\ \text{PROP} & \begin{bmatrix}\textit{proposition} & \\ \text{SIT} & \boxed{2} \\ \text{SOA} & \begin{bmatrix}\text{QUANTS} & \langle\ \rangle \\ \text{NUCL} & \boxed{3}\end{bmatrix}\end{bmatrix}\end{bmatrix} \\
\text{SAL-UTT} & \left\{\begin{bmatrix}\text{CAT} & \boxed{4} \\ \text{CONT}|\text{INDEX} & \boxed{1}\end{bmatrix}\right\} \\
\text{HD-DTR} & \begin{bmatrix}\text{PHON} & \texttt{john} \\ \text{CAT} & \boxed{4}\ \text{NP} \\ \text{CONT}|\text{INDEX} & \boxed{1}\end{bmatrix}
\end{bmatrix}$$

Similarly to short answers, to deal with direct sluices Ginzburg and Sag posit an additional subtype of *hd-frag-ph* called *sluiced-interrogative-clause*, that like *decl-frag-cl* has its content partially determined by the dialogue context. They argue that the context for sluicing involves QUD-maximality of a polar question *whether p*, where *p* is a quantified proposition.

The content of the sluice is resolved to a *wh*-question obtained by λ-abstracting over a quantified element of MAX-QUD. The SAL-UTT is in this case derived from the sign whose content had the widest scope in MAX-QUD. Again, coindexation of index and category between the SAL-UTT and the *wh*-phrase head-daughter account for the semantic resolution and the syntactic matching conditions. I forgo showing the constraint for the type *slu-int-cl* and instead give an example—(78) shows the representation of the direct sluice *"Who?"* as a follow-up to the utterance *"A student phoned"*.

(78)
$$\left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{slu-int-cl}}\\
\text{CONT} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{question}}\\
\text{PARAMS} & \boxed{2}\left\{\left[\begin{array}{ll}\text{INDEX} & \boxed{1}\\ \text{REST} & \left\{\textit{person}(\boxed{1})\right\}\end{array}\right]\right\}\\
\text{PROP} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{proposition}}\\
\text{SIT} & \boxed{4}\\
\text{SOA} & \left[\begin{array}{ll}\text{QUANTS} & \langle\,\rangle\\ \text{NUCL} & \boxed{3}\left[\begin{array}{ll}\multicolumn{2}{l}{\textit{phone-rel}}\\ \text{PHONER} & \boxed{1}\end{array}\right]\end{array}\right]
\end{array}\right]
\end{array}\right]\\
\text{STORE} & \{\}\\
\text{MAX-QUD} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{question}}\\
\text{PARAMS} & \{\}\\
\text{PROP} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{proposition}}\\
\text{SIT} & \boxed{4}\\
\text{SOA} & \left[\begin{array}{ll}\text{QUANTS} & \left\langle\left[\begin{array}{ll}\multicolumn{2}{l}{\textit{some-rel}}\\ \text{INDEX} & \boxed{1}\\ \text{REST} & \left\{\textit{student}(\boxed{1})\right\}\end{array}\right]\right\rangle\\ \text{NUCL} & \boxed{3}\end{array}\right]
\end{array}\right]
\end{array}\right]\\
\text{SAL-UTT} & \left\{\left[\begin{array}{ll}\text{CAT} & \boxed{5}\\ \text{CONT}|\text{INDEX} & \boxed{1}\end{array}\right]\right\}\\
\text{HD-DTR} & \left[\begin{array}{ll}\text{PHON} & \texttt{who}\\ \text{CAT} & \boxed{5}\,\text{NP}\\ \text{CONT}|\text{INDEX} & \boxed{1}\\ \text{STORE} & \boxed{2}\end{array}\right]
\end{array}\right]$$

3.3.3 Grounding and Clarification

Ginzburg and Cooper (2004) follow up the approach offered by Ginzburg and Sag (2001) and extend it to account for clarification NSUs—those NSUs classified as CE in the taxonomy of Chapter 2, like the reprise sluice in (79b) and the clarification fragment in (79c). As seen in the previous chapter, these NSUs are related to the grounding process inherent to dialogue. A sound analysis of clarification NSUs therefore must be paired with a model of the grounding process itself—a model that tell us what this process involves and, more importantly in this case, what happens when it fails.

(79) a. A: Did Bo leave?

b. B: WHO?

c. B: Bo?

The view of grounding that underpins Ginzburg and Cooper (2004) approach to clarification is based on the classic notion of meaning as *a function from context to content* introduced by Montague (1974) and Kaplan (1989). Ginzburg and Cooper (2004) and Purver (2004b) implement this idea by representing utterances as λ-abstracts over a set of contextual parameters. The set of contextual parameters of an utterance gives a characterisation of the context-dependent and hence potentially problematic elements of the utterance content. The grounding process of an utterance then involves instantiating its contextually dependent parameters in the current context in order to derive a fully-fleshed content. According to this model, clarification questions are triggered by the inability to ground one or more contextual parameters in the current context.

This can be formulated as a set of instructions—the Utterance Processing Protocol (UPP)—for a dialogue participant to update her information state, leading either to grounding or clarification.

(80) **Utterance Processing Protocol (simplified version)**

```
1. Add U to PENDING
2. Attempt to ground U by instantiating each contextual parameter $i$ in U
3. If successful:
    (a) remove U from PENDING;
    (b) add $content$(U) to LATEST-MOVE;
    (c) if $content$(U) = $assert(p)$: push $whether(p)$ onto QUD
    (d) if $content$(U) = $ask(q)$: push $q$ onto QUD
4. Else: form clarification question about $i$
```

According to the simple version of the protocol shown in (80), utterances are first added to a field PENDING for the grounding process. Failure to identify the relevant contextual parameters in context leads to the formation of a clarification question relevant to that parameter. Success leads to removal from PENDING and addition to LATEST-MOVE. In the case of assertions, a question related to the asserted proposition is also added to QUD; in the case of *ask* moves, the asked question is added to QUD.

To represent utterances in a way which is compatible with this conception of grounding and clarification, Ginzburg and Cooper (2004) adopt the HPSG grammar of Ginzburg and Sag (2001) introducing two basic modifications. Firstly, they substitute the feature C-INDICES (contextual indices), which was standardly used to include information about speaker, addressee and utterance time, by the feature C-PARAMS (contextual parameters), which encodes the entire inventory of contextual parameters of an utterance. Secondly, given that it is perfectly possible to ask for clarification of any expression of a given utterance, in order to have access to all phrasal constituents they introduce a new feature CONSTITS (constituents), whose value is the set of all constituents (immediate and embedded) of a given sign.

In Ginzburg and Sag's grammar, utterances are analysed by the type *root-clause*, which includes the direct illocutionary force of an utterance as part of its semantic representation. Root clauses denote illocutionary propositions, whose NUCLEUS is an illocutionary relation (*assert, ask, order*,. . .) that holds between a speaker, an addressee and a message (a *proposition*, a *question*, an *outcome*,. . .). Illocutionary relations constrain the type of message they take as argument, with *ask* relations taking *questions*, *assert* relations taking *propositions*, etc. The inclusion of illocutionary force into the grammatical representation of utterance is particularly useful to analyse CE as the content of these NSUs often involves the illocutionary force of the utterance being clarified. For instance, one of the readings of the clarification question *"Bo?"* in (79c) could be paraphrased as *"Are you asking if BO (of all people) left?"*.

Given this and the modifications introduced by Ginzburg and Cooper, the utterance *"Did Bo leave?"* would be represented by the sign in (81). Here the relevant contextual parameters include the speaker and the addressee, the utterance time and a referent for the name *"Bo"*.

(81)
$$\begin{bmatrix}
\textit{root-cl} & \\
\text{PHON} & \text{did bo leave} \\
\text{CAT} & \text{V[+fin]} \\
\text{C-PARAMS} & \left\{ \begin{array}{l}
\begin{bmatrix} \text{INDEX} & \text{i} \\ \text{REST} & \left\{ \textit{spkr}(\text{i}) \right\} \end{bmatrix},
\begin{bmatrix} \text{INDEX} & \text{j} \\ \text{REST} & \left\{ \textit{addr}(\text{j}) \right\} \end{bmatrix}, \\
\begin{bmatrix} \text{INDEX} & \text{k} \\ \text{REST} & \left\{ \textit{utt-time}(\text{k}) \right\} \end{bmatrix},
\begin{bmatrix} \text{INDEX} & \text{t} \\ \text{REST} & \left\{ \textit{precedes}(\text{t,k}) \right\} \end{bmatrix}, \\
\begin{bmatrix} \text{INDEX} & \text{s} \\ \text{REST} & \left\{ \right\} \end{bmatrix},
\begin{bmatrix} \text{INDEX} & \text{b} \\ \text{REST} & \left\{ \textit{named}(\text{Bo,b}) \right\} \end{bmatrix}
\end{array} \right\} \\
\text{CONTENT} & \begin{bmatrix}
\textit{ask-rel} & \\
\text{ASKER} & \text{i} \\
\text{ASKED} & \text{j} \\
\text{MSG-ARG} & \begin{bmatrix}
\textit{question} & \\
\text{PARAMS} & \left\{ \right\} \\
\text{PROP} & \begin{bmatrix}
\text{SIT} & \text{s} \\
\text{SOA} & \begin{bmatrix} \textit{leave-rel} & \\ \text{AGT} & \text{b} \\ \text{TIME} & \text{t} \end{bmatrix}
\end{bmatrix}
\end{bmatrix}
\end{bmatrix} \\
\text{CONSTITS} & \left\{ \begin{array}{l}
\begin{bmatrix} \text{PHON did} \end{bmatrix}, \begin{bmatrix} \text{PHON bo} \end{bmatrix}, \begin{bmatrix} \text{PHON leave} \end{bmatrix}, \\
\begin{bmatrix} \text{PHON did bo leave} \end{bmatrix}
\end{array} \right\}
\end{bmatrix}$$

Successful grounding implies finding appropriate values for the contextual parameters of the utterance. When the available contextual assignment is defective, i.e. the context available to the dialogue participant does not contain all the information needed to instantiate the contextually dependent parameters, a partial update of the existing context with the successfully grounded components of the utterance takes place.

Ginzburg and Cooper (2004) propose a range of *coercion operations* that explicate the partially updated context required for the interpretation of non-sentential clarification questions. Such operations have as input the original utterance, where a problematic contextual parameter i is singled out. The output of the operation is then the partially updated context with the information needed to resolve elliptical clarification questions.

The first of these operations of partial update they consider, however, does not involve positing a clarification question. Given a problematic parameter, speakers might decide to just existentially quantify over the referent that cannot be found in context, if this is sufficient for current purposes. Israel and Perry (1990) point out that in such situations dialogue participants update their context with *pure content* (i.e. content with no intervention of contextual resources) by extracting existential information from the utterance. In (Ginzburg and Cooper 2004) this is formalised in terms of the coercion operation in (82). Given an utterance with a contextual parameter i, this rule allows one to construct a sign where i is quantified away with widest possible scope. Thus a dialogue participant who finds a contextual parameter i problematic can use this rule to construct a contextually less dependent meaning where i is existentially quantified.

(82) **Contextual existential generalisation**$_i$

$$\begin{bmatrix} \textit{root-cl} & \\ \text{C-PARAMS} & \boxed{1}\left\{\ldots \mathrm{i} \ldots\right\} \\ \text{CONTENT} & \begin{bmatrix} \textit{prop} & \\ \text{SIT} & \mathrm{s} \\ \text{SOA} & \begin{bmatrix} \text{QUANTS} & \left\langle\right\rangle \\ \text{NUCL} & \boxed{3} \end{bmatrix} \end{bmatrix} \end{bmatrix} \Rightarrow \begin{bmatrix} \textit{root-cl} & \\ \text{C-PARAMS} & \boxed{1} \setminus \left\{\mathrm{i}\right\} \\ \text{CONTENT} & \begin{bmatrix} \textit{prop} & \\ \text{SIT} & \mathrm{s} \\ \text{SOA} & \begin{bmatrix} \text{QUANTS} & \left\langle \begin{bmatrix} \exists\textit{-rel} \\ \mathrm{i} \end{bmatrix} \right\rangle \\ \text{NUCL} & \boxed{3} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

The other two coercion operations introduced by Ginzburg and Cooper (2004) are used in the interpretation of elliptical clarification questions like those in (79), repeated here in (83b) together with their possible interpretations in (83c) and (83d).

(83) a. A: Did Bo leave?

b. B: WHO? / Bo?

c. *Who$_i$ are you asking if i left? / Are you asking if BO left?*

d. *Who is Bo?*

The interpretation of B's response given in (83c) is accounted for by the rule in (84). In the partially updated context created by the application of the rule, the constituent associated with the problematic parameter becomes the salient utterance. The maximal QUD is coerced to a question formed by abstracting over the problematic parameter from the original content.

(84) **Parameter focusing**$_i$

$$\begin{bmatrix} \textit{root-cl} & \\ \text{C-PARAMS} & \boxed{1}\{\ldots \mathrm{i} \ldots\} \\ \text{CONSTITS} & \left\{\ldots \boxed{2}\begin{bmatrix}\text{CONT} & \mathrm{i}\end{bmatrix}\right\} \\ \text{CONTENT} & \boxed{3}\end{bmatrix} \Rightarrow \begin{bmatrix}\textit{root-cl} & \\ \text{CONTEXT} & \begin{bmatrix}\text{SAL-UTT} & \boxed{2}\\ \text{MAX-QUD} & \begin{bmatrix}\textit{question} & \\ \text{PARAMS} & \{\mathrm{i}\} \\ \text{PROP} & \boxed{3}\end{bmatrix}\end{bmatrix}\end{bmatrix}$$

Applying (84) to (83a) yields to a context where MAX-QUD is a question paraphrasable as *"Who$_b$ named Bo are you asking if b left?"*, whereas SAL-UTT is the sub-utterance of *"Bo"*. Clarification questions like those in (83b) can then be analysed as a short answer to the maximal QUD using the type *decl-frag-cl* shown above in (76). This construction is then the head daughter of a type called *direct-insitu-interrogative-clause*, which basically constructs a polar question out of the propositional content of its head daughter. The parameter which constitutes the content of the *wh*-phrase WHO is retrieved by the *direct-insitu-interrogative-clause* type to yield the *wh*-question reading given in (83c), while the absence of an interrogative parameter yields the polar reading *"Are you asking if BO left?"*.

The other coercion operation proposed by Ginzburg and Cooper (2004) is shown in (85). Again, when the rule is applied, the problematic constituent becomes the salient utterance. In this case however MAX-QUD is a question asking about the content of the problematic sub-utterance as intended by the speaker. To resolve the clarification questions in (83b) according to the reading in (83d) a type *utterance-anaphoric-phrase* is introduced, which enables anaphoric reference to the phonologically identical SAL-UTT. This is then the head daughter of yet a new clausal type *constituent-clarification-interrogative-clause*, which identifies the content of the clarification question with the MAX-QUD, yielding the reading in (83d).

(85) **Parameter identification**$_i$

$$\begin{bmatrix} \textit{root-cl} & \\ \text{C-PARAMS} & \boxed{1}\left\{\ldots \mathrm{i} \ldots\right\} \\ \text{CONSTITS} & \left\{\ldots \boxed{2}\begin{bmatrix}\text{CONT} & \mathrm{i}\end{bmatrix}\ldots\right\} \end{bmatrix}$$

$\Rightarrow$

$$\begin{bmatrix} \textit{root-cl} & \\ \text{C-PARAMS} & \left\{\ldots \mathrm{k}{:}\textit{addr}(\mathrm{k}) \ldots\right\} \\ \text{CONTEXT} & \begin{bmatrix} \text{SAL-UTT} & \boxed{2} \\ \text{MAX-QUD} & \begin{bmatrix} \textit{question} & \\ \text{PARAMS} & \left\{\mathrm{i}\right\} \\ \text{PROP} \mid \text{SOA} & \begin{bmatrix} \textit{spk-meaning-rel} & \\ \text{SPK} & \mathrm{k} \\ \text{CONST} & \boxed{2} \\ \text{CONT} & \mathrm{i} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

3.4 A Dynamic Logic-based Approach

Perhaps with the exception of Schlangen, who simply assumes that his semantic representations are interpreted dynamically, the approaches reviewed in previous sections are all concerned with a static notion of interpretation. Since dialogue is inherently dynamic, i.e. cannot be analysed in terms of isolated sentences, and since NSU resolution clearly depends on the preceding dialogue, this seems to be a shortcoming. In this section I present the main ideas of an approach that, in line with the dynamic semantics tradition, attempts to overcome this limitation by combining, in this case, HPSG and Dynamic Logic.

I first summarise the basic notions of Dynamic Logic, and then introduce the formalisation presented in (Fernández 2003a,b), which is the starting point of the core approach, described in Section 3.4.3.

3.4.1 Dynamic Logic: Basic Notions

The formalisation presented in (Fernández 2003a,b) is based on the first-order version of Dynamic Logic (DL) as it is introduced in (Harel et al. 2000) and (Goldblatt 1992). In short, DL is a multi-modal logic with a possible worlds semantics, which distinguishes between expressions of two sorts: *formulae* and *programs*. The language of DL is that of first-order logic together with a set of modal operators: for each program α there is a box $[\,\alpha\,]$ and a diamond $<\!\alpha\!>$ operator. The set of possible worlds (or states) in the model is the set of all possible assignments to the variables in the language. Atomic programs change the values assigned to particular variables. They can be combined to form complex programs by means of a repertoire of program constructs, such as *sequence* ; , *choice* $\cup$, *iteration* * and *test* ?. The set of programs α is defined as follows, where φ is any quantifier-free first-order formulae:

$$\alpha \ ::= \ \pi \mid \alpha_1;\alpha_2 \mid \alpha_1 \cup \alpha_2 \mid \alpha^* \mid \varphi?$$

In the basic version of DL, atomic programs π are simple assignments $(x := t)$, where x is an individual variable and t is a first-order term.

The language is interpreted in a possible worlds model $\mathcal{M} = \{\mathcal{A}, S, R, V\}$ where $\mathcal{A}$ is a first-order structure; S is a non-empty set of states; R is a function assigning to each program α a binary relation $R_\alpha \subseteq S \times S$; and V is a function $V : S \to S^{\mathcal{A}}$ assigning to each $s \in S$ a mapping v_s from the set of variables to elements in the domain. The definition of the *truth-relation* $\mathcal{M} \models_s A$ of a formula A at state s in model $\mathcal{M}$ is standard. For conciseness, I give only the semantics of modal formulae. For $s, s' \in S$, I write $s(x|a)s'$ to mean that $v_{s'}(x) = a$ and $v_{s'}(y) = v_s(y)$ whenever $y \neq x$.

(86)
$$\begin{array}{lll} \mathcal{M} \models_s <\!\alpha\!> A & \text{iff} & \text{there is an } s' \in S \text{, such that } sR_\alpha s' \text{ and } \mathcal{M} \models_{s'} A \\ \mathcal{M} \models_s [\,\alpha\,]\, A & \text{iff} & \text{for all } s' \in S \text{, if } sR_\alpha s' \text{ then } \mathcal{M} \models_{s'} A \end{array}$$

From the relations $R_\alpha \subseteq S \times S$, we can inductively define accessibility relations for the compound programs. (87) shows the accessibility relations for basic atomic programs and compound programs for all states $s, s' \in S$.

(87)
$$\begin{array}{|lll} sR_{x:=t}s' & \text{iff} & s(x|v_s(t))s' \\ sR_{\alpha;\beta}s' & \text{iff} & \exists s'' \text{ s.t. } sR_\alpha s'' \text{ and } s''R_\beta s' \\ sR_{\alpha\cup\beta}s' & \text{iff} & \text{either } sR_\alpha s' \text{ or } sR_\beta s' \\ sR_{\alpha*}s' & \text{iff} & \text{there are finitely many states } s_1 \ldots s_n \text{ such that } sR_\alpha s_1 \ldots s_n R_\alpha s' \\ sR_{\varphi?}s' & \text{iff} & s = s' \text{ and } \mathcal{M} \models_s \varphi \end{array}$$

Interesting variants of DL arise from allowing auxiliary data structures such as *stacks* and *arrays*. Following (Harel et al. 2000), I consider a version of DL in which programs can manipulate some variables as last-in-first-out stacks. Formally, stacks are modelled as variables ranging over finite strings of elements in the domain. To manipulate these *stack variables*, two additional atomic programs X.**pop** and X.**push**(x) are included. Here X is some stack variable (i.e. a string of elements) and x is an element in X. The accessibility relations for these two new atomic programs are shown in (88), where, for a string σ and an element a, **tail**$(a \cdot \sigma) = \sigma$ and **head**$(a \cdot \sigma) = a$.

(88)
$$\begin{array}{lll} sR_{\mathtt{X}.\mathbf{push}(x)}s' & \text{iff} & s(\mathtt{X} \mid v_s(x) \cdot v_s(\mathtt{X}))s' \\ sR_{\mathtt{X}.\mathbf{pop}}s' & \text{iff} & s(\mathtt{X} \mid \mathbf{tail}(v_s(\mathtt{X}))s' \end{array}$$

3.4.2 A DL Formalisation of the Dialogue Gameboard

Originally, DL was conceived as a formal system to reason about computer programs, formalising correctness specifications and proving rigorously that those specifications are met by a particular program. From a more general perspective, however, it can be viewed as a formal system to reason about transformations on states. In this sense, it is particularly well suited to provide a fine characterisation of the dynamic processes that take place in dialogue as updates on the information states of the dialogue participants.

In Fernández (2003a,b) I use DL to formalise a notion of IS based on Ginzburg's *Dialogue Gameboard* (DGB) (Ginzburg 1996). The DGB provides a structured characterisation of the information which the dialogue participants view as common in terms of three main components: a set of FACTS, which the dialogue participants take as common ground, a partially ordered set of questions under discussion QUD, and the LATEST-MOVE made in the dialogue.

As mentioned earlier, in DL transitions between states are changes in variable assignment. I use the variable names FACTS, QUD and LM to represent the three dynamic components of the DGB. I model QUD as a stack, in a way that is very much inspired by QUD's actual implementation in the GoDiS dialogue system (Larsson et al. 2000). LM ranges over utterances, $m(i, c)$, where i is interpreted as the speaker of the utterance, m is the dialogue move performed, and c represents its content. Although FACTS is assumed to be a (possibly ordered) set, for technical reasons I also model FACTS as a stack. This is so because we want to be able to check whether a particular element (i.e. some proposition) is in FACTS, and we want to be able to express this in the object language. Modelling FACTS as a variable ranging over strings of propositions allows us to use the **pop** program to check whether a particular element x belongs to FACTS or not: if x is in FACTS and we pop the stack repeatedly, x will show up at some point as the head of the stack. Thus, I shall use the notation $x \in \mathtt{FACTS}$ as an abbreviation for $< \mathtt{FACTS}.\mathbf{pop}^* > \ \mathbf{head}(\mathtt{FACTS}) = x$.[21]

The formalisation outlined above can be used to characterise the internal structure of conversational interaction in several ways. In (Fernández 2003b) it is used to constrain the model by means of a set of axioms that restrict the operations that can be performed on the DGB, and ensure that certain desirable properties hold. In (Fernández 2003a) an alternative approach is presented whereby interaction patterns of particular

[21]Note that, according to the semantics of modal formulae given in (86) above, the FACTS stack does not have to be *actually* popped for $<\mathtt{FACTS}.\mathbf{pop}^*> \ \mathbf{head}(\mathtt{FACTS}) = x$ to hold. The formula holds iff it is *possible* to find x on the top of the stack by repeatedly popping it.

communicative exchanges, like asking and responding to a question or acknowledging a proposition, are characterised by means of conventional protocols defined in terms of complex DL programs. Although interesting, the details of these formalisations are not important for current purposes. Hence, I forgo giving them here, and refer the reader to (Fernández 2003a,b) for further particulars.

3.4.3 Utterances as Update Instructions

On the basis of the formal system outlined in the previous section, (Purver and Fernández 2003, Fernández and Purver 2004) develop a follow-up approach whereby utterances are associated with DL programs assigned by an HPSG grammar.

The basic idea of the approach is to view utterances and their subconstituents as instructions for contextual update. This is achieved by associating them with complex DL programs that by definition denote a transition between states $s \xrightarrow{U} s'$. Under this view, processing an utterance U amounts to executing its utterance program. As long as the program contains all relevant instructions for updating the information state, the Utterance Processing Protocol (UPP) of Ginzburg and Cooper (2004) (given earlier in (80), and repeated here in (89) for convenience) no longer needs to be specified as a separate set of instructions. Instead, one can merely say that an information state s can integrate an utterance U iff $\mathcal{M} \models_s < U > \top$ (i.e. U succeeds for the current information state).

(89) **Utterance Processing Protocol (simplified version)**

1. Add U to PENDING
2. Attempt to ground U by instantiating each contextual parameter i in U
3. If successful:
   (a) remove U from PENDING;
   (b) add *content*(U) to LATEST-MOVE;
   (c) if *content*(U) = *assert*(p): push *whether*(p) onto QUD
   (d) if *content*(U) = *ask*(q): push q onto QUD
4. Else: form clarification question about i

The most basic form an utterance program can take is $\texttt{LM} := m(i, c)$, thus assigning a conversational move $m(i, c)$ to $\texttt{LM}$, as per part 3(b) of the original UPP. The effects of parts 3(c,d) of the UPP are achieved by more complex programs for questions and assertions, which are sequences of atomic programs, as in (90):

(90) $\texttt{LM} := ask(a, q); \texttt{QUD}.\mathbf{push}(q)$
$\texttt{LM} := assert(a, p); \texttt{QUD}.\mathbf{push}(whether(p))$

The effect of the above programs can be visualised as a transition between information states:

$$\begin{bmatrix} \texttt{LM} = m(i,c)_0 \\ \texttt{QUD} = qud_0 \end{bmatrix} \xrightarrow{U} \begin{bmatrix} \texttt{LM} = ask(a,q) \\ \texttt{QUD} = q \circ qud_0 \end{bmatrix}$$

This approach also allows programs which achieve the same effect as (Ginzburg and Cooper 2001)'s formulation of utterances as functions from context to content. The program associated with an expression which introduces a contextual parameter must be a program which finds an instantiation of the parameter in context (i.e. that succeeds only when a referent is present). Given the current formalism, this will be a test program $(t \in \texttt{FACTS})?$, which checks that $t \in \texttt{FACTS}$ holds in the current state. A suitable representation for the sentence *"John snores"*, in which a referent for *"John"* must be found in context, might be the following complex program:[22] [23]

(91) $(name(x, john) \in \texttt{FACTS})?;$
$\quad\quad \texttt{LM} := assert(snore(x));$
$\quad\quad\quad \texttt{QUD}.\mathbf{push}(whether(snore(x)))$

It is possible to distinguish between *given* referents which must be *found* in context as with the proper name *"John"* above, and *new* referents (e.g. indefinites) which should be *added* to the context, by associating indefinites with a program which introduces a new referent (thus following the dynamic semantic tradition of (Heim 1982, Kamp and Reyle 1993, Groenendijk and Stokhof 1991), and subsequent DRT-based dialogue theory such as (Poesio and Traum 1998)). As shown in (92), the program associated with a sub-utterance with a given referent tests for existence in the current state, while that for a new referent involves a state change introducing that referent.[24]

(92) a. *the dog* : $\quad (dog(x) \in \texttt{FACTS})?$

b. *a dog* : $\quad \texttt{FACTS}.\mathbf{push}(dog(x))$

[22]Here I take x to be a variable ranging over a finite set of people. In this case the test program $(name(x, john) \in \texttt{FACTS})?$ would succeed only if there is a *john* in FACTS and x happens to be assigned to *john* in the current world. With this move I am thus narrowing down the set of possible assignment functions to those that have this property.

[23]In fact, for proper names and definites, it will not be enough to require that there is *a* known referent: we need there to be a *unique/most salient* referent. The statement of a suitable test program will depend on one's theory of definiteness, but this should not affect the general approach presented here.

[24]In fact, Purver and Fernández (2003) propose more complex programs for expressions that introduce contextual parameters, that account for their clarification potential as specified in part 4 of the utterance processing protocol in (89). More details on this can be found in the original paper.

The approach assumes that DL programs are assigned to utterances by the sentence grammar. The HPSG grammar used is similar to that of Ginzburg and Sag (2001), with the utterance programs assigned to a new feature C(ONTEXTUAL)-PROG(RAMS), which replaces the feature C-PARAMS of Ginzburg and Cooper (2004). By default, this feature is built up by phrases, by linear combination of the programs associated with their syntactic daughters, using the sequence operator as shown in (93).

(93) $$\begin{bmatrix} \text{C-PROG} & A; \ldots ; B \\ \text{DTRS} & \langle\ [\text{C-PROG } A], \ldots, [\text{C-PROG } B]\ \rangle \end{bmatrix}$$

The default program associated with a phrase is therefore a purely sequenced combination of the programs contributed by its daughters, but this default is overwritten for particular phrase types which by their nature make their own contributions to the overall program (this is what the use of *constructions* affords us). For example, the clause type *root-clause* is specified to add the sub-program which updates LM:

(94) $$\begin{bmatrix} \textit{root-clause} & \\ \text{CONT} & \boxed{1}\begin{bmatrix}\textit{illoc-rel}\end{bmatrix} \\ \text{C-PROG} & A; \texttt{LM} := \boxed{1} \\ \text{HEAD-DTR} \mid \text{C-PROG} & A \end{bmatrix}$$

Clauses of type *declarative* (which have propositions as their semantic content) and *interrogative* (which denote questions) add the sub-program which updates QUD:

(95) $$\begin{bmatrix} \textit{declarative} & \\ \text{CONT} & \boxed{1}\begin{bmatrix}\textit{proposition}\end{bmatrix} \\ \text{C-PROG} & A; \texttt{QUD}.\mathbf{push}(\textit{whether}(\boxed{1})) \\ \text{HEAD-DTR} \mid \text{C-PROG} & A \end{bmatrix}$$

(96) $$\begin{bmatrix} \textit{interrogative} & \\ \text{CONT} & \boxed{1}\begin{bmatrix}\textit{question}\end{bmatrix} \\ \text{C-PROG} & A; \texttt{QUD}.\mathbf{push}(\boxed{1}) \\ \text{HEAD-DTR} \mid \text{C-PROG} & A \end{bmatrix}$$

Phrases which contribute given or new referents are specified in an entirely parallel way. Given referents such as those associated with definite NPs and proper names contribute sub-programs which express restrictions on the type of state to which the utterance program can be successfully applied—to wit, that the state contain a suitable antecedent, as specified in (97) for a definite NP.

(97) $$\begin{bmatrix} \textit{definite} & \\ \text{CONT} & \boxed{1}\begin{bmatrix}\textit{parameter}\end{bmatrix} \\ \text{C-PROG} & A; B; (\boxed{1} \in \texttt{FACTS})? \\ \text{DTRS} & \langle\ [\text{C-PROG } A],\ [\text{C-PROG } B]\ \rangle \end{bmatrix}$$

NSUs can also be seen in this way: as being licensed only in certain types of context, and therefore as expressing conditions on the kind of state to which their programs can apply. Thus, in this approach, NSUs are regarded as introducing sub-programs which must ensure that the contextual information required for resolution is present in the current state, and by finding it, fully instantiate their content. The grammatical approach directly follows that of Ginzburg and Sag (2001). The content of a short answer is taken to be a proposition which must be associated with the current maximal question under discussion; the referential index of its head daughter must be identified with that of a SAL-UTT utterance which is also constrained to be syntactically parallel to it.

(98) $$\begin{bmatrix} \textit{decl-frag-cl} & \\ \text{CONTENT} & \boxed{1} \\ \text{HEAD-DTR} & \begin{bmatrix} \text{CAT} & \boxed{2} \\ \text{CONT}\,|\,\text{INDEX} & \boxed{3} \end{bmatrix} \\ \text{CONTEXT} & \begin{bmatrix} \text{MAX-QUD}\,|\,\text{PROP} & \boxed{1} \\ \text{SAL-UTT} & \begin{bmatrix} \text{CAT} & \boxed{2} \\ \text{CONT}\,|\,\text{INDEX} & \boxed{3} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Now, the only change that must be made is that root clauses, besides adding the sub-program which updates `LM`, must add sub-programs which require the specified contextual information to be found, as shown in (99). Note that the order of the programs is important: the contextual information must be identified in the initial state, before it is changed by the utterance program (which may of course update `QUD`), and of course before the `LM` state variable can be set to the fully specified move, i.e. the overall utterance content.

This seems to make the status of this contextual information clearer than in either Ginzburg and Sag (2001) or Schlangen (2003) approaches. In the former, the utterance is left underspecified by the grammar, and we must assume separately specified pragmatic routines as given by Ginzburg's theory of context to fill it in; in the latter, this underspecification is replaced by an *unknown* anaphorical relation essentially unaccompanied by information about possible antecedents, which must be identified by pragmatic inference. In the present DL approach, not only is the method of content specification fully defined by the grammar as a program, the source of the antecedents (particular state variables) is also made clear.

(99)
$$\begin{bmatrix} \textit{root-clause} & \\ \text{CONT} & \boxed{1}\begin{bmatrix}\textit{illoc-rel}\end{bmatrix} \\ \text{CONTEXT} & \begin{bmatrix}\text{MAX-QUD} & \boxed{2} \\ \text{SAL-UTT} & \boxed{3}\end{bmatrix} \\ \text{C-PROG} & \textbf{head}(\texttt{QUD}) = \boxed{2}\,)?;\ (\textbf{head}(\texttt{UTT}) = \boxed{3}\,)?;\ A;\ \texttt{LM} := \boxed{1} \\ \text{HEAD-DTR}\,|\,\text{C-PROG} & A \end{bmatrix}$$

3.5 Summary and Conclusions

This chapter has reviewed several approaches to the resolution of NSUs. These range from purely semantic approaches like those based on HOU and Dekker's proposal, to more syntactic accounts like that of Merchant. Semantic approaches are elegant and include valuable ideas like parallelism and the salience of a property or question in context, that are then implemented in different ways in the hybrid approaches of Section 3.3. Their main problem is, however, that they cannot account for the structural dependencies exhibited by NSUs. The approach of Merchant, on the other hand, is able to explicate some of these dependencies, but it does so at the cost of treating NSUs as *hidden* full sentences, which undermines their status of proper grammatical utterances and brings in several complications derived from the mismatches between full sentences and NSUs.

In contrast to this, the HPSG-based approaches presented in Section 3.3 see NSUs as first class grammatical constructions, and offer a more balanced account of syntax and semantics. At the level of grammatical representation, the approach of Schlangen is very similar to that of Ginzburg and colleagues. Although only the former uses an explicit underspecified semantics, HPSG constraints—being partial descriptions—are in themselves underspecified representations. Hence, at the end of the day, both approaches make use of underspecification to account for the semantic content of NSUs. At the dynamic level, the role played by Ginzburg's theory of context in Ginzburg and Sag's model is played by SDRT in Schlangen's. We have seen however that Schlangen offers an account of the structural dependencies of NSUs that is limited to subcategorisation requirements of predicates.

The Dynamic Logic approach presented in the last section goes one step further in that the representation of utterances incorporates their dynamic import. The main idea of this approach, which builds on Ginzburg and Sag (2001) and Ginzburg and Cooper (2004), is to see utterances as dynamic logic programs that act as update instructions, concretely as a sequence of tests on the current state followed by a sequence of updates that take us to the next state. This is the idea I take as point of departure for the proposal I shall present in the next chapter. It will be formulated in Type Theory with Records, which will allow us to combine aspects from HPSG and Dynamic Logic in one single formalism.

4 A Type-theoretical NSU Grammar

This chapter presents a formal analysis of the main NSU classes identified in Chapter 2. The analysis builds on the ideas presented in Section 3.4 of the previous chapter, but instead of being encoded in a combination of HPSG and Dynamic Logic, here I propose a formalisation in terms of Type Theory with Records. This will be shown to bring in several advantages, most notably perhaps a more consistent notation that allows for a smooth transition between grammar and dialogue processes. I will start by introducing Type Theory with Records rather informally (more precise and formal definitions are given in Appendix B). In Section 4.2, I will then use the formalism to put forward a grammatical representation of utterances akin to update rules in dialogue modelling. This will provide the basis for the representation of the main NSU types, classified according to the complexity of the information state and the kind of information required for the resolution process.

4.1 Type Theory with Records

Constructive Type Theory is a proof-theoretic framework developed in the seventies by the Swedish logician Per Martin-Löf with the aim, shared with the intuitionistic tradition, to provide a foundation for constructive mathematics. Since the early nineties there has been an interest in using Constructive Type Theory for linguistics, attested for instance by Ranta's monograph (Ranta 1994). The theory has been used mostly as a framework for natural language semantics. Indeed work by researchers like Krahmer and Piwek (1998) or Fernando (2001) has shown that it can be an efficient tool to develop accounts of semantic phenomena like presupposition, quantification and anaphora, often with strong connections with DRT approaches.[1]

Here I shall use a particular version of this theory, namely *Type Theory with Records* (TTR), which is an extension of the basic system with *records* and *record types* due to

[1]See for instance (Ahn and Kolb 1990) for a reformulation of early DRT with Constructive Type Theory.

Betarte and Tasistro (1998). In particular, I draw extensively on recent work by Robin Cooper, who demonstrated the viability of using TTR for linguistic purposes. Cooper's pioneering work (Cooper 2000, 2006a) pointed out significant connections between TTR and Situation Theory (Barwise and Perry 1983), and showed how TTR amalgamates several desirable features of other frameworks like DRT (Kamp and Reyle 1993) and Montague semantics (Montague 1974). Together with Ginzburg (Cooper and Ginzburg 2002, Cooper 2005), he has also shown that there are interesting similarities between TTR and HPSG. In addition, as we will see later on in this chapter, he has recently explored the possibility of using TTR to formalise update rules in *issue-based dialogue management* (Cooper 2006b). Without any doubt, this body of foundational work by Cooper and colleagues has been a very valuable source of inspiration for the formalisation I shall present here.

The following subsections are devoted to introducing the basic notions of the framework.

4.1.1 Basics of TTR

4.1.1.1 Typing Judgements

The most basic idea underlying any type theory is that objects can be classified as being of different types. Thus, one of the most fundamental notions of TTR is the *typing judgement*. Typing judgements are of the form $x : T$ (read as "x is of type T"), where x is a variable standing for some object of type T. Consider for instance the following sentence:

(100) A spy procrastinates.

Assuming a system that includes a basic type Ind as the type of individuals or entities and predicates Spy and $Procrastinate$, the content of the sentence in (100) could be modelled by the following judgements:

(101) $x : Ind\ , \quad y : Spy(x)\ , \quad z : Procrastinate(x)$

Here variable x stands for an arbitrary object of type $Ind(ividual)$, while the types $Spy(x)$ and $Procrastinate(x)$ can be regarded as propositions, or in TTR terminology as *proof types*.[2] For an object y to be of type $Spy(x)$ means that y is a *proof* of the proposition $Spy(x)$, i.e. a proof that the object assigned to x is a spy.[3] Thus when applied to basic

[2]As we will see below, they are in fact *families* of proof types, whose elements depend on the objects assigned to x. I ignore this for now and return to it in Subsection 4.1.1.5 where dependent record types are introduced.

[3]This is in fact the idea at the core of the *propositions-as-types proofs-as-objects* paradigm (Curry and Feys 1958), under which propositions are equated with the set (or the type) of its proofs.

types, the colon in $x : \mathit{Ind}$ can be read as set membership. When applied to proof types like in $y : \mathit{Spy}(x)$ however, it is more naturally read as "proves".

4.1.1.2 Records and Record Types

Besides objects, basic types and proof types, in TTR we also have *records* and *record types*. In a general sense, records are mathematical objects consisting of finite sets of pairs $\langle l, v \rangle$ of labels l and values v called *fields*. They are usually represented graphically as a matrix:

$$\begin{bmatrix} l_1 & v_1 \\ \vdots & \\ l_n & v_n \end{bmatrix}$$

Record types are records whose fields correspond to typing judgements. Judgements in a record type are therefore pairs of labels (variables) and types. Given this, the content of sentence (100) can be represented by the following record type:

(102) $\begin{bmatrix} \text{x} & : & \mathit{Ind} \\ \text{y} & : & \mathit{Spy}(\text{x}) \\ \text{z} & : & \mathit{Procrastinate}(\text{x}) \end{bmatrix}$

One can say that a type *is true* if it is inhabited or *witnessed*, i.e. if it is the case that there is at least one object of that type. The inhabitants of record types are *records*. Records are sequences of fields which are pairs of labels l and objects a, written as $l = a$. A record r is of type ρ if all typing judgements in ρ are satisfied by r. More precisely,

Definition 1 (Record Typehood) *A record r is of type ρ iff for each field $\langle l, T \rangle$ in ρ there is a field $\langle l, a \rangle$ in r such that $a : T$.*

Hence, record type (102) is inhabited if we can find a record like (103), such that $a : \mathit{Ind}$, $p_1 : \mathit{Spy}(a)$ and $p_2 : \mathit{Procrastinate}(a)$. Which in turn means finding an individual who is a spy and who procrastinates. Therefore record types can be considered truth-bearing objects, and thus we obtain an effect of existential quantification like in DRT (Cooper 2000).

(103) $\begin{bmatrix} \text{x} = a \\ \text{y} = p_1 \\ \text{z} = p_2 \end{bmatrix}$

It is in fact hard to miss that there is a parallelism between record types like (102) and DRSs. Indeed fields of the form $l : \mathit{Ind}$ in a record type can be seen as corresponding to discourse referents in a DRS, while fields of the form $l : \mathit{ProofType}$ (where $\mathit{ProofType}$ is the type of proof types) would correspond to DRT conditions.[4] Record type (102) could thus be "transformed" into the following DRS:

(104) $\begin{array}{|c|}\hline \mathrm{x} \\ \hline \mathrm{spy(x)} \\ \mathrm{procrastinate(x)} \\ \hline \end{array}$

Going back to the definition of typehood in Definition 1, note that according to it, record (103) could have additional fields and still be of type (102). Indeed, record (103) is of all the types in (105).

(105) a. $\left[\begin{array}{lll} \mathrm{x} & : & \mathit{Ind} \\ \mathrm{y} & : & \mathit{Spy}(\mathrm{x}) \end{array}\right]$ b. $\left[\begin{array}{lll} \mathrm{x} & : & \mathit{Ind} \end{array}\right]$ c. $[\ ]$

This leads to the following definition of the type inclusion (or subtyping) relation $\sqsubseteq$ between types:

Definition 2 (Subtyping) $T_1 \sqsubseteq T_2$ *iff* $a : T_1$ *implies* $a : T_2$.

In words, type T_1 is a *subtype* of type T_2 just in case any object of type T_1 is also of type T_2. In the case of record types, $\rho_1 \sqsubseteq \rho_2$ holds just in case for each field $\langle l, T_2 \rangle$ in ρ_2 there is a field $\langle l, T_1 \rangle$ in ρ_1 such that $T_1 \sqsubseteq T_2$. The type inclusion relation is reflexive and transitive.

For the examples above it holds that (102) $\sqsubseteq$ (105a) $\sqsubseteq$ (105b) $\sqsubseteq$ (105c). All records are of the empty record type (105c), the type which imposes no constraints. Hence it follows that all record types are subtypes of the empty record type [ ].

4.1.1.3 Nested Records and Paths

Records and record types can be nested, i.e. values in a record can be records themselves (and accordingly for record types). Such recursivity makes records and record types strikingly similar to the attribute-value matrices (AVMs) familiar from constraint-based formalisms, like the AVMs used to represent feature structures in HPSG. Like them, they are useful means of encoding different kinds of information in a structured manner. Although the examples used so far look pretty much like logical forms, the information encoded in a record does not have to be restricted to semantics. Nothing stops us from using records and record types to model other kinds of linguistic (and non-linguistic!) information. For instance, assuming a type Phon as the type of phonological strings and

[4]Note however that, as pointed out by Stephen Pulman (p.c.) while DRSs denote world entities, record types are inhabited by records, which act as an additional level of representation.

a type Cat as the type of syntactic categories, our sentence in (100) could be modelled by the record type in (106):

(106)
$$\begin{bmatrix} \text{phon} & : & \mathit{Phon} \\ \text{syn} & : & \mathit{Cat} \\ \text{sem} & : & \begin{bmatrix} \text{x} & : & \mathit{Ind} \\ \text{y} & : & \mathit{Spy}(\text{x}) \\ \text{z} & : & \mathit{Procrastinate}(\text{x}) \end{bmatrix} \end{bmatrix}$$

Here I have used the labels 'phon', 'syn' and 'sem' to encode phonological, syntactic and semantic information respectively in one relational structure, as it is common in HPSG grammars. Note that the value of label 'sem' , representing the content of the sentence (or a simplified version thereof), is the record type in (102).

This example allows me to introduce some useful notation. I will use $r.l$ to denote the value of label l in record *r*. To refer to values in nested records, I will use sequences of labels or *paths*.

Definition 3 (Paths) *A path π in a record r (written as $r.l_1.l_2.\ldots.l_n$) is a sequence of labels $l_1, l_2, \cdots, l_n$ such that l_1 is a label in a field in r and for any label l_{k+1}, the value of l_k is a record r' and l_{k+1} is a label in r'.*

Thus, in a record r of type (107) for instance, we can refer to the value of label x with the path $r.\text{sem.x}$, which will denote the object assigned to x in r. If r is the record in (108), then $r.\text{sem.x}$ denotes object a.

(107)
$$\begin{bmatrix} \text{syn} & : & \mathit{Cat} \\ \text{sem} & : & \begin{bmatrix} \text{x} & : & \mathit{Ind} \\ \text{y} & : & \mathit{Ind} \\ \text{z} & : & \mathit{Greet}(\text{x}, \text{y}) \end{bmatrix} \end{bmatrix}$$

(108)
$$\begin{bmatrix} \text{syn} & = & \text{S} \\ \text{sem} & = & \begin{bmatrix} \text{x} & = & a \\ \text{y} & = & b \\ \text{z} & = & p_1 \end{bmatrix} \end{bmatrix}$$

4.1.1.4 Type Constructors

So far we have seen that the system of TTR includes basic types, proof types and record types. From this ground, we can recursively build up the universe of types with the help of several type construction operations.

The singleton type This type constructor allow us to raise objects to the level of types, by creating types with a single object as inhabitant. More precisely, for any element a of type T, we can create the *singleton type* T_a such that only a is of type T_a. Building on the concept of singleton types, Coquand et al. (2004) introduce the notion of *manifest field*. A manifest field in a record type is a field whose value is a singleton type. For convenience, manifest fields like for instance $\mathrm{x} : T_a$ will be written as $\mathrm{x} = a : T$.

Singleton types and manifest fields can be used to further specify record types. For instance, assuming (as I did above) that our model includes linguistic objects $\mathrm{S}, \mathrm{NP}, \mathrm{VP}, \mathrm{N}, \ldots$ of type *Cat*, record type (109) is also a type for record (108)—a more specific type than (107). In particular it holds that (109) is a subtype of (107).

(109)
$$\begin{bmatrix} \mathrm{syn} = \mathrm{S} : \mathit{Cat} & & \\ \mathrm{sem} & : & \begin{bmatrix} \mathrm{x} & : & \mathit{Ind} \\ \mathrm{y} & : & \mathit{Ind} \\ \mathrm{z} & : & \mathit{Greet}(\mathrm{x}, \mathrm{y}) \end{bmatrix} \end{bmatrix}$$

List types As is common in type theory, TTR also provides us with list types. Given a type T we can create the list type $\langle T \rangle$, i.e. the type of lists of elements of type T. L is of type $\langle T \rangle$ only if L is a list and whenever a is an element of L then $a : T$. I shall use the symbol $\oplus$ as the concatenation operator for lists.

Lists and list types can be used for instance to model phrasal structure—in a way familiar from HPSG, we can use a list type to represent the daughters of a phrase. In (110) I use a label dtrs for this purpose. Its value is required to be of type $\langle \mathit{Rec} \rangle$, i.e. a list of records (elements of type Rec) corresponding to the daughters of the phrase.

(110)
$$\begin{bmatrix} \mathrm{phon} & : & \mathit{Phon} \\ \mathrm{syn} & : & \mathit{Cat} \\ \mathrm{sem} & : & \mathit{Type} \\ \mathrm{dtrs} & : & \langle \mathit{Rec} \rangle \end{bmatrix}$$

Function types The next type constructor I will introduce is perhaps the most basic one in any type theory—the constructor that generates *function types*. Application of the function type constructor to types T_1 and T_2 results in the function type $T_1 \to T_2$, i.e. the type of functions from elements of type T_1 to elements of type T_2. In TTR functions are assumed to be total. Therefore a function f is of type $(T_1 \to T_2)$ if f is a function whose domain is $\{a \mid a : T_1\}$ and whose range is a subset of $\{a \mid a : T_2\}$.

The availability of function types provides us with all the tools from the lambda calculus. Thus, as is common in Montague-style formal semantics, we can use functions to represent e.g. the denotations of verbs and VPs, and then use functional application and β reduction to compositionally build up the content of complex structures in the standard way.

Simplifying somewhat, we can take the content of verbs to be functions of type $\mathit{Ind} \rightarrow \mathit{ProofType}$ (roughly equivalent to Montague's type $\langle e, t\rangle$) and NPs to denote elements of type *Ind*. The content of a transitive VP can then be constructed by applying the denotation of the verb to that of the complement NP. This is what the following record type achieves, where I have used @ to represent functional application.

(111)
$$\begin{bmatrix} \text{syn} = \text{VP} : \mathit{Cat} \\ \text{sem} \quad : \quad \text{d1.sem@d2.sem} \\ \text{dtrs} = \left\langle \text{d}_1 : \begin{bmatrix} \text{syn} = \text{V} : \mathit{Cat} \\ \text{sem} : \mathit{Ind} \rightarrow \mathit{ProofType} \end{bmatrix}, \text{d}_2 : \begin{bmatrix} \text{syn} = \text{NP} : \mathit{Cat} \\ \text{sem} : \mathit{Ind} \end{bmatrix} \right\rangle : \langle \mathit{Rec} \rangle \end{bmatrix}$$

This constitutes a genuine advantage over HPSG unification-based semantic representations, where typically the semantic content of a phrase is identified with that of its head daughter by a *head feature principle*. Hence, when a head is accompanied by a complement, as in the case of transitive VPs, the semantic type assigned to the head already includes the contribution of the complement. In contrast, TTR allows us to assign a much more intuitive content to heads, making explicit their contribution to the semantic type of the mother.

Boolean operations on types Finally, new types can be constructed by means of basic boolean operations. Given a type T, its negation $\neg T$ is the type $T \rightarrow \bot$. We can also create new types by conjunction and disjunction of existing types. If T_1 and T_2 are types, so are $T_1 \wedge T_2$ and $T_1 \vee T_2$. A witness of a type $T_1 \wedge T_2$ is a pair $\langle a, b\rangle$ such that $a : T_1$ and $b : T_2$. A witness of a type $T_1 \vee T_2$ is an element a such that $a : T_1$ or $a : T_2$.

4.1.1.5 Families of Types and Dependent Record Types

There is more in a record type like (102), repeated here as (112), than what I have so far pointed out. Note that the proof types in this record type (i.e. the values of labels y and z) are somehow not fully determined, as they *depend* on the individual objects assigned to x. If $\text{x} = a$ the value of y is the proof type $\mathit{Spy}(a)$, whose witness is a proof that individual a is a spy. If x labels a different individual b, then y necessarily labels a different type, namely $\mathit{Spy}(b)$, whose witness in this case is a proof that b is a spy. Thus $\mathit{Spy}(\text{x})$ and $\mathit{Procrastinate}(\text{x})$ in (112) do not denote single types but rather sets or *families of types*. These are functions $\lambda(x : T).T'$ from elements x of some type T to types T' dependent on x such that applying $\lambda(x : T).T'$ to any element a of type T yields $T'_{[x \rightarrow a]}$, i.e. a type which is identical to T' except that all occurrences of x have been substituted by a.

(112) $$\begin{bmatrix} \text{x} & : & \mathit{Ind} \\ \text{y} & : & \mathit{Spy}(\text{x}) \\ \text{z} & : & \mathit{Procrastinate}(\text{x}) \end{bmatrix}$$

Record type (112) is therefore a *dependent record type*, i.e. a record type some of whose types depend on the choice of objects satisfying the judgements in preceding fields. Dependent record types have the following general form:[5]

(113) $$\begin{bmatrix} l_1 & : & T_1 \\ l_2 & : & T_2(l_1) \\ \vdots & & \\ l_n & : & T_n(l_1, l_2, \ldots, l_{n-1}) \end{bmatrix}$$

In general the record types that will be used in the sequel are dependent record types as they allow us to encode interesting dependencies between their components that are useful for linguistic purposes. I will also make extensive use of families of *record types*, i.e. functions $\lambda(x : T).T'$ whose output is a record type.

4.1.1.6 Some Operations on Record Types

To be able to use TTR for semantics and as a grammatical formalism, we will often need to combine the information contained in two of more record types. For this I define a notion of *record type union*. This notion has its grounds on the operation of extension used to inductively construct the set of record types from the empty record type by adding a new field (as defined in Appendix B). To make things easier, our record type union operation will allow us to extend a record type by adding several fields in one go. Intuitively the record type union operation will create a new record type containing all fields present in the record types to be united.

Definition 4 (Record type union) *Let Fields(ρ) be the set of fields in a record type ρ, and Labels(ρ) be its set of labels. Given record types $\rho_1, \ldots, \rho_n$ with pairwise disjoint sets of labels, i.e. $\mathit{Labels}(\rho_i) \cap \mathit{Labels}(\rho_j) = \emptyset$ for distinct ρ_i, ρ_j, their union is a record type $\mathcal{R}$ such that $\mathit{Fields}(\mathcal{R}) = \bigcup_{i=1}^{n} \mathit{Fields}(\rho_i)$. Hence it holds that $\mathcal{R} \sqsubseteq \rho_1, \ldots, \mathcal{R} \sqsubseteq \rho_n$.*

Because labels can occur at most once in record types, union is defined only on record types with disjoint sets of labels. In order to ensure that our record types don't share any labels, I define a re-labelling operation $\mathit{relabel}$ as follows:

[5]Note that not all labels l_i need to be *used* in subsequent dependent types.

Definition 5 (Relabelling) *Let $\mathcal{T} = \{\rho_1, \cdots, \rho_n\}$ be a set of record types ρ_i. We define a function $relabel$ such that $relabel(\mathcal{T}) = \{\rho_{1[l \mapsto l(\rho_1)]}, \cdots, \rho_{n[l \mapsto l(\rho_n)]}\}$ where $\rho_{i[l \mapsto l(\rho_i)]}$ is the record type created by substituting all labels l in ρ_i with $l(\rho_i)$.*[6]

I assume that the record types on which *record type union* operates don't share any labels, possibly because $relabel$ has been applied to them.

A precise definition of the type system can be found in Appendix B.

4.2 Utterance Representation in TTR

Now that we have the system of TTR at our disposal, I shall devote this section to put it to use by developing a suitable representation of utterances in dialogue that in particular is appropriate to model NSUs.

4.2.1 Contextual Dependence

The most basic feature exhibited by NSUs is perhaps their radical context-dependence. Thus it seems fair to take as the simplest, most minimal requirement of an adequate representation of NSUs some version of the classical context-sensitive approach to meaning common in formal semantics since the seminal work of Lewis (1979), Montague (1974) and Kaplan (1989). Such an approach basically differs from a classical, Tarskian, satisfaction semantics in that context-dependent expressions like e.g. deictics (*I, you, here, now*) are interpreted as functions from a set of contextual indices representing relevant contextual coordinates, to denotations or contents. If NSUs are to be seen in this way, then their analysis amounts to establishing what the necessary contextual coordinates are for each NSU class and to formalising the appropriate function. And this is indeed the task I undertake in the current chapter of this thesis.

As we have seen in Section 3.3.3 of the previous chapter, the notion of meaning as a function from context to content underpins the view of grounding taken by Ginzburg and Cooper (2004). Under this view grounding an utterance—i.e. grasping its content—is identified with the process of finding values for its contextual parameters; while failure to do so results in the need for clarification. This conception leads the authors to adopt a particular HPSG representation of utterances, exemplified in (81) in the previous chapter and repeated here in (114) for convenience. The example represents an utterance of the interrogative sentence *"Did Bo leave?"*.

[6]For families of record types $\delta_1, \cdots, \delta_n$ dependent on $\rho_1, \cdots, \rho_n$ respectively, the relabelling of ρ_i also needs to be applied to any label l from ρ_i on which δ_i depends.

(114)
$$\begin{bmatrix}
\textit{root-cl} & \\
\text{PHON} & \texttt{did bo leave} \\
\text{CAT} & \text{V[+fin]} \\
\text{C-PARAMS} & \left\{ \begin{array}{l}
\begin{bmatrix} \text{INDEX} & \text{i} \\ \text{REST} & \left\{ \textit{spkr}(\text{i}) \right\} \end{bmatrix}, \begin{bmatrix} \text{INDEX} & \text{j} \\ \text{REST} & \left\{ \textit{addr}(\text{j}) \right\} \end{bmatrix}, \\
\begin{bmatrix} \text{INDEX} & \text{k} \\ \text{REST} & \left\{ \textit{utt-time}(\text{k}) \right\} \end{bmatrix}, \begin{bmatrix} \text{INDEX} & \text{t} \\ \text{REST} & \left\{ \textit{precedes}(\text{t,k}) \right\} \end{bmatrix}, \\
\begin{bmatrix} \text{INDEX} & \text{s} \\ \text{REST} & \left\{ \right\} \end{bmatrix}, \begin{bmatrix} \text{INDEX} & \text{b} \\ \text{REST} & \left\{ \textit{named}(\text{Bo,b}) \right\} \end{bmatrix}
\end{array} \right\} \\
\text{CONTENT} & \begin{bmatrix}
\textit{ask-rel} & \\
\text{ASKER} & \text{i} \\
\text{ASKED} & \text{j} \\
\text{MSG-ARG} & \begin{bmatrix}
\textit{question} & \\
\text{PARAMS} & \left\{ \right\} \\
\text{PROP} & \begin{bmatrix}
\text{SIT} & \text{s} \\
\text{SOA} & \begin{bmatrix} \textit{leave-rel} & \\ \text{AGT} & \text{b} \\ \text{TIME} & \text{t} \end{bmatrix}
\end{bmatrix}
\end{bmatrix}
\end{bmatrix} \\
\text{CONSTITS} & \left\{ \begin{array}{l}
\begin{bmatrix} \text{PHON did} \end{bmatrix}, \begin{bmatrix} \text{PHON bo} \end{bmatrix}, \begin{bmatrix} \text{PHON leave} \end{bmatrix}, \\
\begin{bmatrix} \text{PHON did bo leave} \end{bmatrix}
\end{array} \right\}
\end{bmatrix}$$

In (Cooper and Ginzburg 2002) the authors sketch a formulation of the aforementioned analysis of grounding and clarification in terms of TTR. As they point out, the TTR version of this conception has several advantages over its HPSG counterpart. To start with, the set of contextual parameters encoded by means of the feature C-PARAMS in the HPSG representation is intended to represent the domain of the meaning function. However the semantics of typed feature structures cannot by itself denote such a function. Something similar can be said about the content of the utterance. The value of the feature CONTENT in (114) is intended as a representation of the situation theoretic abstract $\lambda\{\ \}.\textit{leave}(b,t)$, but given that typed feature structures do not enable direct use of λ-calculus tools, this is only the *intended* representation.

On the other hand, as I have shown, TTR incorporates all the tools from the λ-calculus, and hence seems more adequate to represent semantic entities. The context-dependent notion of meaning as a function from contextual assignments to full contents can be straightforwardly expressed by a family of record types—a function like (115) where $T_2 : RecType$.

(115) $\lambda(r : T_1).T_2$

The function maps records r of type T_1, representing the type of context required to interpret the utterance, to T_2, which constitutes the type of the utterance content resulting from application to the current context. (116) is a slightly simplified representation of the meaning of the utterance *"Did Bo leave?"* given in (Cooper and Ginzburg 2002).

(116) $$\lambda\left(r : \begin{bmatrix} \mathrm{i} & : & \mathit{Ind} \\ \mathrm{j} & : & \mathit{Ind} \\ \mathrm{c}_1 & : & \mathit{spk}(\mathrm{i}) \\ \mathrm{c}_2 & : & \mathit{add}(\mathrm{j}) \\ \mathrm{t}_1 & : & \mathit{Time} \\ \mathrm{t}_2 & : & \mathit{Time} \\ \mathrm{c}_3 & : & \mathit{utt\_time}(\mathrm{t}_1) \\ \mathrm{c}_4 & : & \mathit{ev\_time}(\mathrm{t}_2) \\ \mathrm{c}_5 & : & \mathit{prec}(\mathrm{t}_2, \mathrm{t}_1) \\ \mathrm{x} & : & \mathit{Ind} \\ \mathrm{c}_6 & : & \mathit{named}(\mathrm{x}, \mathrm{Bo}) \end{bmatrix}\right) . \begin{bmatrix} \mathrm{msg} & : & ?\mathit{Leave}(r.\mathrm{x}, r.\mathrm{t}_2) \\ \mathrm{cont} & : & \mathit{ask}(r.\mathrm{i}, r.\mathrm{j}, \mathrm{msg}) \end{bmatrix}$$

The content of the utterance is a record type with two labels: a label msg whose value is a question message, and a label cont whose value is an illocutionary content. Both values are dependent on record r, which is instantiated to the current context. This is required to contain fields for all the contextual parameters of the utterance.

The TTR representations offered by Cooper and Ginzburg (2002) are concerned with a semantic level of analysis. However, as I have pointed out earlier (in line with Cooper (2005) and Ginzburg (forthcoming)), TTR can be used to underly other aspects of linguistic analysis as well—in the spirit of HPSG *signs*, it can be employed as a full grammatical theory that comprehends phonology, syntax and semantics. Indeed, one could integrate the aforementioned conception of meaning into a *sign*-based type-theoretical representation like (117)—a record type characterising expressions simultaneously in their phonological, syntactic and semantic components, the latter defined as a meaning function like the one in (115).

(117) $$\begin{bmatrix} \mathrm{phon} & : & \mathit{Phon} \\ \mathrm{syn} & : & \mathit{Cat} \\ \mathrm{sem} & : & \lambda(r : T_1).T_2 \end{bmatrix}$$

Some kinds of NSUs, however, provide evidence that the type in (117) is defective in at least one respect, namely in that contextual dependence is limited to semantics. As we have seen in the introduction and in the previous chapter, since the early work of Ross (1969) on sluicing and Morgan (1973, 1989) on short answers, it has been repeatedly noticed in the literature that some NSUs exhibit a limited amount of structural parallelism with their source. Although I have already illustrated this phenomenon in preceding chapters, here I give some additional examples from the literature that summarise the main parallelisms observed. These can more easily be appreciated in case-marking languages like for instance German and Korean, and sometimes also in languages with less rich case systems like Hebrew, which shows morphological case in clitic pronouns, or even English:

(118) a. **German**

A: Hans will jemanden loben.	B: Wen?/#Wem?
`A: Hans wants someone`$_{acc}$ `praise.`	`B: Who`$_{acc}$`?/Who`$_{dat}$`?`
A: Hans wants to praise someone.	B: Who?

b. **English**, from Merchant (2004):
A car is parked on the lawn—find out {whose/#who}.

(119) a. **Korean**, from Morgan (1989):

A: Nu-ka ku chaek-ul sa-ass-ni?	B: Yongsu{–ka/#–rul}.
`A: Who`$_{nom}$ `this book`$_{acc}$ `bought?`	`B: Yongsu`$_{nom}$`/Yongsu`$_{acc}$
A: Who bought this book	B: Yongsu.

b. **Hebrew**, from Ginzburg and Sag (2001):

A: lemi hixmeta?	B: lemoti/#moti.
`A: To-who flater`$_{2nd-sg}$`?`	`B: to-moti/#moti`
A: Whom did you flatter?	B: Moti.

c. **German**, from Schlangen (2003):

A: Wen hast Du gelobt?	B: Den Mann.
`A: Who`$_{acc}$ `have`$_{2nd-sg}$ `you praised?`	`B: The man`$_{acc}$
A: Who did you praise?	B: The man.

d. **English**, from Merchant (2004):
A: Whose car did you take? B: John's/#John.

The examples in (118) show that the form of direct sluices is sensitive to the case requirements of argument-filling source NPs. Similarly, example (119) illustrates case matching between short answers and their argumental *wh*-phrase antecedents.

The existence of partial structural parallelism between NSU and source is perhaps even stronger in CE. As with short answers and direct sluices, a non-sentential clarification must match the source in case, as example (120a) from (Ginzburg and Cooper 2004) shows. Clarification fragments, however, exhibit an intricate pattern of connectivity where phonological parallelism can also play a role. This is demonstrated by example (120b) adapted from (Ginzburg and Cooper 2004).

(120) a. A: Ist dieser Platz noch frei? B: Dieser/#Diesen Platz?
`A: Is this`$_{nom}$ `place still free?` `B: This`$_{nom}$`/#This`$_{acc}$ `place?`
A: Is this place free? B: This place?

b. A: Did Bo leave?
B1: My cousin? ($\rightsquigarrow$ *Are you referring to my cousin with your utternace 'Bo'?*)
B2: Bo? ($\rightsquigarrow$ *Who are you referring to with your utterance 'Bo'?*)

Example (120b) shows that when a clarification NP is interpreted as asking about the meaning of some sub-utterance as intended by the speaker,[7] a non-phonologically parallel clarification fragment like B1 in (120b) conveys a polar question, while a *wh*-question interpretation can only be conveyed by a phonologically identical NP like B2.[8] Thus, an adequate treatment of the readings conveyed by B2 in (120b) will have to refer to phonological information present in the antecedent.

A general conclusion emerges from the data above. This is better formulated as a twofold statement:

(121) a. Utterances can be context dependent above and beyond semantics, i.e. besides content, other formal properties of utterances (e.g. syntactic and/or phonological) can also depend on context. Therefore,

b. the context with respect to which utterances are interpreted must include some degree of structural information, such as some syntactic and phonological properties of previous utterances.

(121b) can be seen as a version of the *Hybrid content hypothesis* formulated by Ginzburg and Cooper (2004), given here in (122):

(122) **Hybrid content hypothesis:** The content which is updated in dynamic semantics consists of structure expressing detailed relations between the content and formal properties (syntax, phonology, etc) of the various parts of an utterance.

[7]In the terminology of Ginzburg and Cooper (2004), this corresponds to a *constituent reading*. According to the authors, B1 and B2 in (120b) can also convey a *clausal* interpretation paraphraseable as *Are you asking if my cousin/Bo of all people left?*. This reading does not seem to be sensitive to phonological information.

[8]This reading can also be conveyed by using a bare *wh*-phrase.

Both (121b) and (122) are connected with the issue of how the context type (T_1 in (115)) should be defined, to which I shall turn in a minute. As to (121a), clearly it calls for a modification of the type in (117). The technical alternative I favour to modify (117) in a way which is consistent with (121a) involves making the λ-abstracted context to take scope over the entire sign-like record type as follows:[9]

(123) $\lambda(r : T_1).\begin{bmatrix} \text{phon} & : & \mathit{Phon} \\ \text{syn} & : & \mathit{Cat} \\ \text{sem} & : & \mathit{Type} \end{bmatrix}$

Like the meaning function in (115), (123) is a family of record types. In this case however the range type does not stand for the content of an expression as in (115), but for a sign-like record type. Functions like (123) from context to sign-like types seem preferable to record type (117) as they potentially allow us to encode complex context-dependencies amongst semantic and structural properties of utterances.

This brings us to the issue of what the nature of the context type should be. Intuitively the context type characterises the *context of use* or the *context of utterance* with respect to which an expression is interpreted. In a dialogue setting context is usually equated with the information state of the dialogue participants, or perhaps more typically with the shared information or *common ground* of the conversants. For now, let us just assume that the context of utterance is defined by a type *IS* characterising the information state of the agents engaged in dialogue. This gives us a general (static) picture of utterances as families of sign-like record types dependent on the current information state (a record r of type *IS*).

(124) $\lambda(r : \mathit{IS}).\begin{bmatrix} \text{phon} & : & \mathit{Phon} \\ \text{syn} & : & \mathit{Cat} \\ \text{sem} & : & \mathit{Type} \end{bmatrix}$

[9] Given the definition of dependent record types and their relation to families of types, a similar effect would be achieved by a dependent record type like the one in (i) , where the domain of the meaning function in (117) is raised to the status of independent field in the record type, on which potentially all subsequent fields can depend. In (i) I have used the notation $r : T_1$ to make the connection with record type (117) more transparent. A more meaningful notation would perhaps be $\text{ctxt} : \mathit{Type}$ or $\text{c\_param} : \mathit{Type}$, as the contextual field would roughly correspond to the HSPG feature C-PARAMS.

(i) $\begin{bmatrix} r & : & T_1 \\ \text{phon} & : & \mathit{Phon}(r) \\ \text{syn} & : & \mathit{Cat}(r) \\ \text{sem} & : & \mathit{Type}(r) \end{bmatrix}$

Given that (123) and (i) are equivalent for all relevant purposes I will, as pointed out before, stick to the formulation in (123), which singles out the context type in a somehow sharper way than (i), and which will prove to be clearer and more convenient for subsequent developments.

Before giving a more precise definition of the type *IS*, it is worth remarking that the TTR formulation in (124), as well as the representations put forward by (Ginzburg and Cooper 2004, Ginzburg forthcoming) and the HPSG signs of (Ginzburg and Sag 2001, Purver 2004b), are concerned with a *static* notion of interpretation. In all these cases the context dependent nature of interpretation is taken into account, but nothing is said about how contextual resources become available, i.e. about how utterances and their interpretation change and create context. This is of course the leitmotiv of the dynamic semantic approaches, and as we are concerned with dialogue and not with isolated sentences, it seems most appropriate to embrace it. The Dynamic Logic approach described in Section 3.4 of the previous chapter aimed at overcoming some of the limitations of a static conception of interpretation. Here I will show how the transition from statics to dynamics can be smoothly achieved using TTR.

4.2.2 From *Statics* to Dynamics

In the approaches of (Ginzburg and Sag 2001, Ginzburg and Cooper 2004, Purver 2004b, Ginzburg forthcoming), which I take as point of departure, dynamics enters into the picture only at the level of dialogue management (read *pragmatics*), which can be seen as a species of *post-module* acting on compositionally built, static representations.

In the Information State Update (ISU) approach to computational dialogue modelling context change is modelled in terms of update processes to the information states of the dialogue participants (see e.g. Traum et al. 1999, Larsson and Traum 2000). Such updates are typically brought about by a set of update rules mainly (but not exclusively) triggered by the observance of speech acts (Matheson et al. 2000, Larsson 2002). For instance, the rule in (125) is used in (Larsson 2002) to update the information state once the user of the system has asked a question:[10]

(125)
$$\left| \begin{array}{lll} \text{RULE} : \textbf{integrateUsrAsk} & & \\ \text{PRECOND} & : & \left\{ \begin{array}{l} \text{speaker} = \texttt{user} \\ \text{latest\_move} = \texttt{ask}(Q) \end{array} \right. \\ & & \\ \text{EFFECTS} & : & \left\{ \begin{array}{l} \text{push}(\text{QUD}, Q) \\ \text{push}(\text{agenda}, \texttt{respond}(Q)) \end{array} \right. \end{array} \right.$$

In recent work, Cooper (2006b) proposes an abstract formalisation of ISU processes in terms of TTR. The central aim of his approach is to offer a declarative treatment of update rules. As in (Larsson 2002), information states are taken to be records. Cooper then formalises update rules as *update functions* like (126):

(126) $\lambda(r : T_i).T_{i+1}$

[10]The rule in (125) has been slightly simplified for readability's sake.

Update functions map an information state record r of type T_i (the type of the current state) onto a record type T_{i+1} (the type of the next state) dependent on r. They are therefore families of record types, i.e. functions from records to record types.[11]

The following function is a (simplified) example of an update function given in (Cooper 2006b), which in turn can be thought of as a TTR formulation of (125):

(127)
$$\lambda r : \begin{bmatrix} \text{is} & : & \begin{bmatrix} \text{agenda} & : & \langle\text{Action}\rangle \\ \text{lu} & : & \begin{bmatrix} \text{spk} = \text{usr} : \text{Participant} \\ \text{moves} \;:\; \{\text{Moves}\} \end{bmatrix} \\ \text{qud} & : & \langle\text{Question}\rangle \end{bmatrix} \\ \text{cond} & : & \begin{bmatrix} \text{q} & : & \text{Question} \\ \text{c} & : & \text{member}(\text{ask}(\text{cond.q}), \text{is.lu.moves}) \end{bmatrix} \end{bmatrix} \left(\begin{bmatrix} \text{is} & : & \begin{bmatrix} \text{agenda} = \text{respond}(r.\text{cond.q}) \mid r.\text{is.agenda} : \langle\text{Action}\rangle \\ \text{qud} = r.\text{cond.q} \mid r.\text{is.qud} : \langle\text{Question}\rangle \end{bmatrix} \end{bmatrix} \right)$$

The domain type of the function in (127) mimics the preconditions of the rule in (125). Cooper uses records with two fields: a field is to represent the information state and another field cond to express conditions that must hold of the information state. The is field postulates that the user was the speaker of the latest utterance. The conditions require that there is a question q such that ask(q) is a member of the set of moves performed by the latest utterance (is.lu.moves).[12] The type that emerges when the function is applied to a record of the type specified by the domain requires that q is added to the qud list and an action respond(q) is added to the agenda.

Thus, by providing an abstract characterisation of update rules as update functions, Cooper shows that TTR is appropriate not only to formalise compositional semantics as we have seen above, but also update processes in the context of ISU dialogue management. However, how these two sides of the same coin are related to each other is not a clear issue. The treatment of compositional semantics offered by Cooper and colleagues is concerned with static meaning, while dynamic matters are brought about by update

[11]It might be objected that the use of types to formalise update is counterintuitive, as typically updates do not involve types but objects. That is, the idea underlying, in some way or other, most dynamic semantics approaches is that updates are mappings between information states (not information state types). In our formulation this would correspond to mappings from records to records. As Cooper points out, however, the use of types is justified if the goal of the formalisation is to *reason* about updates. Update functions can be used to draw conclusions about the *type* of the next information state on the premise of the type of the current state. Since there is typically more than one object of each type, types can be seen as underspecified representations. Consequently, knowledge about the actual information state does not have to be exhaustive in order to draw a conclusion on the type of the next information state, even though the latter of course depends on the precise nature of the former. Note that the definition of the constraint *update* in SDRT for instance follows the same spirit, as it acts on underspecified logical forms, i.e. on descriptions or *types*.

[12]In this setting utterances can realise more than one move.

functions in a dialogue management setting. Although it is worth stressing that the long term goal of Cooper and colleagues is "to obtain a single computationally tractable theory of dialogue management and compositional semantics",[13] the work developed so far has just shown that these two aspects can be approached using the same formal tools. At the current stage, however, there is no fully-fledged account of the role of compositional semantics in the update processes of information states.

Recall that one of the aims of the DL approach summarised in Chapter 3 was precisely to shed light on the interaction between these aspects. The strategy followed in that work involved combining HPSG with Dynamic Logic. Now TTR seems to have all the ingredients we need for an account similar to the HPSG-DL one that is formally more uniform: representations akin to HPSG feature structures that allow Montague-style semantic composition, and at the same time a way of formalising updates by means of update functions.

The approach I take is rather simple: I shall add dynamics to the static representations of Section 4.2.1 by using update functions like (126) not only to represent update rules, but also to represent utterance types themselves, which are seen as a special kind of update rule—as we shall see, one that involves a locutionary event.[14] In particular, I model utterance tokens as mappings between information states, i.e. as functions from records to records, and utterance types as *families of information state types*.

(128) $\lambda(r : IS_n).IS_{n+1}$

Families of information state types, or *IS* families, are mappings like (128) from records of type IS_n to a record type IS_{n+1}, where IS_n, IS_{n+1} are subtypes of the general type of information states IS, which will be defined in a minute. I call IS_n the *domain type* of an utterance type. This specifies the contextual coordinates required for interpretation, and as such will play a prominent role in the formulation of NSU classes. The contextual update determined by an utterance type—its dynamic import—is then specified by its *range type* IS_{n+1}.

4.2.2.1 The *IS* Type: Preliminaries

I will start by assuming that the type of information states can minimally be modelled by a record type with two fields: a field labelled facts, which contains the information that is taken as shared by the dialogue participants; and a field labelled utt, which represents the latest utterance contributed to the dialogue. Both facts and utt take propositions as value.

[13] From `http://www.ling.gu.se/~cooper/records/`.

[14] A preliminary account of this approach has been presented in Fernández (2005).

(129) $$IS \;=_{def}\; \begin{bmatrix} \text{facts} & : & \mathit{Prop} \\ \text{utt} & : & \mathit{LocProp} \end{bmatrix}$$

Propositions The view of propositions I adopt is inspired by the Situation Semantics tradition.[15] In a nutshell, within this tradition propositions are structured objects consisting of a *situation* and a basic SOA or *infon*. Situations are spatiotemporally located *parts of reality*, while basic SOAs classify situations as having certain properties. Basic SOAs are structured objects themselves, made up from a relation and an assignment from entities in the universe to the arguments in the relation, as illustrated by the following SOA:

(130) ⟨⟨ *Rain* ; loc: clapham, time: 8:15 GMT ⟩⟩

Given this, situation s *supports* SOA σ ($s \models \sigma$) if the properties designated by σ hold in s. For instance, a situation s would support the SOA in (130) if s is a situation where it is raining in Clapham at 8:15 GMT. A proposition consisting of a situation s and a SOA σ is true just in case $s \models \sigma$.

I follow Ginzburg (2005)'s TTR modelling of propositions, which exhibits a straightforward correspondence with the sitsemic conception outlined above. Situations correspond to records, while SOAs correspond to record types classifying situations as being of a particular kind (i.e. to situation types labelled s_T). Proposition are then modelled as records of type *Prop*, whose definition is given in (131).

(131) $$\mathit{Prop} \;=_{def}\; \begin{bmatrix} \text{s} & : & \mathit{Rec} \\ \text{s}_T & : & \mathit{RecType} \end{bmatrix}$$

Truth conditions are then defined as follows:

Definition 6 (Truth conditions) *A proposition* $\begin{bmatrix} \text{s} & = & r \\ \text{s}_T & = & \rho \end{bmatrix}$ *is true iff* $r : \rho$.

For instance, the proposition expressed by the sentence *A woman reads a book* is true in a particular situation if that situation can be modelled as a record of the following type:

(132) $$\begin{bmatrix} \text{x} & : & \mathit{Ind} \\ \text{c}_1 & : & \mathit{woman}(\text{x}) \\ \text{y} & : & \mathit{Ind} \\ \text{c}_2 & : & \mathit{book}(\text{y}) \\ \text{c}_3 & : & \mathit{Read}(\text{x}, \text{y}) \end{bmatrix}$$

[15]See e.g. Barwise and Perry (1983) and Barwise and Etchemendy (1990).

Appropriate boolean operations on propositions can be defined in accordance to the general boolean definitions on types given in Section 4.1.1.4. I essentially follow Ginzburg (2005) for this. The definitions can be found in Section B.2 of Appendix B.

Thus facts represents that propositional content that the dialogue participants take to be true and shared amongst them. I will normally use an abbreviated notation to refer to elements involved in the propositions in facts. To avoid clutter, I shall represent fields in the situation type of some proposition in facts simply as in (133a), i.e. jumping through the appropriate paths, which would typically include an extra record type labelled s_T as in the non-abbreviated representation in (133b).

(133) a. $\left[\ \mathrm{facts} = [\ \mathrm{x} : \mathit{Ind}\ ]\ \right]$

b. $\left[\ \mathrm{facts} = \begin{bmatrix} \mathrm{s} = r \\ \mathrm{s}_T = [\ \mathrm{x} : \mathit{Ind}\ ] \end{bmatrix} : \mathit{Prop}\ \right]$

The value of utt is actually also a fact—a fact that we want to single out in the dialogue, i.e. the fact that some utterance has taken place. This fact has a special status because it is typically the antecedent with respect to which NSUs are resolved. In type definition (129), I have specified the value of utt as being of type $\mathit{LocProp}$—a subtype of Prop that stands for the type of *locutionary propositions*. I use this type for a particular kind of propositions that involve situations where a speaker (conventionally labelled as a) utters a linguistic expression (a sign labelled as z). The definition of $\mathit{LocProp}$ is the following:

(134) $$\mathit{LocProp} \quad =_{def} \quad \begin{bmatrix} \mathrm{s} & : & \mathit{Rec} \\ \mathrm{s}_T & : & \begin{bmatrix} \mathrm{z} & : & \mathit{Sign} \\ \mathrm{a} & : & \mathit{Ind} \\ \mathrm{c} & : & \mathit{Utter}(\mathrm{a}, \mathrm{z}) \end{bmatrix} \end{bmatrix}$$

The type Sign is given in (135). I assume that phrases include an extra field labelled dtrs, whose type is a list of signs. [16]

(135) $$\mathit{Sign} \quad =_{def} \quad \begin{bmatrix} \mathrm{phon} & : & \mathit{Phon} \\ \mathrm{syn} & : & \mathit{Cat} \\ \mathrm{sem} & : & \mathit{Type} \end{bmatrix}$$

Thus, I represent locutionary acts as updates of the field utt. From a processing perspective, it seems reasonable to assume that a locutionary act is associated (at least) with each word. Under this incremental view, each word would specify an update of

[16]In fact, the type Sign includes an additional field quant used to provide a very simple account of indefinite NPs. This account, as well as the details of the grammar I assume, are explained in Appendix C.

the information state that (possibly amongst other things) registers the fact that a locutionary act has occurred—the act of uttering the word in question. However, although appealing,[17] semantic incrementality in this fashion poses some non-trivial complications. Perhaps the most obvious one is that for the incremental semantic update to work out it has to be paired with some form of incremental syntactic processing, whose realisation is far from clear. The intricacies of syntactico-semantic incrementality have been the focus of attention of several lines of research, including e.g. Milward (1994) and Kempson et al. (2000). One approach that faces this challenge and that is related in spirit to the perspective I adopt here is the one offered by Poesio and Traum (1997), Poesio and Muskens (1997). The aim of this proposal, formulated using Muskens' compositional DRT (Muskens 1994), is to develop a theory that accounts for both anaphoric accessibility and the intentional aspects related to speech acts performance. For this the authors present a model of language processing in context where the meaning of each word specifies an update of the common ground or *discourse situation*—the situation of the conversation taking place, where information about the speech acts performed and their descriptive content (the *described situations*) is recorded. They use a predicate **utter** to characterise locutionary acts at all levels, from phonemes to full sentences. This leads them to adopt an inferential process whereby syntactic and semantic composition are the result of applying defeasible inference rules.

Although this is an appealing approach, here I opt for simplifying matters by abstracting over this issue. Furthermore I start from the top and look into how the locutionary acts brought about by root utterances are to be represented. The compositional principles by means of which these are constructed are given in detail in the grammar fragment in Appendix C.

I take root utterances to be associated with *illocutionary propositions*, i.e. propositions of type $IllocProp$ as defined below, such that $IllocProp \sqsubseteq LocProp \sqsubseteq Prop$.

(136) $$IllocProp \;=_{def}\; \begin{bmatrix} \text{s} & : & Rec \\ \text{s}_T & : & \begin{bmatrix} \text{z} & : & \begin{bmatrix} \text{phon} & : & Phon \\ \text{syn} & : & Cat \\ \text{sem} & : & Message \end{bmatrix} \\ \text{a} & : & Ind \\ \text{b} & : & Ind \\ \text{c}_1 & : & Utter(\text{a}, \text{z}) \\ \text{c}_2 & : & IllocRel(\text{a}, \text{b}, \text{z.sem}) \end{bmatrix} \end{bmatrix}$$

[17] One of the advantages of incremental syntactic processing is that it has the potential to provide an account of the interpretation of incomplete utterances.

With respect to type $\mathit{LocProp}$, $\mathit{IllocProp}$ incorporates an additional individual who acts as the addressee of the utterance, and an *illocutionary relation* ($\mathit{IllocRel}$) that holds between the speaker, the addressee and the content of the linguistic expression uttered, which is specified to be of type $\mathit{Message}$. I will only be dealing with propositions and questions, and assume that these message types determine the direct or primary illocutionary force of root utterances. This is expressed in the following equivalences:

- $\mathit{Message} \equiv \mathit{Prop} \vee \mathit{Question}$
- $\mathit{IllocRel}(\mathrm{a} : \mathit{Ind}, \mathrm{b} : \mathit{Ind}, \mathrm{m} : \mathit{Message}) \equiv$
  $\mathit{Assert}(\mathrm{a} : \mathit{Ind}, \mathrm{b} : \mathit{Ind}, \mathrm{m} : \mathit{Prop}) \vee \mathit{Ask}(\mathrm{a} : \mathit{Ind}, \mathrm{b} : \mathit{Ind}, \mathrm{m} : \mathit{Question})$

In turn, I assume that the message type of utterances is determined by the structural properties of linguistic constructions—essentially I take declarative sentences to denote propositions and interrogative sentences to denote questions.

Questions Following Ginzburg's work on questions, I take questions to be propositional abstracts. In TTR propositional abstracts can be modelled as functions from records to propositions. Consider for instance the interrogative sentence in (137a). The question denoted by this interrogative can be modelled as the abstract in (137b).

(137) a. Who hates Mia?

b. $\lambda\left(r : \begin{bmatrix} \mathrm{x} & : & \mathit{Ind} \\ \mathrm{c} & : & \mathit{person}(\mathrm{x}) \end{bmatrix}\right) . \begin{bmatrix} \mathrm{s} = r_1 \\ \mathrm{s}_T = \begin{bmatrix} \mathrm{c} = \mathit{Hate}(r.\mathrm{x}, \mathrm{mia}) : \mathit{Type} \end{bmatrix} \end{bmatrix}$

That is, the content of (137a) can be represented as a function from records r of type (138) to propositions like (139) dependent on r.

(138) $\begin{bmatrix} \mathrm{x} & : & \mathit{Ind} \\ \mathrm{c} & : & \mathit{person}(\mathrm{x}) \end{bmatrix}$

(139) $\begin{bmatrix} \mathrm{s} = r_1 \\ \mathrm{s}_T = \begin{bmatrix} \mathrm{c} = \mathit{Hate}(r.\mathrm{x}, \mathrm{mia}) : \mathit{Type} \end{bmatrix} \end{bmatrix}$

I associate with each *wh*-word in a language a particular record type, which I call a *wh*-restrictor. The record type in (138), which I shall refer to as T_{who}, corresponds to the *wh*-restrictor associated with the *wh*-word *who*. While, for instance, the restrictor for *what* would be the record type T_{what} in (140).

(140) $\begin{bmatrix} \mathrm{x} & : & \mathit{Ind} \\ \mathrm{c} & : & \mathit{thing}(\mathrm{x}) \end{bmatrix}$

Some *wh*-restrictors are not as easily characterised as T_{who} and T_{what} however. This is the case for interrogative determiners like *which*, whose restrictor actually emanates from the noun it takes as a complement. We can accommodate this complication by using a more underspecified restrictor like (141), where *N* stands for the type of nominal predicates.

(141) $\begin{bmatrix} \text{x} & : & \textit{Ind} \\ \text{c} & : & N(\text{x}) \end{bmatrix}$

Thus, (unary) *wh*-questions are functions from records r of type T_{wh} to propositions dependent on *r*, where T_{wh} can be thought of as the following disjunctive type covering all *wh*-restrictors associated with *wh*-words:

(142) $T_{wh} \quad =_{def} \quad T_{who} \vee T_{what} \vee T_{where} \vee T_{when} \vee \ldots$

Polar questions are also modelled as propositional abstracts. This is done by using a notion of vacuous abstraction, which allows for 0-ary abstracts. In TTR 0-ary abstracts can be formalised in terms of functions whose domain is the empty record type [ ]. This allows us to treat uniformly the domain type of both vacuous and non-vacuous abstraction as being a record type.[18]

Given these observations, I define the *question domain type* Δ_q as follows:

- The empty record [ ] is of type Δ_q
- If T_{wh} is a *wh*-restrictor and $r : T_{wh}$ then $r : \Delta_q$
- If r and r' are of type Δ_q, then so is $r \cup r'$
- Nothing else is of type Δ_q

The set of inhabitants of the question domain type Δ_q is built up recursively by adding *wh*-restrictors to the empty record type, which is the domain of polar questions. This provides us with a uniform way of accounting for polar questions and n-ary *wh*-interrogatives such as (143a), which denotes the question in (143b). The domain type of this question is the union of the restrictors in (138) and (140).

(143) a. Who drinks what?

[18]As any record is of type [ ], this definition makes 0-ary abstracts and in particular polar questions constant functions from the set of all records. An alternative account could be to consider the domain type of 0-ary abstracts to be the type *Rec*^{0}, whose sole member is the empty record. Since there is no evidence of empirical differences between these two alternatives, the former seems preferable as it allows for a more uniform treatment of interrogatives.

$$\text{b. } \lambda\left(r : \begin{bmatrix} \text{x} & : & \mathit{Ind} \\ \text{c} & : & \mathit{person}(\text{x}) \\ \text{y} & : & \mathit{Ind} \\ \text{c}' & : & \mathit{thing}(\text{y}) \end{bmatrix}\right) . \begin{bmatrix} \text{s} = r_1 \\ \text{s}_T = \begin{bmatrix} \text{c} = \mathit{Drink}(r.\text{x}, r.\text{y}) : \mathit{Type} \end{bmatrix} \end{bmatrix}$$

We are now ready to define the type *Question*, which I shall often abbreviate as Q. As shown in (143), questions are functions from records $r : \Delta_q$ to propositions, i.e. records $r' : \mathit{Prop}$ as defined in (131).

$$\mathit{Question} \quad =_{def} \quad \Delta_q \rightarrow \mathit{Prop} \tag{4.1}$$

I shall use the types WhQ and PolQ (both subtypes of $\mathit{Question}$) for questions whose domain is a non-empty record type and the empty record type, respectively.

4.2.3 Notation

Let us see an example of how an utterance type of a simple declarative sentence like *Leo rides a bike* could be represented with the tools we have so far.

$$(144)\ \lambda\left(r : \begin{bmatrix} \text{facts} & : & \begin{bmatrix} \text{a} : \mathit{Ind} \\ \text{b} : \mathit{Ind} \\ \text{x} : \mathit{Ind} \\ \text{c} : \mathit{named}(\text{x}, \mathit{Leo}) \\ \text{s} : \mathit{Rec} \end{bmatrix} \\ \text{utt} & : & \mathit{LocProp} \end{bmatrix}\right) . \begin{bmatrix} \text{facts} : \mathit{Prop} \\ \text{utt} : \begin{bmatrix} \text{s}_1 & : \mathit{Rec} \\ \text{s}_{T_1} & : \begin{bmatrix} \text{z} & : & \begin{bmatrix} \text{phon} : \texttt{leo rides a bike} \\ \text{syn} = \text{S} : \mathit{Cat} \\ \text{sem} = \text{p} : \mathit{Prop} \end{bmatrix} \\ \text{c}_3 & : & \mathit{Utter}(r.\text{facts.a}, \text{z}) \\ \text{c}_4 & : & \mathit{Assert}(r.\text{facts.a}, r.\text{facts.b}, \text{z.sem}) \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

In (144) the utterance *"Leo rides a bike"* is represented as an *IS* family. The update can take place if the current state r is of the type specified by the domain type of the function. This is required to contain individuals that can be identified as the speaker and the addressee of the utterance and a referent for the proper noun *Leo*, as well as a contextual situation for the proposition conveyed by the utterance. Provided that these contextual requirements are satisfied, the function determines the type of the next information state. The value of utt is updated with the latest illocutionary proposition. This involves a situation where a utters the linguistic expression z corresponding to *Leo rides a bike*, and asserts to b its propositional content. This is abbreviated as p, which stands for the proposition in (145):

(145) $$\mathrm{p} \equiv \begin{bmatrix} \mathrm{s}_2 = r.\mathrm{facts.s} \\ \mathrm{s}_{T_2} = \begin{bmatrix} \mathrm{y} : \mathit{Ind} \\ \mathrm{c}_1 : \mathit{Bike}(\mathrm{y}) \\ \mathrm{c}_2 : \mathit{Ride}(r.\mathrm{facts.x}, \mathrm{y}) \end{bmatrix} \end{bmatrix}$$

As the representation of *IS* families can be rather cumbersome, for the representation of utterance types I shall use some notational shortcuts.

- I will generally ignore those fields, in both domain and range types of the function, that are not relevant for current purposes. When there is no need to zoom into either the domain or the range types, I will use the general type *IS*.
- Within the domain type, often I will need to refer to contextual parameters that will typically be part of the situation type of the propositional content of facts. As I already shown in (133a), for convenience of notation, I will usually refer to them directly under facts, i.e. ignoring the intermediate label s_T.
- Given the complexity of the notation of the utt value within the range type of the function, I will usually simplify it for readability's sake. Depending on current purposes, I shall represent the value of utt with one of the main fields of the situation type of illocutionary propositions. Whenever I need to exclusively refer to the sign in utt—i.e. the value of label z within the situation type of the locutionary proposition in utt—I will use the label utt.z instead of utt to avoid clutter and directly refer to that sign. Alternatively, I will sometimes only use the locutionary predication, or the illocutionary one. Thus, for instance, the label utt in the range type of function (144) could take one of the following abbreviated notations:

(146) a. $$\mathrm{utt.z} : \begin{bmatrix} \mathrm{phon} = \texttt{leo rides a bike} : \mathit{Phon} \\ \mathrm{syn} = \mathrm{S} : \mathit{Cat} \\ \mathrm{sem} = \begin{bmatrix} \mathrm{s}_2 = r.\mathrm{facts.s} \\ \mathrm{s}_{T_2} = \begin{bmatrix} \mathrm{y} : \mathit{Ind} \\ \mathrm{c}_1 : \mathit{Bike}(\mathrm{y}) \\ \mathrm{c}_2 : \mathit{Ride}(r.\mathrm{facts.x}, \mathrm{y}) \end{bmatrix} \end{bmatrix} : \mathit{Prop} \end{bmatrix}$$

b. $\mathrm{utt} : \mathit{Utter}(r.\mathrm{facts.a}, \mathrm{z})$

c. $\mathrm{utt} : \mathit{Assert}(r.\mathrm{facts.a}, r.\mathrm{facts.b}, \mathrm{z.sem})$

I believe these conventions are rather intuitive. We will see them at work in subsequent sections.

4.3 Formal Analysis of NSU

In the previous section, I have put forward a representation of utterances where utterance tokens are transitions between information states and utterance types are families of information state types. In this section, I will use this formal framework to define utterance types for the main NSU classes in the corpus-based taxonomy presented in Chapter 2.

Recall that IS families are functions from the current information state, restricted to be of a particular type IS_n, to the type of the next information state IS_{n+1}, where IS_n and IS_{n+1} are subtypes of the general type of information states IS. As NSUs are highly context dependent, the domain type of these functions has a prominent role in the definition of the NSU classes, as it sets the type of the contextual background required for resolution.

Not all NSU classes require the same contextual background though. From a descriptive point of view, I will distinguish between two main groups of NSUs, which differ in terms of the granularity of the contextual information required for resolution. The NSU classes in the first group are those whose antecedent is a sentential entity (often a complete utterance) taken as a monolith, while the second group includes those NSU classes that are sensitive to deeper components within the internal structure of the antecedent. For instance, Plain Acknowledgement, Plain Rejection, or Propositional Modifier require the presence in context of either a proposition P or a polar question $\lambda(r : [\,]).P$, where P is seen as a monolith—as a proposition in propositional logic for instance. On the other hand, the contextual background of NSU classes like Repeated Acknowledgement, Helpful Rejection, or Direct Sluice needs to be characterised in a more fine-grained manner. In these cases, we cannot merely refer to a contextual proposition P or a polar question $\lambda(r : [\,]).P$, but rather we need to refer to the internal structure of P, typically to some component that acts as a species of *parallel element* in the HOU sense. This component *can be* sentential, but it is often a non-sentential constituent of the antecedent utterance. I shall use the term *Sentential Antecedent* as a binary feature [+/– SA] to distinguish between these two groups of NSUs. Table 4.1 gives an overview of the NSU classes that fall under Acknowledgements, Questions or Answers organised according to this feature. As mentioned in Chapter 2, I will not be dealing with the resolution of Extensions and Completions.

This classification turns out to be convenient because it helps us to distinguish amongst NSU classes according to the complexity of the information state and the mech-

+ SA	– SA
Plain Affirmative Answer	Short Answer
Plain Rejection	Repeated Affirmative Answer
Plain Acknowledgement	Helpful Rejection
Check Question	Repeated Acknowledgement
Sluice (clarification reading)	Direct Sluice
Propositional Modifier	Sluice (reprise reading)
	CE

Table 4.1: [+/– SA] NSUs

anisms required for resolution. Not surprisingly, [– SA] NSUs require a more complex context than [+ SA] NSUs do. As we shall see, however, these two main groups are not completely homogeneous, and thus it will be possible to establish a more fine-grained ranking of NSU classes. I shall now approach each of these groups in turn, pointing out possible aspects that allow further distinctions.

4.3.1 [+ SA] NSUs

As explained above, [+ SA] NSU classes are characterised by having as antecedent a sentential entity that denotes a message. For instance, check questions like *"okay?"* and sluices with a clarification reading like *what?* that merely ask for repetition seem to refer to the antecedent utterance as a whole, without need to access its internal structure. Note furthermore that, in contrast to [– SA] NSU, these NSU classes are realised by standalone lexemes or lexicalised expressions like *"yes", "no way", "right", "possibly", "what?", "okay?"*. Grammatically, I will analyse them as message-denoting lexemes whose content is constructed by operating on contextual information. Now we need to ask ourselves what this contextual information is. For several [+ SA] NSUs a minimal information state that singles out the latest utterance uttered in the dialogue, as introduced in (129), seems to be sufficient. This seems to be the case at least for feedback NSUs, i.e. Plain Acknowledgement, Check Question and Sluice with a clarification reading. Because these NSUs deal with meta-communicative interaction, they are local utterances, i.e. they are typically adjacent to their antecedent. This is of course not surprising as they deal with grounding interaction and establishing mutual understanding is a precondition for advancing a conversation.

Let us start with Plain Acknowledgement, which explicitly indicates that the latest utterance in the dialogue has been understood and integrated appropriately. This can be modelled as an *IS* family that updates facts with the illocutionary predication contributed by the latest utterance.

(147) Plain Acknowledgement

$$\lambda\left(r : \begin{bmatrix} \text{facts} & : & \mathit{Prop} \\ \text{utt} & : & \mathit{IllocRel}(\text{a}, \text{b}, \text{m}) \end{bmatrix}\right) . \begin{bmatrix} \text{facts} = r.\text{facts} \wedge r.\text{utt} : \mathit{Prop} \\ \text{utt} = \mathit{Ack}(\text{b}, \text{a}, r.\text{utt}) : \mathit{IllocProp} \end{bmatrix}$$

Note that (147) represents a simple backchannel that merely shows that the conversation is being followed, without entailing acceptance. For instance, in a dialogue like (148a), the representation in (147) would model B's acknowledgement as grounding (148b) but not necessarily (148c):

(148) a. A: Bartleby's gentleness is the effect of beer.
B: Mhm.

b. *A asserted that Bartleby's gentleness is the effect of beer.*

c. *Bartleby's gentleness is the effect of beer.*

Acknowledgements that also count as acceptances can be represented as functions that update facts not only with the latest move made in the dialogue, but also with their descriptive content, which will typically be a proposition.

(149) Plain Acknowledgement (acceptance)

$$\lambda\left(r : \begin{bmatrix} \text{facts} & : & \mathit{Prop} \\ \text{utt} & : & \mathit{Assert}(\text{a}, \text{b}, \text{p}) \end{bmatrix}\right) . \begin{bmatrix} \text{facts} = r.\text{facts} \wedge r.\text{utt} \wedge r.\text{utt.p} : \mathit{Prop} \\ \text{utt} = \mathit{Ack}(\text{b}, \text{a}, r.\text{utt}) : \mathit{IllocProp} \end{bmatrix}$$

As has been repeatedly observed, acknowledgement and acceptance can be conveyed implicitly by showing continuous attention or by providing an appropriate response (Clark and Schaefer 1989). Sometimes, however. speakers use check questions like *"okay?"* to request explicit feedback from their addressees about the grounding/acceptance status of a previously asserted proposition p. This can be modelled as follows:

(150) Check Question

$$\lambda(r : \begin{bmatrix} \text{utt} : \mathit{Assert}(\text{a}, \text{b}, \text{p}) \end{bmatrix}) . \begin{bmatrix} \text{utt.z} & : & \begin{bmatrix} \text{phon} : \texttt{okay?} \\ \text{sem} = \lambda(r' : [\,]).r.\text{utt.p} : Q \end{bmatrix} \end{bmatrix}$$

Given an antecedent utterance where a proposition p has been asserted, a check question raises the question $\lambda(r' : [\,]).\text{p}$, thereby requesting explicit feedback about the common status of p.

Sluices with a clarification reading that ask for a repetition of the preceding utterance can be represented as in (151). The semantic content of the sluice is a question that could be paraphrased as *"what did you just say/utter?"*. This is constructed by abstracting the linguistic sign (referred to as $r.\text{utt.z}$) from the preceding locutionary proposition (that is, $r.\text{utt}$).

(151) Sluice (clarification reading)

$$\lambda(r : \left[\text{utt} : \mathit{LocProp} \right]).\left[\text{utt.z} : \left[\begin{array}{l} \text{phon} : \texttt{what?} \\ \text{sem} = \lambda(\text{r}' : [\text{x} = r.\text{utt.z} : \mathit{Sign}]).r.\text{utt} : Q \end{array} \right] \right]$$

While meta-communicative [+ SA] NSUs are typically adjacent to their antecedents, other NSU classes have a stronger potential for non-adjacent antecedents. Indeed, although in practice a high percentage of [+ SA] NSUs are local, some of them (typically those that act as answers) have the potential of addressing issues brought about by utterances that were uttered previously to the latest utterance contributed to the dialogue. It is easy to construct examples that show this:

(152) A: Are you going to take part in the city marathon?
B: But wasn't that last month?
A: No, it's going to be in a couple of weeks.
B: Oh, then ***yes/no/probably***.

This kind of evidence is at the core of Ginzburg's analysis of NSUs, who observes that some NSUs (especially short answers) can occur at an unbounded distance from their antecedent. This is Ginzburg's main motivation for incorporating into the representation of context a repository of *questions under discussion*—QUD—as an ordering mechanism that determines the current topic of discussion, as well as potential NSU antecedents. This is in line with several other approaches that model topics as questions, like for instance (Roberts 1996) or the approach of Dekker (2003a,b) reviewed in Chapter 3.

The QUD ordering is usually defined by Ginzburg as a partial order where the latest issue raised takes conversational precedence. In the examples above the first two utterances by A and B respectively introduce two issues q_1 and q_2 in QUD, ordered as $< q_2, q_1 >$ with q_2 taking conversational precedence. Once q_2 is addressed and considered resolved for current purposes, it is *downdated* from QUD and q_1 becomes again the current conversational topic, which can be picked up as antecedent by subsequent NSUs.

To reflect this, I shall modify the *IS* type in (129) by introducing a new field labelled qud whose type is a list of questions that represents the order of questions under discussion, thus adopting an *IS* type that mirrors Ginzburg's DGB. Usually I will be concerned with the first element in the qud list. Therefore I will commonly employ the label qud_1 instead of qud to refer to that first element.

(153) $$\mathit{IS} \;=_{def} \left[\begin{array}{lcl} \text{facts} & : & \mathit{Prop} \\ \text{qud} & : & \langle \mathit{Question} \rangle \\ \text{utt} & : & \mathit{LocProp} \end{array} \right]$$

In Ginzburg's theory of context, questions under discussion are introduced into context by means of illocutionary acts, both Ask and Assert. For Ask moves, this is clear: asking a question q raises q for discussion. Asserted propositions are not necessarily accepted and therefore are also open for discussion. Asserting a proposition p raises the polar question "*whether* p" for discussion (or $\lambda(r : [\,]).\mathrm{p}$ in our TTR notation) . This could be incorporated in our dynamic type-theoretical grammar by defining IS families for illocutionary propositions that would update qud according to their illocutionary force. However, if QUD is to be used as the source of antecedents for NSU resolution, this is not quite sufficient. The following examples show that subordinate questions and propositions, not directly embedded under any illocutionary operator, are also available for NSU resolution.

(154) a. A: Jo wonders/is investigating where Millie got that book from.
B: Oh, ***from that secondhand bookshop in Camden Town.***

b. A: Cris asked me who won this year's Booker Prize.
B: ***Coetzee***.

c. A: Cris forgot/found out Marc's surname.
B: ***Ferrer*** (like everyone in this town).

(155) A: Cris told me that Coetzee won the Booker Prize again.
B1: ***No*** (it was Banville this time).
B2: ***Probably*** (I wouldn't be surprised if he did).
B3: ***Yes*** (the prize was awarded yesterday).

Given this evidence, I opt for associating the projection of questions into context not with illocutionary force, but with locutionary acts involving linguistic expressions that denote either questions, like interrogatives and question-denoting NPs, or propositions like (embedded or otherwise) declarative clauses.[19] [20]

[19]As pointed out by Öle Nielsen (p.c.), question projection and accessibility seem to be subjected to island constraints:

(i) A: Max went for lunch with Tessa, who is wondering where John is.
B: In the library.

(ii) A: Max went for lunch with the guy who is wondering where John is.
B: #In the library.

[20]There is a difference between questions introduced by illocutionary acts and those introduced only by locutionary acts. The former impose an obligation on the addressee to answer the question, while the latter although available as NSU antecedents, don't necessarily have to be answered. This could be taken as motivating a distinction within QUD, perhaps similar to the one adopted by Larsson (2002), or the use of obligations together with QUD (Kreutel and Matheson 1999, Traum 2003). As my main interest is related to the resolution of NSUs, here I don't make this distinction.

The *IS* family of question-denoting expressions is given in (156). Their content type is $Question$ ($\mathrm{sem} : Question$), and such content is projected onto qud ($\mathrm{qud} = \mathrm{utt.s}_T\mathrm{.z.sem} \oplus r\mathrm{.qud}$). The equivalent *IS* family for proposition-denoting expressions is shown in (157). In this case the propositional content ($\mathrm{sem} : Prop$) projects the polar question *whether*(p) onto qud ($\mathrm{qud} = \lambda[\,]\mathrm{.utt.s}_T\mathrm{.z.sem} \oplus r\mathrm{.qud}$).

(156) Question projection of question-denoting expressions

$$\lambda\left(r : \begin{bmatrix} \mathrm{facts} & : & Prop \\ \mathrm{qud} & : & \langle Question\rangle \\ \mathrm{utt} & : & LocProp \end{bmatrix}\right).$$
$$\begin{bmatrix} \multicolumn{3}{l}{\mathrm{qud} = \mathrm{utt.s}_T\mathrm{.z.sem} \oplus r\mathrm{.qud} : \langle Question\rangle} \\ \mathrm{utt} & : & \begin{bmatrix} \mathrm{s} & : & Rec \\ \mathrm{s}_T & : & \begin{bmatrix} \mathrm{z} & : & \begin{bmatrix} \mathrm{sem} : Question \end{bmatrix} \\ \mathrm{a} & : & Ind \\ \mathrm{c} & : & Utter(\mathrm{a}, \mathrm{z}) \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

(157) Question projection of proposition-denoting expressions

$$\lambda\left(r : \begin{bmatrix} \mathrm{facts} & : & Prop \\ \mathrm{qud} & : & \langle Question\rangle \\ \mathrm{utt} & : & LocProp \end{bmatrix}\right).$$
$$\begin{bmatrix} \multicolumn{3}{l}{\mathrm{qud} = \lambda(r : [\,])\mathrm{.utt.s}_T\mathrm{.z.sem} \oplus r\mathrm{.qud} : \langle Question\rangle} \\ \mathrm{utt} & : & \begin{bmatrix} \mathrm{s} & : & Rec \\ \mathrm{s}_T & : & \begin{bmatrix} \mathrm{z} & : & \begin{bmatrix} \mathrm{sem} : Prop \end{bmatrix} \\ \mathrm{a} & : & Ind \\ \mathrm{c} & : & Utter(\mathrm{a}, \mathrm{z}) \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

This applies for instance to the types I have associated with check questions (150) and clarification sluices (151). For example, recall that in (150) I modelled check questions of an assertion of p as conveying the polar question $\lambda(r : [\,]\mathrm{.p}$. According to (156), this question now updates qud as follows:

(158) Check Question (revised)

$$\lambda(r : \begin{bmatrix} \mathrm{qud} : \langle Question\rangle \\ \mathrm{utt} : Assert(\mathrm{a}, \mathrm{b}, \mathrm{p}) \end{bmatrix}). \begin{bmatrix} \multicolumn{3}{l}{\mathrm{qud} = \mathrm{utt.sem} \oplus r\mathrm{.qud} : \langle Question\rangle} \\ \mathrm{utt.z} & : & \begin{bmatrix} \mathrm{phon} : \mathtt{okay?} \\ \mathrm{sem} = \lambda(r' : [\,]).r\mathrm{.utt.p} : Q \end{bmatrix} \end{bmatrix}$$

Note that this is to some extent redundant as an assertion of p, according to (157), would already update qud with $\lambda[\,].p$. This redundancy however seems to be part of the "checking" nature of these questions. Note that, should the addressee decide to explicitly address a previous assertion, the same range of responses are available regardless of whether the check question is asked or not, as shown in (159). A lack of response from B, however, would be rather odd if a check question is asked, while it would most likely be taken as implicit acceptance otherwise.

(159) A: Dimitri's guest will get the bridal suite. (***Okay?***)
B: Okay. / No way.

Sluices with a clarification reading also update qud accordingly. For acknowledgements however, in line with Traum's work on grounding (1994), I assume that there is no qud update.[21]

Let us take stock. We have seen that, while [+ SA] NSUs that are intrinsically related to meta-communicative interaction can generally be dealt with by keeping track of the latest utterance contributed to the dialogue, the resolution of other NSUs that have a greater potential for distant antecedents motivates the introduction of an extra structuring mechanism—a list of questions under discussion. Within the group of [+ SA] NSUs, these are Plain Affirmative Answer, Plain Rejection and Propositional Modifier, which form the class of stand alone lexemes. These three NSU classes are realised by propositional lexemes such as *"yes"*, *"no"* and *"probably"*. I restrict myself here to a rather general account of these items, adopting an approach that closely follows that of Ginzburg and Sag (2001). I assume that these NSU classes require the presence in context of a polar question $\lambda(r : [\,]).P$ currently under discussion, which has been introduced into context either by a question or by a proposition as determined by (156) and (157), respectively.

(160) A: Dimitri's guest will get the bridal suite.
A': Will Dimitri's guest get the bridal suite?
B: ***Yes.*** / ***No.*** / ***Probably.***

The content of these stand-alone lexemes can then be modelled as a relation R which holds of the propositional core of the question under discussion, obtained by applying $\lambda[\,].P$ to the empty record. This is expressed by the following type:

(161) Propositional Lexeme

$$\lambda(r : \left[\ \text{qud}_1 : \mathit{PolQ}\ \right]).\left[\ \text{utt.z} : \left[\ \text{sem} = R(r.\text{qud}_1@[\,]) : \mathit{Prop}\ \right]\ \right]$$

For Plain Affirmative Answers R is the *Identity* relation; for Plain Rejection R is a relation *Neg* sensitive to the polarity of its argument P, such that if P is a positive proposition

[21]Which basically amounts to assuming that acknowledgements do not have to be acknowledged, thus avoiding the circularity of grounding models like (Clark and Schaefer 1989).

Neg(P) corresponds to its negation, while if P is negative *Neg(P)* equals P. This captures the following contrast:

(162) A: Are rats comrades? / Aren't rats comrades?
B: No. ($\rightsquigarrow$ *Rats are not comrades.*).

The types for Plain Affirmative Answer and Plain Rejection are given in (163) and (164), respectively. As for Propositional Modifier, the relation R corresponds to *PropRel*, which subsumes different modalities like *probably* or *possibly*. The type for this NSU class is given in (165).[22]

(163) Plain Affirmative Answer

$$\lambda(r : \left[\ \text{qud}_1 : \mathit{PolQ}\ \right]).\left[\ \text{utt.z} : \left[\ \text{sem} = \mathit{Id}(r.\text{qud}_1@[\,]) : \mathit{Prop}\ \right]\ \right]$$

(164) Plain Rejection

$$\lambda(r : \left[\ \text{qud}_1 : \mathit{PolQ}\ \right]).\left[\ \text{utt.z} : \left[\ \text{sem} = \mathit{Neg}(r.\text{qud}_1@[\,]) : \mathit{Prop}\ \right]\ \right]$$

(165) Propositional Modifier

$$\lambda(r : \left[\ \text{qud}_1 : \mathit{PolQ}\ \right]).\left[\ \text{utt.z} : \left[\ \text{sem} = \mathit{PropRel}(r.\text{qud}_1@[\,]) : \mathit{Prop}\ \right]\ \right]$$

4.3.2 [– SA] NSUs and the Need for Fine-Grained Information

Let us now turn to those NSU classes classified as [– SA] in Table 4.1. As mentioned earlier, these are characterised by maintaining a certain degree of connectivity with their antecedent that cannot be explicated without appealing to fine-grained properties of its internal structure. As we have seen, this kind of NSUs—like Short Answer, Direct Sluice or CE—exhibit structural dependencies with a constituent of the antecedent utterance: a *wh*-phrase in the case of short answers, a quantified NP for direct sluices, and the repeated constituent for repeated affirmative answers, for instance. This means that the antecedent utterance cannot be taken as one piece where only sentential entities are visible, but should rather be represented as an object with further internal structure whose components are available in context. My aim in this section is to determine what these components are, and how they interact with the NSU phrase to account for resolution.

Syntactically, I assume a structure like that of Ginzburg and Sag (2001) and Schlangen (2003), where the NSU is analysed as a message-denoting construction with a single daughter corresponding to the NSU phrase. The content of the construction is

[22]Note that, in order to avoid getting into the semantics of adjuncts (which would require at least a chapter on its own), I am adopting a rather coarse-grained approach here, which does not take into account possible accessibility constraints that might govern the behaviour of these propositional modifiers.

resolved by enriching the semantic contribution of the phrase with contextual information.

Let us start with short answers, which are perhaps the most prototypical NSU class we find in this group. Given the setting developed so far, the semantic resolution of Short Answer is straightforward:

(166) Short Answer

$$\lambda(r : \left[\text{qud}_1 : \textit{WhQ} \right]). \left[\text{utt.z} : \left[\begin{array}{l} \text{sem} = (r.\text{qud}_1@\text{d.sem}) : \textit{Prop} \\ \text{dtrs} = \langle \text{d} : [\text{sem} : \textit{Type}] \rangle : \langle \textit{Sign} \rangle \end{array} \right] \right]$$

As shown in (166), short answers are resolved by functional application of the current QUD to the semantics of the NSU phrase. Hence, on purely semantic grounds, access to the internal structure of the antecedent utterance is not required in this case—even though there is semantic identity between the NSU phrase and the *wh* constituent, this comes in for free courtesy of β conversion. However, as we know, [– SA] NSUs are generally subjected to structural parallelisms that go over and beyond semantics, as exemplified for instance in (119) and (118) for short answers and direct sluices.

In Chapter 3 I have reviewed some proposals as to how this kind of connectivity effects can be accounted for. I will focus here on two of them, namely those that take a constructionist approach (Section 3.3). The first of these proposals is the one offered by Ginzburg and Sag (2001), who deal with structural connectivity by assuming that the context includes a set of salient utterances or SAL-UTTs, and then enforcing several identities between the NSU and an element of that set. Although, as I will argue below, this turns out to be essentially an appropriate strategy, it has a number of non-trivial implications. Firstly, it involves keeping track of a set of potential utterance antecedents for each QUD, which means keeping track of full *signs* (i.e. entities with phonological, syntactic and semantic information) within purely semantic entities like QUDs. Secondly, it implies that any NSU that exhibits connectivity must have an antecedent SAL-UTT. Although the data shows that structural information cannot be completely ignored, the first of these implications may seem a bit too strong for NSUs like short answers and sluices that, as Schlangen observes, seem to be sensitive to subcategorisation requirements only—why should we keep track of full signs if argumental information is all that appears to be required? The second implication, on the other hand, seems to make inappropriate predictions. In A's utterance in (167), for instance, there is no explicit SAL-UTT that can act as antecedent of the sluice uttered by B, and yet case matching with the implicit argument ensues.

(167) A: I borrowed a car.
B: Whose?

As we saw in Chapter 3, an alternative approach is offered by Schlangen (2003), who accounts for the structural parallelism exhibited by certain NSUs by taking into account the role that their fragment phrase plays in the contextually provided predication—contributed, in our case, by the current QUD.

Our TTR account could be easily extended to incorporate a similar approach. Let us see what this would involve. Firstly we would need to assume that the syntax of predicates V includes a function arg_str_V of type $Type \rightarrow Cat$ from the set of V's arguments to a set of syntactic categories. For instance, the argumental structure of the German predicate $loben$ (*"to praise"*) would be determined by a function arg_str_{loben}, such that it assigns to the first argument of the predicate the category $\mathrm{SN}_{[\mathrm{nom}]}$ and to the second argument the category $\mathrm{SN}_{[\mathrm{acc}]}$. Secondly, we should also assume that this function is part of the semantic field of the predicate as in (168), and that it is inherited by the clauses containing that predicate, being therefore present in QUD.

(168)
$$\begin{bmatrix} \mathrm{phon} & : & \texttt{loben} \\ \mathrm{syn} = \mathrm{V} : \mathit{Cat} & & \\ \mathrm{sem} & : & \begin{bmatrix} \mathrm{arg\_str} = \mathit{arg\_str}_{loben} : \mathit{Type} \rightarrow \mathit{Cat} \\ \mathrm{c} \;\; : \;\; \lambda([\mathrm{x} : \mathit{Ind}]), (\lambda([\mathrm{y} : \mathit{Ind}]).\mathit{Loben}(\mathrm{x}, \mathrm{y})) \end{bmatrix} \end{bmatrix}$$

When the content of an argumental short answer is resolved by applying the current QUD to the content of the NSU phrase, the argument structure of the QUD predicate becomes part of the propositional content expressed by the short answer. Also, courtesy of β conversion, the entity denoted by the phrase automatically becomes an argument of this predicate. Now the only thing we need in order to obtain the right syntactic category of the phrase is to apply the argument structure function to its content (as in the syn field of the daughter $\mathrm{syn} = \mathrm{utt.sem.s}_T.\mathrm{arg\_str@sem} : \mathit{Cat}$ below):

(169) Short Answer – subcategorisation strategy

$$\lambda\left(r : \begin{bmatrix} \mathrm{qud}_1 : \lambda(r : \mathit{Rec}). \begin{bmatrix} \mathrm{s} & : & \mathit{Rec} \\ \mathrm{s}_T & : & [\mathrm{arg\_str} : \mathit{Type} \rightarrow \mathit{Cat}] \end{bmatrix} \end{bmatrix}\right).$$
$$\begin{bmatrix} \mathrm{utt.z} : \begin{bmatrix} \mathrm{sem} = (r.\mathrm{qud}_1@\mathrm{d.sem}) : \mathit{Prop} \\ \mathrm{dtr} = \left\langle \mathrm{d} : \begin{bmatrix} \mathrm{sem} : \mathit{Type} \\ \mathrm{syn} = \mathrm{utt.sem.s}_T.\mathrm{arg\_str@sem} : \mathit{Cat} \end{bmatrix} \right\rangle : \langle \mathit{Sign} \rangle \end{bmatrix} \end{bmatrix}$$

Thus this seems a rather elegant way of accounting for syntactic parallelism, which introduces only a limited amount of structural information into the context: the subcategorisation requirements of predicates. Furthermore, the strategy could easily be extended to deal with implicit arguments—all that is required is that these are somehow represented within the argumental structure of the predicates in question.

However attractive, a subcategorisation strategy has a limited scope: as I mentioned in Chapter 3, by definition it can only capture structural connectivity effects that derive from the argumental role of some NSUs. Not all parallelisms exhibited by NSUs and other anaphoric expressions however are due to the argumental role they play within a predication. This was exemplified in Chapter 3 with the dialogue repeated here in (170), which is similar to examples we have seen earlier in (120b). As I pointed out in Chapter 3, the form of this NSU cannot be explicated in terms of the argumental role it plays.

(170) a. A: Leo saw her.
B: Her? / #She?

b. *Who are you referring to with your utterance "her"?*

It is worth looking as well at other anaphoric phenomena that, like NSU, also exhibit structural connectivity. Pronominal anaphora, where pronouns must agree in gender and number with their antecedents, constitutes a well known example of this. Agreement requires morphosyntactic information to be somehow present in context, which has always constituted a problem for dynamic systems like DPL (Groenendijk and Stokhof 1991) and standard DRT (Kamp and Reyle 1993), where context includes only information about referents.

All this suggests that a general theory of anaphoric expressions and their resolution would require the contextual availability of a variety of structural information. Although I do not attempt to develop such a theory here, it is a desirable property of an account of NSUs that it has the potential to be generalised to other anaphoric phenomena. On top of this general requirement, we have in fact seen evidence from NSUs on its own that shows that [– SA] NSUs require the presence in context of suitable antecedents from which we should be able to access a range of different information: sluices exhibit a semantic dependency with their antecedent, some forms of CE demand for their felicity a phonologically identical antecedent, while they all require an antecedent which is morphosyntactically parallel. This suggests that *signs* are the optimal entities to represent [– SA] NSUs contextual dependencies.

Given this, I shall consider that each question under discussion q is accompanied by a list of signs corresponding to topical constituent(s) of the utterance from which q stems, and which act as sub-utterance antecedents or *parallel elements* for [– SA] NSUs. A suitable way to model this is to take qud to be a list of records instead of questions—more precisely, a list of records with two fields: a field labelled que, which encodes the question under discussion proper; and a field I will label top (for *topical*), whose value is a list of signs.

(171) $$\begin{bmatrix} \text{qud} & : & \left\langle \begin{bmatrix} \text{que} & : & \mathit{Question} \\ \text{top} & : & \langle \mathit{Sign} \rangle \end{bmatrix} \right\rangle \end{bmatrix}$$

How do topical sub-utterances become available in context? As we have seen, *wh*-phrases present in *wh*-interrogatives are the typical sub-utterances that act as antecedents for short answers, while any quantified constituent has the potential to be an antecedent sub-utterance for a direct sluice:

(172) A: A man saw something strange in a street nearby.
B: Who? / What? / Where?

Thus the availability of sub-utterance antecedents for short answers and direct sluices can be captured by an incremental approach, whereby *wh*-phrases and quantified constituents incrementally update the information state by introducing potential antecedents into top. This is for instance the strategy adopted in implementations such as (Fernández et al. in press) and (Purver 2004a), and we can model it in TTR by letting the grammar monotonically associate the relevant updates with *wh*- and quantified phrases, as shown in the following simplified representations:[23]

(173) *Wh*-phrase top update

$$\lambda(r : \mathit{IS}).\begin{bmatrix} \mathrm{qud}_1 & : & \begin{bmatrix} \mathrm{top} = \langle \mathrm{utt} \rangle \end{bmatrix} \\ \mathrm{utt.z} & : & \begin{bmatrix} \mathrm{sem} & : & T_{wh} \end{bmatrix} \end{bmatrix}$$

(174) Quantifier NP top update

$$\lambda(r : \mathit{IS}).\begin{bmatrix} \mathrm{qud}_1 & : & \begin{bmatrix} \mathrm{top} = \langle \mathrm{utt} \rangle \end{bmatrix} \\ \mathrm{utt.z} & : & \begin{bmatrix} \mathrm{sem} & : & T_{quant} \end{bmatrix} \end{bmatrix}$$

Now that suitable topical sub-utterances are contextually available, the type for Short Answer can be modified to include the syntactic connectivity between the fragment and the topical antecedent.

(175) Short Answer (revised)

$$\lambda\left(r : \begin{bmatrix} \mathrm{qud}_1 : \begin{bmatrix} \mathrm{que} & : & \mathit{WhQ} \\ \mathrm{top} & : & \begin{bmatrix} \mathrm{sem} & : & T_{wh} \\ \mathrm{syn} & : & \mathit{Cat} \end{bmatrix} \end{bmatrix} \end{bmatrix} \right).$$
$$\begin{bmatrix} \mathrm{utt.z} : \begin{bmatrix} \mathrm{sem} = (r.\mathrm{qud}_1 @ \mathrm{d.sem}) : \mathit{Prop} \\ \mathrm{dtrs} = \left\langle \mathrm{d} : \begin{bmatrix} \mathrm{sem} & : & \mathit{Type} \\ \mathrm{syn} = r.\mathrm{qud}_1.\mathrm{top.syn} : \mathit{Cat} \end{bmatrix} \right\rangle : \langle \mathit{Sign} \rangle \end{bmatrix} \end{bmatrix}$$

[23]More detailed representations are given in Appendix C. Note that in (174) I have used a type T_{quant} as the type of existentially quantified content. This should be understood as an abbreviation. As shown in Appendix C, existentially quantified NPs are characterised by an additional label quant. As this is not important for current purposes, I forgo given more details here.

As we have seen then, the need to access the internal structure of the antecedent utterance seems to be limited to syntax in the case of Short Answer, as semantic identity is a consequence of functional application and β-conversion. The resolution of direct sluices on the other hand cannot do without appealing to the internal semantic structure of the antecedent. Consider the following example:

(176) A: Someone called.
B: Who?

A's utterance is an assertion of the proposition P in (177a), which according to our current setting has updated the information state by introducing the question $\lambda(r : [\,]).P$ and the sign corresponding to the quantified NP in the current qud. In this context B's sluice resolves to the question in (177b), i.e. a question obtained by abstracting the quantified element from the propositional core of the current QUD.

(177) a.
$$\begin{bmatrix} \mathrm{s} = \mathit{Rec} \\ \mathrm{s}_T = \begin{bmatrix} \mathrm{x} & : & \mathit{Ind} \\ \mathrm{c}_1 & : & \mathit{person}(\mathrm{x}) \\ \mathrm{c} & : & \mathit{Call}(\mathrm{x}) \end{bmatrix} \end{bmatrix}$$

b.
$$\lambda\left(r : \begin{bmatrix} \mathrm{x} & : & \mathit{Ind} \\ \mathrm{c}_1 & : & \mathit{person}(\mathrm{x}) \end{bmatrix}\right) . \begin{bmatrix} \mathrm{s} = \mathit{Rec} \\ \mathrm{s}_T = \begin{bmatrix} \mathrm{c} & : & \mathit{Call}(r.\mathrm{x}) \end{bmatrix} \end{bmatrix}$$

In order for the right resolution to ensue, the indices of the *wh*-phrase and the quantified NP (i.e. their labels of type *Ind*) must be identified. In other words, we need to make sure that the sluice is asking about who the individual who called was. This can be expressed with the following IS family:

(178) Direct Sluice
$$\lambda\left(r : \begin{bmatrix} \mathrm{qud}_1 : \begin{bmatrix} \mathrm{que} & : & \mathit{Question} \\ \mathrm{top} & : & \begin{bmatrix} \mathrm{sem} & : & T_{\mathit{quant}} \\ \mathrm{syn} & : & \mathit{Cat} \end{bmatrix} \end{bmatrix} \end{bmatrix}\right) .$$
$$\begin{bmatrix} \mathrm{utt.z} : \begin{bmatrix} \mathrm{sem} = \lambda(\mathrm{d.sem}).r.\mathrm{qud}_1.\mathrm{que}@[\,] : \mathit{Question} \\ \mathrm{dtrs} = \left\langle \mathrm{d} : \begin{bmatrix} \mathrm{sem} \;\; : \;\; [\mathrm{y} = r.\mathrm{qud}_1.\mathrm{top.sem.x} : \mathit{Ind}] \\ \mathrm{syn} = r.\mathrm{qud}_1.\mathrm{top.syn} : \mathit{Cat} \end{bmatrix} \right\rangle : \langle \mathit{Sign} \rangle \end{bmatrix} \end{bmatrix}$$

Note that this analysis of Direct Sluice, like that of Ginzburg and Sag (2001), faces the problem of examples like (167), which lack an explicit sub-utterance antecedent. As I have pointed out earlier, it may seem odd in these cases to assume the existence of a contextually available full sub-utterance. Again, however, the presence of varied structural information in context without explicit linguistic antecedents is motivated independently by other anaphoric phenomena, like deictic anaphora. In languages where nouns have grammatical gender, pronouns with a deictic interpretation display gender agreement with some kind of implicit utterance type associated with the demonstrated object. If I throw a ball at Laia and ask her in Catalan to catch it, I will do so using a feminine pronoun that agrees in gender with the noun one would typically predicate of the object in question, i.e. *pilota* ('ball'):

(179) Agafa-la! / # Agafa'l!
`catch -it`$_{[fem,acc]}$ `/ catch -it`$_{[masc,acc]}$

The mechanisms behind this phenomenon seem to involve some kind of utterance type accommodation process, which I will not attempt to make precise here. This observation however highlights the fact that the availability of sub-utterance antecedents for [– SA] NSUs cannot always be explicated by an incremental update of the context, as it is the case for most short answers and direct sluices. For instance, Repeated Affirmative Answers and Repeated Acknowledgments have a semantic dependency with a sub-utterance antecedent. As mentioned in Chapter 2 and shown in (22) (repeated here as (180)), the focus-ground structure of the antecedent may affect the range of potential topical sub-utterance. However, determining the informational structure of the antecedent is not always an easy task, and when this one is not clearly marked the appropriate sub-utterance cannot easily be singled out a priori.

(180) A: Did you *SHOUT* very loud?
B1: #***Very loud***, yes.
B2: ***Shout***, yes.

For Repeated Acknowledgements this is still less clear cut, as often the parallel sub-utterance seems to be repeated as an almost reflex follow-up (cf. Ginzburg and Cooper 2004):

(181) A: I'll be having chips and beans and a capuccino.
B: ***And a capuccino***, OK.

A perhaps stronger case can be made for CE and Helpful Rejection, whose antecedent can be any constituent of the antecedent utterance:

(182) A: Did Raskolnikov receive a summons?
B1: ***Raskolnikov? / a summons? / receive?***
B2: (No,) ***Nastasya / a letter / WROTE a summons.***

In these cases an incremental strategy that predicts which are the topical sub-utterances in a forward-looking fashion would lead us to introduce a topical sub-utterance for each constituent, which basically would amount to not introducing any topicality at all. Thus, for some [– SA] NSU classes it seems more sensible to make use of a *recoverability strategy*, whereby the topical sub-utterance is not triggered by the antecedent utterance on its own in an incremental fashion, but *recovered* or *accommodated* when the NSU is processed. This is similar to the obtention of discourse referents for plural pronouns in DRT, which are "constructed" by means of different operations whenever they are required to interpret a plural pronoun (Kamp and Reyle 1993, Ch. 4). In the following subsection I propose accommodation rules that allow us to recover antecedents for some [– SA] NSU classes.

4.3.2.1 Some Accommodation Rules

As we have seen in Chapter 3, Ginzburg and Cooper (2004) employ *coercion operations* to construct a context that can explicate the resolution of CE. These operations involve accommodating relevant questions in qud. I will argue that a similar strategy can be used to account for Helpful Rejection. For Repeated Affirmative Answer and Repeated Acknowledgement, however, we do not need to accommodate questions, but only topical constituents. To this end, I propose to use a principle, which I dub *constituent topicalization*, that raises a constituent of the antecedent utterance to the status of topical sub-utterance. Obviously for this to work out we need to have access to the set of constituents of a given utterance—a move made as well by Ginzburg and Cooper (2004). To indicate that a sign is a constituent of another sign I will use the constraint $\mathrm{const}(\mathrm{v}, \mathrm{z})$, where z will label the sign corresponding to the full utterance and v one of its constituents. The general formulation of *constituent topicalization* is given in (183).[24] The precise nature of the accommodated topical sub-utterance will depend on the type of NSU to be processed.

[24]I assume that in the domain type of *constituent topicalization* the current QUD and the content of the latest utterance are co-propositional in the sense of Ginzburg (forthcoming). This ensures that the accommodated topical constituent belongs to the utterance whose content raised the current QUD. Note that as this is formulated, it enforces adjacency of the NSU. This can be solved by taking the value of utt to be a bounded list instead of a single element, in line with (Purver 2004b) for instance.

(183) Constituent topicalization

$$\lambda\left(r : \begin{bmatrix} \text{qud}_1 : [\text{que} : \mathit{Question}] \\ \text{utt} : \begin{bmatrix} \text{s} : \mathit{Rec} \\ \text{s}_T : \begin{bmatrix} \text{z} : \mathit{Sign} \\ \text{v} : \mathit{Sign} \\ \text{c} : \text{const}(\text{v},\text{z}) \end{bmatrix} \end{bmatrix} \end{bmatrix}\right). \begin{bmatrix} \text{qud}_1 : \begin{bmatrix} \text{que} = r.\text{qud}_1.\text{que} : \mathit{Question} \\ \text{top} = \langle r.\text{utt.s}_T.\text{v}\rangle : \langle \mathit{Sign}\rangle \end{bmatrix} \end{bmatrix}$$

Constituent topicalization is only one of the operations that allows for the recovery of NSU antecedents. This is all that seems to be required to account for Repeated Affirmative Answer and Repeated Acknowledgement.

The propositional content of Repeated Affirmative Answer arises by the same means as Plain Affirmative Answers as seen in (163) above, but in this case this is in virtue of the fact that the repeated or reformulated NSU phrase is co-referent with a constituent of the antecedent utterance, which I assume has become topical by application of *constituent topicalization*.

(184) Repeated Affirmative Answer

$$\lambda\left(r : \begin{bmatrix} \text{qud}_1 : \begin{bmatrix} \text{que} : \mathit{PolQ} \\ \text{top} : \langle \mathit{Sign}\rangle \end{bmatrix} \end{bmatrix}\right). \begin{bmatrix} \text{utt.z} : \begin{bmatrix} \text{sem} = \mathit{Id}(r.\text{qud}_1.\text{que}@[\,]) : \mathit{Prop} \\ \text{dtrs} = \langle \text{d} : [\text{sem} = r.\text{qud}_1.\text{top.sem} : \mathit{Type}]\rangle : \langle \mathit{Sign}\rangle \end{bmatrix} \end{bmatrix}$$

The *IS* family associated with Repeated Acknowledgement is the acknowledgement version of (184). This NSU has the same semantic import as Plain Acknowledgement, but, like above, we enforce co-referentiality with a topical sub-utterance of the antecedent, which again has been accommodated by *constituent topicalization*:

(185) Repeated Acknowledgement

$$\lambda\left(r : \begin{bmatrix} \text{qud}_1 & : & \begin{bmatrix} \text{top} = \langle \text{utt.v}\rangle : \mathit{Sign} \end{bmatrix} \\ \text{utt} & : & \begin{bmatrix} \text{z} : [\text{sem} : \mathit{Prop}] \\ \text{v} : \mathit{Sign} \\ \text{c} : \text{const}(\text{v},\text{z}) \\ \text{c}' : \mathit{Assert}(\text{z.sem}) \end{bmatrix} \end{bmatrix}\right). \begin{bmatrix} \text{facts} = r.\text{facts} \wedge r.\text{utt} : \mathit{Prop} \\ \text{utt.z} : \begin{bmatrix} \text{sem} = [\text{c} : \mathit{Ack}(r.\text{utt})] : \mathit{Type} \\ \text{dtrs} = \langle \text{d} : [\text{sem} = r.\text{qud}_1.\text{top.sem} : \mathit{Type}]\rangle : \langle \mathit{Sign}\rangle \end{bmatrix} \end{bmatrix}$$

Finally, Helpful Rejection and CE require accommodating topical constituents *and* questions under discussion. We have already seen that the context of a Helpful Rejection is a polar question q under discussion, stemming either from a query or an assertion. I take its resolved content to be a substitution instance of the propositional core of q.

(186) A: Raskolnikov got a summons. / Did Raskolnivok get a summons?
B: (No,) a letter. ($\rightsquigarrow$ *Raskolnikov got a letter*)

I adopt an approach that has the flavour of the HOU analysis summarised in Chapter 3 and resembles the proposal of Engdahl et al. (2000): I assume that Helpful Rejections presuppose a *wh*-question constructed by abstracting over the substituted parameter, and that they can be resolved in a way akin to short answers—by applying this *wh*-question to the content of the NSU phrase. This resolution involves positing an additional recovery principle that allows us to accommodate onto qud a *wh*-question constructed by abstracting the content of one of its sub-utterances, which in turn become topical. This principle, which I dub *parameter abstraction*, can be formulated as follows:[25]

(187) Parameter abstraction

$$\lambda\left(r : \begin{bmatrix} \text{qud}_1 : [\text{que} : \mathit{PolQ}] \\ \text{utt} : \begin{bmatrix} \text{s} : \mathit{Rec} \\ \text{s}_T : \begin{bmatrix} \text{z} : \mathit{Sign} \\ \text{v} : \mathit{Sign} \\ \text{c} : \text{const}(\text{v}, \text{z}) \end{bmatrix} \end{bmatrix} \end{bmatrix}\right).$$
$$\begin{bmatrix} \text{qud}_1 : \begin{bmatrix} \text{que} = \lambda(r' : \text{top.sem}).\text{P} : \mathit{WhQ} \\ \text{top} = \langle r.\text{utt.s}_T.\text{v}\rangle : \mathit{Sign} \end{bmatrix} \end{bmatrix}$$

Thus in (186), *parameter abstraction* can be applied to the output of A's utterance to raise the question *"What did Raskolnikov get?"* together with the topical sub-utterance *"a summons"*. This would construct the antecedents that can account for the resolution of the helpful rejection.

We have already seen in Chapter 3 how the resolution of clarification NSUs can be explicated by means of Ginzburg and Cooper coercion operations. Here I only focus on exchanges like the following, where the NSU gets the interpretation in (188b):

(188) a. A: Did Bartleby leave?
B: Bartleby?

b. *Who is Bartleby?/Who are you referring to with your utterance 'Bartleby'?*

[25]Here again I assume co-propositionality between the content of the latest utterance and the current QUD in both domain and range types.

In situations like the one above, where a speaker B cannot identify a referent for a contextual parameter, B can still process A's utterance by *contextual existential generalisation*—partially updating her context by existentially quantifying over the unknown parameter:

(189) $\lambda(r : \mathit{IS}).$

$$\begin{bmatrix} \text{qud}_1 & : & \begin{bmatrix} \text{que} = \text{utt.sem} : \mathit{Question} \end{bmatrix} \\ \text{utt.z} & : & \begin{bmatrix} \text{phon} : \texttt{did bartelby leave} \\ \text{syn} = \text{S} : \mathit{Cat} \\ \text{sem} = \lambda[\,].\begin{bmatrix} \text{s} : \mathit{Rec} \\ \text{s}_T : \begin{bmatrix} \text{q} : \begin{bmatrix} \text{x} : \mathit{Ind} \\ \text{c} : \mathit{Named}(\text{x}, \mathit{Bartleby}) \end{bmatrix} \\ \text{c} : \mathit{Leave}(\text{q.x}) \end{bmatrix} \end{bmatrix} : Q \end{bmatrix} \end{bmatrix}$$

This move may sometimes be sufficient for the current purposes. However in situations where B considers this not to be enough (for instance because it does not allow her to reply to A's utterance appropriately), she can coerce her somewhat unsatisfactory information state into one where (i) the problematic sub-utterance becomes topical, and (ii) where the main question under discussion is not whether Bartleby left, but rather who Bartleby is. This can be achieved by the accommodation rule *parameter identification*:

(190) Parameter identification

$$\lambda\left(r : \begin{bmatrix} \text{utt} : \begin{bmatrix} \text{s} : \mathit{Rec} \\ \text{s}_T : \begin{bmatrix} \text{a} : \mathit{Ind} \\ \text{z} : \mathit{Sign} \\ \text{v} : \mathit{Sign} \\ \text{c} : \text{const}(\text{v}, \text{z}) \end{bmatrix} \end{bmatrix} \end{bmatrix} \right).$$

$$\begin{bmatrix} \text{qud}_1 & : & \begin{bmatrix} \text{que} \equiv \text{Q} : \mathit{Question} \\ \text{top} = \langle r.\text{utt.v} \rangle : \mathit{Sign} \end{bmatrix} \end{bmatrix}$$

The question introduced onto qud, which asks about the meaning of the topical sub-utterance as intended by speaker A ($\text{a} : \mathit{Ind}$), is given in more detail in (191) (abbreviated as Q above). Note that the situation of the propositional core of the question is identified with the utterance situation of the latest utterance ($\text{s} = r.\text{utt.s} : \mathit{Rec}$).

(191) $\lambda\left(r' : \begin{bmatrix} \text{x} & : & \mathit{Ind} \end{bmatrix}\right). \begin{bmatrix} \text{s} = r.\text{utt.s} : \mathit{Rec} \\ \text{s}_T : \begin{bmatrix} \text{c} : \mathit{Meaning}(r.\text{utt.s}_T.\text{a}, r.\text{utt.s}_T.\text{v}, r'.\text{x}) \end{bmatrix} \end{bmatrix}$

This rule creates a context where the sign associated with the sub-utterance *Bartleby* is topical, and where a question about the meaning of that sub-utterance is under discussion. The output of (190) is then the input context of the clarification question. As shown in (192), its content is resolved to the accommodated question, raising it explicitly. The NSU phrase is constrained to be phonologically and categorically identical to the topical sub-utterance, which as we saw earlier in (120b) is a prerequisite to obtain a *wh* constituent reading—in this case, the question *'Who is Bartleby?'*.

(192) CE

$$\lambda\left(r : \begin{bmatrix} \text{qud}_1 : & \begin{bmatrix} \text{que} = \lambda(r_1 : [\text{x} : \mathit{Ind}]).\begin{bmatrix} \text{s} = \text{utt.s} : \mathit{Rec} \\ \text{s}_T : [\text{c} : \mathit{Mean}(\text{utt.s}_T\text{.a}, \text{top}, r_1\text{.x})] \end{bmatrix} \\ \text{top} = \text{utt.v} : \mathit{Sign} \end{bmatrix} \\ \text{utt} : & \begin{bmatrix} \text{s} : \mathit{Rec} \\ \text{s}_T : \begin{bmatrix} \text{a} : \mathit{Ind} \\ \text{z} : \mathit{Sign} \\ \text{v} : \mathit{Sign} \\ \text{c} : \text{const}(\text{v}, \text{z}) \end{bmatrix} \end{bmatrix} \end{bmatrix}\right).$$

$$\begin{bmatrix} \text{utt.z} : \begin{bmatrix} \text{sem} = r.\text{qud}_1\text{.que} : \mathit{Question} \\ \text{syn} = \text{S} : \mathit{Cat} \\ \text{dtrs} = \left\langle \begin{bmatrix} \text{phon} = r.\text{qud}_1\text{.top.phon} : \mathit{Phon} \\ \text{syn} = r.\text{qud}_1\text{.top.syn} : \mathit{Cat} \end{bmatrix} \right\rangle : \langle \mathit{Sign} \rangle \end{bmatrix} \end{bmatrix}$$

4.4 Summary and Conclusions

This chapter has provided a formal grammatical analysis of the main NSU classes in the taxonomy put forward in Chapter 2. For this I have employed Type Theory with Records, which allows us to combine sign-based relational structures encompassing phonology, syntax and semantics, with dynamic representations. My proposal has been to formalise NSU types—and utterance types in general—as families of information state types, i.e. functions from the current information state, which sets the contextual background needed for resolution, to the type of the next information state.

I have also pointed out several factors that distinguish amongst NSU classes in terms of the complexity of the contextual information and the mechanisms required for resolution. I have established a main distinction between [+/– SA] NSUs—i.e. between those NSUs whose antecedent is a sentential entity, and those that require access to more fine-grained sub-components of the antecedent. This has provided a ranking of NSU classes, where [+ SA] that deal with meta-communicative interaction are seen as the NSUs that require the simplest information state. All other NSUs need the presence in context of a structuring mechanism like QUD. As for [– SA] NSUs, we have seen that they can exhibit different types of dependencies. Short Answer is the simplest class within this group, exhibiting only a syntactic dependency with its antecedent. Direct Sluice, on the other hand, adds to this the need to access the internal semantic structure of the antecedent utterance. Finally, NSUs like Helpful Rejection and CE require accommodation operations that allow us to coerce the context and recover some material that was not originally present. Besides being analytically attractive, this ranking also seems to have some cognitive plausibility, as it reflects some distinctions in the order of acquisition of NSUs. In particular, Ginzburg and Kolliakou (2004) show that, for English and Greek speaking children, the acquisition of CE and sluicing takes place at a later stage than the acquisition of short answers, which as we have seen are the simplest [– SA] NSUs.

5 Abstract Models for Dialogue Protocols

In this chapter I focus on the formal properties of dialogue protocols, adopting an approach whose core draws on work done jointly with Ulle Endriss and presented in (Fernández and Endriss 2003a,b). After briefly introducing the notion of dialogue protocol, I identify a variety of features that have an impact on the complexity of the dialogue structure. This motivates a hierarchy of abstract models for dialogue protocols based on the expressive power of well-known machine models from formal language theory.

As we shall see, there are some correlations between the hierarchy of protocols, which differ in the information manipulated at each state by an abstract machine model, and the ranking of NSU classes put forward in the previous chapter. This opens the door to interesting connections between dialogue semantics and the theory of computation.

5.1 Communication Protocols

In communication modelling, it is common to distinguish between two main traditions: on the one hand, classical Artificial Intelligence approaches, inspired by ideas originated in analytical philosophy, are built on general models of rational agency, emphasising the role that mental attitudes such as knowledge, belief, desire and intention play in conversational behaviour. This is the perspective adopted by plan-based approaches, most prototypically the BDI (Beliefs, Desires and Intentions) framework (see e.g. Cohen and Levesque 1990, Grosz and Sidner 1990, Sadek 1991). On the other hand, one can identify a parallel line of research, which the present thesis may be seen as an instance of, that follows the work of philosophers like Lewis (1979) and Stalnaker (1978), and that instead of focusing on the intentional attitudes of the interacting agents and the plans that guide their contributions, highlights the public and conventional aspects of communication. Under this perspective, a dialogue can be seen as a *conversational scoreboard* that keeps track of the state of the conversation.

A particular tendency within this latter tradition is the one inspired by the notion of *dialogue game* (Hamblin 1970, Carlson 1983) and the related concept of *adjacency pair* (Schegloff and Sacks 1973, Levinson 1983, Clark 1996). The underlying idea here is that each participant's contribution determines a set of preferred options for follow-up in the dialogue. This relies on the evidence, corroborated by any corpus of real dialogues, that conversations are composed of frequently reoccurring sequences of utterance types. For instance, questions are followed by answers, assertions are either acknowledged, discussed or elaborated upon, and proposals are usually either accepted, rejected, or countered. These *interaction patterns* have inspired a line of research whose object of description is, broadly speaking, the rule-governed behaviour exhibited by dialogue interaction.

In formal approaches, interaction patterns are usually modelled by means of *communication protocols*, i.e. formal constructs modelling public conventions that specify the range of possible follow-ups available to the participating agents. Conventional protocols have been shown to be a powerful descriptive and explanatory means of formalising the *rules of encounter* that characterise coherent interaction both in natural language dialogue, as well as in dialogue between autonomous software agents. In the case of multiagent systems involving autonomous software agents, protocols are simpler and typically more rigid—i.e. they describe the set of *allowed* or *legal* dialogue continuations. In natural language dialogue, on the other hand, protocols should be understood as characterising the range of *preferred* or *less-marked* follow-ups in particular dialogue situations.[1] As such they do not restrict what counts as coherent in a general sense, but rather formalise in simple terms an array of (possibly ranked) unmarked follow-ups that reflects the expectations of the dialogue participants. In this sense, the violation of a protocol can also be informative, as it can be seen for instance as signalling a topic or task change. Thus, in multiagent systems protocols are prescriptive constructs, while in natural language dialogue they tend to be descriptive and can therefore be evaluated in terms of their coverage of the data.

It is worth pointing out that protocols need to be distinguished from *strategies*. While each dialogue participant may be equipped with its own strategy determining its actual responses, a protocol is a social concept common to all participants. That is, protocols are concerned with *shared* conventions, and as such can be thought of as part of the general dialogical competence of speakers. The notion of strategy on the other hand is a *private* one, and in this respect it is clear that a dialogue participant's strategy is shaped by the epistemic notions at the core of plan-based approaches. Protocols, in contrast, are

[1] In this respect it is worth mentioning statistical approaches like e.g. (Taylor et al. 1998, Wright et al. 1999), that extract structural patterns from real data and then model protocols by assigning probabilities to the different options for follow-up. The present approach abstracts from the possibility of statistically ranking allowed continuations.

not meant to determine what to say next, although of course they can be used to guide that decision process by specifying a range of possible next actions, or those actions with the highest probability.

5.2 Rationale of the Chapter

The focus of this chapter is on the formal properties of communication protocols needed to account for different kinds of dialogue phenomena. As we shall see, in dialogue interaction one can identify several features than have an impact on the complexity of the dialogue structure. In the present approach, this variety of phenomena motivates a hierarchy of abstract models for protocols based on the expressive power of well-known machine models from the theory of computation. The basic idea of the hierarchy is that, given a particular dialogue phenomenon, the type of abstract machine that is (usually implicitly) encoded in a protocol or dialogue management system able to handle such a phenomenon can be taken as one relevant dimension according to which the complexity of the system can be classified. My starting point within this hierarchy will be protocols based on deterministic finite automata. From there I will proceed by looking at particular examples that justify either an enrichment or restriction of this initial model, taking inspiration from both natural language dialogue and multiagent systems.

I should stress that, for the subject matter of this chapter, I restrict myself to conventional protocols that characterise the set of possible continuations according to externally observable features, such as the actual utterances taking place in the dialogue. In multiagent systems, protocols designed in accordance with this criterion have recently been put forward by a number of authors (Singh 1998, Pitt and Mamdani 1999a, Colombetti 2000, Jones and Parent 2004). This stands in marked contrast to the BDI or so-called mentalistic approach mentioned above, where legality conditions are explained in terms of the mental attitudes of the agents participating in a dialogue (Cohen and Levesque 1990, FIPA). For the protocols I consider in this chapter, utterances are assumed to occur sequentially. This is perhaps a common assumption in natural language dialogue modelling, whereas multiagent systems research has also tried to address concurrent communication. Also, I am concerned here with *duo-logues*, i.e. dialogues that only involve two participants. Some initial ideas about protocols involving multiple agents can be found in (Ginzburg and Fernández 2005a).

Recall that in the previous chapter I have put forward a ranking of NSU classes on the grounds of the complexity of the information state required for their resolution. Here we shall see that, to some extent, this correlates with the proposed hierarchy of abstract models for dialogue protocols, as these also differ on the information manipulated at each state by an abstract machine model. Thus this correlation opens the door to a novel synthesis between formal language theory and dialogue semantics.

5.3 Protocols as Finite Automata

The starting point of my proposed classification of communication protocols will be protocols based on deterministic finite automata (DFAs). Indeed DFAs have been widely used to represent communication protocols, mostly in the area of multiagent systems (Parsons et al. 1998, Pitt and Mamdani 1999a), although their use is also very common within the spoken natural language community.[2] In this section, after some simple examples, I introduce the class of DFA-based protocols and show how to define the central concept of *possible follow-up* with respect to this model.

5.3.1 Some Simple Examples

Several spoken dialogue systems (e.g. the Nuance Communications' system or the CSLU speech toolkit) use DFA-based models to determine the course of well-formed conversations. One of them is the SRI-Autoroute system (Lewin 1998), which is based on Conversational Game Theory (e.g. Power 1979). In this system finite automata are used to model *games* that represent the conversational rules governing exchanges. The diagram in Figure 5.1 shows a graphical representation of a *confirmation game* given in (Lewin 1998).[3] This automaton characterises what the system (A) can expect from the user (B) in an interaction where A asks B for confirmation of some utterance. In this situation the user is expected to reply either affirmatively with a $reply\_yes$ move or negatively with a $reply\_no$ move. Alternatively, the user may correct the hypothesis of the system by a $reply\_mod$, in which case the confirmation game starts again.

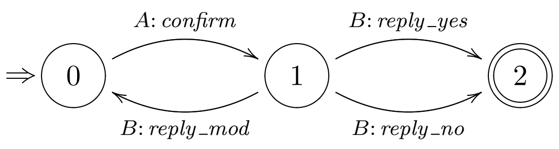

$$\begin{tikzcd} \arrow[r, Rightarrow] & 0 \arrow[r, bend left, "A\colon confirm"] & 1 \arrow[l, bend left, "B\colon reply\_mod"] \arrow[r, bend left, "B\colon reply\_yes"] \arrow[r, bend right, "B\colon reply\_no"'] & 2 \end{tikzcd}$$

Figure 5.1: Confirmation game from (Lewin 1998)

Similar protocols are used to characterise interaction between autonomous software agents. Pitt and Mamdani (1999b) give several examples of automata-based protocols used in the area of multiagent systems. One of them is the so called *continuous update*

[2]A survey of some of these systems is given in Bohlin et al. (1999).

[3]As a convention, when depicting DFAs I indicate their initial state with a double arrow, and their final state(s) with a double circle.

protocol (Figure 5.2), which specifies a class of dialogues between two agents A and B where A continuously informs B on the value of some proposition p.[4]

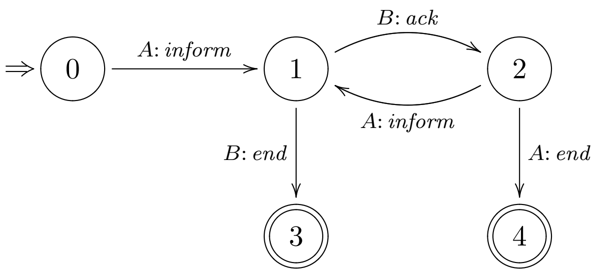


Figure 5.2: Continuous update protocol from (Pitt and Mamdani 1999b)

I will give a precise definition of DFA-based protocols shortly, but for now the notion of what constitutes a possible follow-up in a dialogue conforming to the above protocols should be intuitively clear from the diagrams. In the *continuous update protocol*, for example, immediately after agent A has *inform*ed agent B, the latter can either choose to *acknowledge* that fact or it may decide to *end* the dialogue. However, A cannot continue the dialogue with another *inform* move unless it has received an *ack*nowledgement from B first, and so forth. The protocol could for instance account for dialogues like the following (assuming that eventually either A or B will end the interaction):

(193)	A: [...] we have a family party once a year	$[inform]$
	B: Mm.	$[ack]$
	A: and er we do all the cooking between us and my, my sons	$[inform]$
	B: Mm.	$[ack]$
	[BNC: J8F 144–147]	

DFA-based protocols are not necessarily as simple as the examples I have shown so far. A more sophisticated DFA-protocol is the finite-state model of grounding proposed by Traum (1994). The protocol provides constrains on possible grounding act sequences, allowing for a fine-grained monitoring of the grounding status of an utterance. In this model the utterances that need to be grounded are realised by *initiate* and *continue* acts. Other actions included are (self-)repairs ($repair$), requests for repair (req_repair), acknowledgements (ack) and requests for acknowledgement (req_ack). The automaton also includes an action $cancel$ and a dead state labelled D which is reached when an utterance cannot longer be grounded. Figure 5.3 shows a part of the automaton, the

[4] For this particular protocol, the value of p is intended not to be relevant and therefore I ignore it; any *inform* move uttered by agent A will take us from state 0 to state 1, whatever the content of the transmitted piece of information may be.

part corresponding to the transitions from and to state 1, i.e. a state where a not-yet-grounded utterance has taken place. Only at the final state will the utterance be considered grounded. At state 1 all that is needed for grounding is an acknowledgement; other states are concerned with possible self-corrections, clarification requests, and requests for acknowledgement (our "check questions"), covering dialogues like for instance the following:

(194)	A: I'm gonna charge you, let's say, thirty pence.	$[initiate]$
	A: Okay?	$[req\_ack]$
	B: Mm.	$[ack]$
	[BNC: GYR 660–662]	
(195)	A: Today they can't do it.	$[initiate]$
	B: Mm.	$[ack]$
	A: If you'd got the sack	$[initiate]$
	B: What?	$[req\_repair]$
	A: <*shouting*> if you got the sack	$[repair]$
	B: Yeah.	$[ack]$
	[BNC: HDH 294–299]	

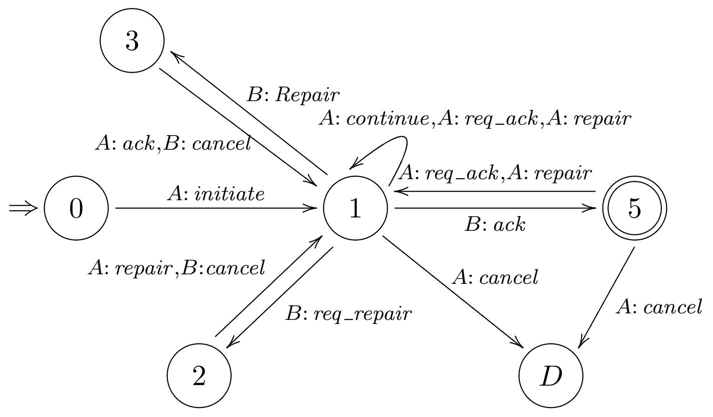


Figure 5.3: Grounding DFA from (Traum 1994)

5.3.2 DFA-based Protocols

Having seen these examples, I shall now precisely define the class of *DFA-based protocols*, i.e. the class of protocols that can be defined in terms of a DFA. As mentioned above, this will provide the starting point for the proposed classification of communication protocols.

Although using a terminology appropriate for current purposes, my definition is in line with standard definitions of a DFA (see e.g. Hopcroft et al. 2001, Lewis and Papadimitriou 1998).

A DFA-based protocol is a quintuple $\langle Q, q_0, F, \mathcal{L}, \delta \rangle$, consisting of

- a finite set of dialogue states Q,
- including an initial state $q_0 \in Q$ and
- a set of final states $F \subseteq Q$,
- a communication language $\mathcal{L}$, and
- a transition function $\delta : Q \times \mathcal{L} \rightarrow Q$.

The elements of the communication language $\mathcal{L}$ are *utterances* (i.e. locutionary actions) and are constructed from a finite set $\mathcal{A}$ of *agents* (or dialogue participants), a finite set $\mathcal{M}$ of *categories* in a broad sense (typically dialogue moves or illocutionary acts, but see below), and a *content language* $\mathcal{C}$. For current purposes, I assume that every utterance has the structure $i : m(c)$ with $i \in \mathcal{A}$, $m \in \mathcal{M}$, and $c \in \mathcal{C}$. In general, at the level of describing abstract models for dialogue protocols rather than concrete instances thereof, I do not put any restrictions on the content language $\mathcal{C}$, i.e. utterances of the form $i : m(c)$ cover any type of utterance. Moreover, for simplicity, in my representations I will typically omit c . As the types of dialogues I consider only involve two participants, $\mathcal{A}$ can be thought of as the set $\{A, B\}$.

Usually, when talking about DFAs in general (i.e. not just in the context of protocols), $\mathcal{L}$ is referred to as an *input alphabet* rather than a communication language. The input alphabet is usually defined as a finite set (Hopcroft et al. 2001). This may seem at odds with our set of utterances $\mathcal{L}$ which, intuitively, could be infinite. While this may indeed be the case, we can always group utterances into a finite number of equivalence classes with respect to the effect they have on the state transition function δ. For any given input state, there are only a finite number of output states the system could move to. These equivalence classes are determined by the elements in $\mathcal{M}$, which is a finite set. As mentioned above, in the context of communication protocols, these are typically dialogue moves, but they can be any other sort of category like e.g. clausal types, or for that matter, NSU classes. Any finite set of categories or types could be suitable.

Representing protocols as DFAs allows for a simple formalisation of the notion of *possible follow-up* at a given point in a dialogue. Given the current dialogue state q, an utterance u constitutes a possible follow-up of the dialogue iff there exists a state $q' \in Q$ such that $\delta(q, u) = q'$ holds. Before a dialogue starts we are in the initial state q_0. The dialogue state then gets updated whenever an utterance is performed, following the transition function δ. A complete dialogue *conforms* to a particular protocol iff it is *accepted* by the DFA representing the protocol in question, i.e. iff each utterance in the dialogue is a possible follow-up and the final utterance leads to a final state in F.

In the context of multiagent systems, where protocols have a prescriptive function, possible follow-ups may also be interpreted as *legal* follow-ups. I will use the latter term when appropriate.

5.4 Shallow Protocols

The last couple of examples of DFA-based protocols given in the previous section (Figures 5.2 and 5.3) deal with grounding behaviour. As such, they include transitions that could be realised by NSUs like Plain Acknowledgement, Check Question or clarification Sluice, that are intrinsically related to meta-communicative interaction. Note however that, given the way in which meta-communicative [+ SA] NSUs have been defined in Section 4.3.1, one does not require the full expressive power of a DFA to characterise interactions where they may be used. This is so because of the local nature of grounding: understanding our interlocutor enough for current purposes seems to be a prerequisite for further interaction (Clark 1996). Recall that this allowed us to formalise meta-communicative [+ SA] NSUs as IS families where the only aspect that stands out from the input state is the latest utterance—that is, as functions simpler than δ above. As we will shortly see in more detail, this corresponds to a different, simpler class of protocols, that arises from restricting the transition function of the basic DFA-based model introduced in the previous section.

The kind of protocols I am referring to is the class of so-called *shallow protocols*, which has recently been introduced in the context of multiagent systems by Endriss et al. (2003). A shallow protocol is a protocol where the appropriateness of an utterance (or, alternatively, what constitutes a possible follow-up) can be determined on the sole basis of the previous utterance in the dialogue. For example, to express that any proposal by agent A must be followed by either an acceptance, a rejection, or a counter proposal by agent B, we may use the following shallow rule, where the *next-operator* $\bigcirc$, familiar from linear temporal logic (Goldblatt 1992), is used to refer to the next turn in the dialogue:

$$A : \mathit{propose} \;\rightarrow\; \bigcirc\,(B : \mathit{accept}\; \vee B : \mathit{reject}\; \vee B : \mathit{counter})$$

In general, a shallow rule is of the form $u \rightarrow \bigcirc(u_1 \vee \cdots \vee u_n)$, where u as well as $u_1, \ldots, u_n$ are utterances. This rule expresses that any of the utterances on the right-hand side of the rule constitutes a legal continuation of a dialogue where the latest utterance has been u. In addition, a special symbol is required to indicate the start of a dialogue in order to be able to provide a rule of the above form specifying the range of legal initial utterances.[5]

This approach can be of great interest in the area of multiagent systems. As shown by Endriss et al. (2003), it is possible to check *a priori* whether a given agent will behave in conformance to a given shallow protocol by inspecting the agent's specification, rather than just observing an actual dialogue at runtime. This is the case, at least, in the sense of the agent being guaranteed not to utter anything *illegal*; guaranteeing that an agent actually utters anything at all turns out to be somewhat more difficult a problem. The ability to check conformance to a protocol not only at runtime, i.e. while a dialogue is actually taking place, but also before entering into an interaction with other agents, allows system designers to build more successful software agents which, for example, may be able to avoid penalties associated with behaviour that is deemed illegal according to the social interaction protocol in place.

Shallow protocols may be understood either in terms of reactive rules of the kind given above or as restricted DFA-based protocols, as in (Fernández and Endriss 2003a). A DFA-based protocol is shallow iff the value of the transition function δ is always uniquely identifiable given only its second argument—the utterance. Thus in a shallow protocol, to determine the state we will end up given an utterance u, we do not need to know the state we come from. A shallow rule tells us that, regardless of that state, whenever utterance u in the left-hand side of the rule takes place, we end up in a state whose possible outgoing utterances are those given in the right-hand side of the rule. On that account, meta-communicative [+ SA] NSUs are utterance types that can appear in the right-hand side of a shallow rule.

A *non*-shallow protocol would be a DFA where two edges with the same label point to two different states. For example, it is common for spoken dialogue systems to include a protocol that allows the system to ask the user to repeat their input at most, say, three times in case of recognition problems. If the problem persists after the third repetition, the system would terminate the interaction and possibly pass on to a human operator. This change of behaviour after the third repetition cannot be modelled by means of a shallow protocol.

In principle, it is always possible to turn a DFA-based protocol into a shallow protocol by renaming any duplicate transitions. In fact, many of the simpler DFA-based protocols to be found in the literature happen to be shallow or could at least be made shallow by renaming only a very small number of transitions (besides Lewin (1998), examples include the protocols proposed by Parsons et al. (1998) and Pitt and Mamdani (1999b)).

In a context where the renaming of utterances labelling problematic transitions is

[5]There are a number of requirements a set of shallow rules forming a protocol need to meet. Notably, there can only be (at most) one rule for any "trigger" u. For full details see (Endriss et al. 2003).

not acceptable, however, shallow protocols really do determine a proper subclass to DFA-based protocols. While there is no standard machine model that corresponds to the class of shallow protocols, they can nevertheless be characterised in terms of the type of grammar that could generate a dialogue following a shallow protocol.

Recall that the regular languages, which are the languages accepted by DFAs, are the languages generated by *right-linear grammars* (Lewis and Papadimitriou 1998). A context-free grammar is called right-linear iff the right-hand side of every rule in the grammar consists of any number of terminal symbols followed by at most one nonterminal. In fact, for any regular language there exists a right-linear grammar generating that language where there is also at most one terminal symbol in any rule. To characterise shallow protocols, an additional restriction to the structure of admissible grammar rules is needed. To account for this additional restriction Fernández and Endriss (2003a) introduce the concept of *shallow-right-linear grammar*. A context-free grammar is *shallow-right-linear* iff (i) every rule is of the form $A \rightarrow bB$ or $A \rightarrow b$, where A and B are nonterminal symbols and b is a terminal symbol, and (ii) for any terminal symbol b there is a unique non-terminal symbol B such that for any non-terminal C other than B there is no rule with bC on the right-hand side. The first part of this definition characterises right-linear grammars, while the second part encapsulates the shallowness condition. Now the languages generated by shallow-right-linear grammars over the alphabet given by a communication language $\mathcal{L}$ are precisely the strings of utterances that correspond to legal dialogues according to shallow protocols.

5.5 Protocols with Memory

The previous section has been concerned with a restriction of the basic model of DFA-based protocols. In this section I will move in the opposite direction and present a first example of a dialogue feature that cannot be modelled by a simple DFA-based protocol in a satisfactory manner. As we shall see, this observation gives rise to an extension of the basic model, which involves adding a *memory component*. This component allows us to store utterances (or abstractions thereof) that may affect the range of possible follow-ups later on in the dialogue.

5.5.1 Protocols that Allow for Subdialogues

As discussed in the previous section, shallow protocols can only deal with interactions that can be characterised in terms of strict adjacency. It is clear, however, that not all dialogues follow patterns that conform to this simple feature. Indeed in the preceding chapter I argued that one of the properties characterising some NSU classes is that their antecedents are not necessarily adjacent utterances. This motivated the introduction of the QUD structure in the information state as a kind of accessibility manager, making available for NSU resolution utterances that were not the latest in the dialogue.

Certainly, responses to questions are a paradigmatic case of utterances that do not necessarily require adjacency. It is in fact not uncommon to find embedded pairs of questions and answers, where a sequence of questions is followed by a sequence of answers, which answer the questions in reverse order. This is exemplified in (196), taken from (Levinson 1983), and the made-up dialogue in (197).

(196)	A: May I have a bottle of Mich?	$[Q_1]$
	B: Are you twenty one?	$[Q_2]$
	A: No.	$[A_2]$
	B: No.	$[A_1]$

(197)	A: Who should we invite?	$[Q_1]$
	B: Should we invite Bill?	$[Q_2]$
	A: Which Bill?	$[Q_3]$
	B: Jack's brother.	$[A_3]$
	A: Oh, yes.	$[A_2]$
	B: OK, then we should invite Gill as well.	$[A_1]$

In abstract terms, a protocol characterising this kind of dialogues would, at the very least, have to be able to keep track of the number of questions asked so that the number of answers can be matched against it. Now this could be modelled by a DFA as long as the level of embeddings is bounded. If the number of questions is not bounded, however, this would require an unlimited amount of memory to be able to store that number. This is not possible with DFA-based protocols, because DFAs have a limited amount of memory, encoded by the fixed set of states of the automaton.

Thus—in theory—the presence of embedded subdialogues (or *insertion sequences* in the terminology of Levinson (1983)) creates a structure that calls for an enrichment of the basic DFA-based model. As we shall see in Section 5.5.3, this can for instance be modelled by adding a *stack* to a DFA. In the example above, questions would get pushed onto the stack, to be then popped by their respective answers. As is well-known, the machine model of a DFA together with a stack corresponds to a *pushdown automaton* (Hopcroft et al. 2001, Lewis and Papadimitriou 1998).

5.5.2 Adding a Memory Component to the Basic Model

Storing questions in the manner suggested above is an example of storing an *abstraction* from the full dialogue history—we only keep track of those parts of the history that are relevant for future follow-ups, and we do so in a convenient format. For embedded question-answer sequences, an appropriate format seems to be a stack, which as we will see below, is the data structure that is often used to implement QUD (Larsson 2002).

DFAs are abstract machines with a limited amount of memory. Adding a (finite) stack amounts to enriching the automaton with an unlimited memory component. Modelling this memory as a stack is just one of many options. Besides stacks, we may consider a variety of *abstract data types* (ADTs) such as, for instance, *queues*, *sets* or *lists*.[6] Following Fernández and Endriss (2003a), I call the set of objects that can be stored in memory the *memory alphabet* (which may or may not be identical to the communication language). Every ADT comes with a set of basic operations ($push(x)$ and pop in the case of a stack) and functions (top to return the top element on a stack, for example). A *configuration* of a memory component is an instance of the ADT used to model that memory component. For example, in the case of a set, a configuration would be a subset of the memory alphabet, while for a stack it would be a string of elements of that alphabet. The *visible* part of a configuration is the part that can be checked using the available ADT functions. In the case of a stack, for instance, only the topmost element is visible. In the case of a set, on the other hand, all elements are visible.

After these preliminaries, I can now define a class of *protocols with memory* as a $\langle Q, q_0, F, A, \mathcal{L}, \mathcal{L}', \delta \rangle$ where

- Q is a finite set of dialogue states,
- $q_0 \in Q$ is the initial state,
- $F \subseteq Q$ is a set of final states,
- A is a finite set $\{\alpha_1, \cdots, \alpha_n\}$ of memory components,
- $\mathcal{L}$ is the communication language,
- $\mathcal{L}'$ is a set $\{L_1, \cdots, L_n\}$ of memory alphabets such that each $L_i \in \mathcal{L}'$ is the memory alphabet for $\alpha_i \in A$,
- $\delta : Q \times \Gamma \times \mathcal{L} \to Q \times \Gamma$ is a transition function, where Γ denotes the set of all possible configurations of the memory components; that is, Γ is a set of tuples $\langle \gamma_1, \cdots \gamma_n \rangle \in \Gamma_1 \times \cdots \times \Gamma_n$, where Γ_i is the set of all possible configurations of memory component $\alpha_i \in A$.

[6]The use of ADTs plays an important role in the definition of the information state in implemented dialogue systems following the ISU approach. For a list of some ADTs useful for dialogue management in such computational systems see e.g. (Traum et al. 1999, Larsson 2002, Bos et al. 2003).

The definition of what constitutes a possible follow-up at a given point during a dialogue for this extended model is very similar to the case of DFA-based protocols. Given the current dialogue state q and the current configuration of the memory components $\gamma = \langle\gamma_1, \cdots \gamma_n\rangle$, an utterance u constitutes a *possible follow-up* in the dialogue iff there exist a state $q' \in Q$ and a configuration $\gamma' \in \Gamma$ such that $\delta(q, \gamma, u) = (q', \gamma')$. A complete dialogue conforms to a particular protocol with memory iff it is accepted by the automaton in question, i.e. iff each and every utterance is a possible follow-up and the last utterance takes us to one of the final states in F.

In the following sections, I will discuss several choices for ADTs as memory components enriching the basic DFA-based model, which are required to account for different dialogue phenomena.

5.5.3 Protocols with a Stack

As we have seen, not surprisingly, not all dialogue structures are satisfactorily captured by a protocol whose machine model corresponds to a simple DFA. In particular, we have seen that the phenomenon of embedded subdialogues is best modelled by adding a finite *stack* to a DFA-based protocol. A stack allows us to store arbitrarily large amounts of information that are accessible in a last-in-first-out (LIFO) manner. Such information can be manipulated by means of the function *top*, which returns the top element on the stack, and the operations *push*(x), which pushes element x onto the stack, and *pop*, which removes the top element from the stack.

Besides *questions under discussion*, other aspects relevant to dialogue management can be successfully modelled with a stack. A clear example are *discourse obligations* (Traum and Allen 1994, Kreutel and Matheson 1999), which constitute a structuring mechanism similar to QUD. Simplifying a great deal, the main idea of the approach is that some utterance types impose obligations on the addressee to respond to these utterances. The assumption is then that dialogue participants will always try to address the topmost element on their obligation stack. Another notion that can be (and indeed usually is) modelled by means of a stack is the *agenda of actions* to be performed by an agent (Traum et al. 1999). Again the assumption is that an agent will always try to perform the action on top of its agenda. In general terms, thus, any repository of information that requires only limited accessibility related to recency—i.e. to the LIFO property of stacks—can in principle be modelled by this kind of datatype.

Expressive power As pointed out earlier already, a DFA together with a stack corresponds to a *pushdown automaton*. As is well known, pushdown automata are strictly more powerful than DFAs (see e.g. Lewis and Papadimitriou 1998). In the current context this means that the class of dialogue protocols (and thereby the class of dialogues conforming to those protocols) that can be specified in terms of a DFA extended by a stack strictly includes the class of protocols (and dialogues) that can be specified using a DFA alone.

Recall that [– SA] NSUs are distinguished from [+ SA] NSUs by the complexity of the elements that are stored in QUD. The latter do with single questions, while the former require pairs of questions and salient constituents thereof. Although this makes the memory alphabet more complex (as it is not made up of single elements but pairs of questions and constituents), at the level of abstraction I am adopting here this does not demand further expressive power beyond that provided by a standard stack.

While these observations regarding the expressive power of different protocol models are easy exercises from a computation-theoretic point of view, I believe that they offer an interesting and novel perspective on dialogue modelling. As mentioned above, the type of abstract machine that is encoded in a protocol or dialogue management system is certainly one relevant dimension according to which we can classify the complexity of such a system.

5.5.4 Protocols with a Stack of Sets

So far I have argued that in dialogues like the one in (197) where several questions are posed in sequence, it seems reasonable to use a protocol where the last question asked takes precedence (i.e. it is the first one to be addressed), and that such a protocol can be modelled by a pushdown automaton. In this section, I will discuss a generalisation of this model that makes accessibility more flexible.

As some authors have noticed (Asher 1998, Ginzburg forthcoming), when successive queries are asked by a single speaker, the simple kind of protocol discussed in the previous section would not always correctly account for the data. This is illustrated by the following example (adapted from Asher 1998):

(198) A: Where were you on the 15th? $[Q_1]$
Do you remember talking to anyone after the incident? $[Q_2]$

B1: I didn't talk to anyone. $[A_2]$
I was at home. $[A_1]$

B2: I was at home. $[A_1]$
I didn't talk to anyone. $[A_2]$

Dialogues as the one above show that when two or more questions are uttered in succession by the same speaker, the respondent is sometimes allowed to answer them in any order.[7] When this is the case, a protocol based on a DFA plus a stack would not be appropriate to handle this phenomenon.

[7]In such dialogues, the questions under discussion are in what has been called a *coordinate structure*

Larsson (2002) gives some examples using short answers that point in the same direction. As he notes, “a stack-like structure would suggest that [example (199a)] should be easily processed by dialogue participants, but in fact it is very unclear what B means”. Similarly, assuming that the questions in (199b) are organised as a stack “suggests a very unintuitive interpretation of B’s answer, where 10:30 is the time when B is coming back and 11:30 is the time when B is leaving”.[8]

(199) a. A: Where are you going? $[Q_1]$
Where is your wife going? $[Q_2]$
B: Paris. $[A_?]$
London. $[A_?]$

b. A: When are you leaving? $[Q_1]$
When are you coming back? $[Q_2]$
B: Ten thirty and eleven thirty. $[A_?]$ & $[A_?]$

In fact, in Ginzburg’s theory the questions currently under discussion form a *partially ordered set*. It should be noted, however, that his proposed model does not actually make use of the full expressive power of a partially ordered set. This is so because questions can only be popped from the top—that is, questions can be addressed if they are topmost (in a more flexible sense than in a simple stack); we do not have access to arbitrary elements further down the order, and we cannot insert elements at any position.

In terms of the hierarchy of protocols proposed here, such an architecture can be modelled using a DFA together with a finite *stack of sets*. The questions under discussion that currently have maximal conversational precedence are those in the top set of the stack. Now, adding a new question strictly above the currently maximal ones corresponds to pushing a singular set containing only that question onto the stack. To add a new question *next* to the currently maximal questions, on the other hand, we first pop the top set off the stack, then insert the new question into that set, and then push the new set back onto the stack. To delete a question (with maximal precedence), we first pop the top set off the stack, then delete that question from the set, and finally push the remaining set back onto the stack—unless that set is empty (i.e. unless there has been only a single topmost question). A delete operation will fail in case the question given as a parameter is not a member of the top set.[9]

(Asher 1998), with none of them taking precedence over the others.

[8]The quotes and the examples are from (Larsson 2002) p. 154.

[9] Larsson (2002) proposes to model QUD as an *open-stack*, i.e. a structure that retains the stack order but possesses set-like properties. For instance, as a set, an open stack cannot contain repeated elements. This makes $push(x)$ a complex operation that involves first checking whether x is a member of the open-stack; in the affirmative case, x is deleted prior to being pushed on the top position. Even though non-topmost elements can be accessed and deleted, questions can only be added at the top. Contrary to our *stack of*

An interesting issue, which I will discuss here only very briefly, is what causes a question to be inserted into the top set of the stack or, alternatively, to be pushed onto the currently maximal set. A simple hypothesis one could make is that the first operation takes place when successive queries are posed within a single turn (as in the examples above), while the second one is executed when a different speaker replies to a question with another question (as in the example in Section 5.5.2). The following dialogue, taken from (Ginzburg forthcoming), however, suggests that successive querying within a single turn does not always imply that the questions enjoy equal status:

(200) A: Who will you be inviting? And why?
B: Mary and Bill, I guess.
A: Aha.
B: Yeah, (because) they are very undemanding folks.

Notice that here the first question asked seems to take precedence over the last one—only after the first question is answered does the second question get addressed. Although a reply along the lines of *"I'd like to have a quiet dinner, so Mary and Bill I guess"* would also be possible, the order of answers in (200) seems to be the least marked option in this situation.[10] This suggests that the order in QUD is not determined solely by conventional means (i.e. in terms of the order of occurrence in the dialogue or the speakers of the questions), but is also guided by semantic relations that may hold between its elements. This is of course noted by Asher (1998) and Ginzburg, who explicate the differences between examples (197) and (200) in terms of the different relation that are said to hold between the questions: *coordination* in the first case, and *query-extension* in the second one.

There is not much to say about this from the abstract point of view I take here, besides noting that complex relations between the elements of the content language $\mathcal{C}$ would have to be encoded as part of the definition of the transition function δ.

Expressive power It turns out that, from a computational point of view, the model of a DFA enriched with a stack of sets is in fact not more expressive than that of a pushdown automaton (with a simple stack). This may be seen by considering that, given a DFA with

sets, however, this model cannot account for bunches of elements that are "at the same level" while being ordered with respect to other elements. As it will be shown for sets in Section 5.5.5, the expressive power of this model is not higher than that of a DFA.

[10]In early work, Ginzburg uses examples like (200) above to motivate a modification of his QUD-*update* protocol, namely the addition of a new operation ("+QUD-FLIP"), which pushes a question *under* the maximal element in QUD, i.e. between the topmost element and the rest of the stack. In the abstract model proposed here this would correspond to first popping the top element off the stack, then pushing the new question, and finally pushing the former top element back onto the stack.

a stack of sets using the memory alphabet $\mathcal{L}'$, we can construct a pushdown automaton that uses the power-set of $\mathcal{L}'$ as its memory alphabet and that accepts the same inputs as the original automaton.

An alternative way of constructing a pushdown automaton that is equivalent to a given DFA with a stack of sets would be to extend the memory alphabet by a special "separator" symbol. Now all elements in the stack between two separators would be considered a set. Each state in the original DFA would have to be replaced with a sub-automaton to simulate checking for membership and deleting elements from the top set.

In either case, the simulation would result in an exponential blow-up of the model, either with respect to the memory alphabet or with respect to the number of states. Therefore, in practice, the model of a DFA-based protocol with a stack of sets may often be preferred over protocols with a simple stack.

5.5.5 Protocols with a Set

So far we have seen abstract models for protocols that are related either to strict adjacency, like shallow protocols, or to different degrees of recency, like protocols that make use of a stack or a stack of sets. Sometimes however, determining what constitutes an appropriate follow-up with respect to a protocol can be related to aspects that are independent from the order in which inputs have been received. An example of this order-independent model is the so called *blackboard architecture*, which has been used in the context of multiagent systems to formalise protocols inspired by work on argumentation in dialogue modelling. Such protocols have recently been used to model different types of dialogues (such as negotiation dialogues or persuasion dialogues) between software agents (Amgoud et al. 2000). Central to this approach is the notion of *commitment store* originally introduced by Hamblin (1970). For example, *asserting* a proposition amounts to placing that proposition into one's commitment store. A *retract* move would then be considered *legal* only if the corresponding formula can be found in the agent's commitment store (and would itself cause the respective formula to be deleted again). Similarly, to model the rule that an agent may only challenge a proposition that has previously been asserted by its opponent, we may stipulate that a *challenge* move is only legal in a situation where the proposition that is being challenged has previously been added to the commitment store.

This kind of blackboard architecture may be modelled in terms of a DFA-based protocol extended with a memory component that is structured as a *set*. The ADT operations available when working with this kind of memory component are *insert*(x) to insert element x into the set and *delete*(x) to remove it again. The central function available is *member*(x) to test whether x can be found in the set (Aho et al. 1983). Any utterances that may affect the legality of utterances later on in a dialogue would be stored in this set. As pointed out above, this kind of architecture requires us to abstract from the order in which utterances occur. We can only keep track of the fact that a given utterance either has or has not been uttered in the past.

Besides modelling argumentation-based dialogue, another application of this model would be protocols involving *social commitments*, like the agent interaction protocols in (Singh 1998, Colombetti 2000) and the approach of Matheson et al. (2000), who combine social commitments with discourse obligations. An example, taken from Singh (1998), would be that an agent, if asked for a price quote by different agents (possibly only within a particular time interval), must always reply with the same quote. This social commitment may be modelled by storing the first price quote in the set component (possibly together with the query and the time of the first query). Any subsequent reply to a price quote may then be checked against the contents of the set to detect potential violations of the protocol.

Ginzburg's FACTS, which I have made use of to formalise the information state in the previous chapter, is usually thought of as a set as well, whose elements in this case do not represent the commitments or beliefs of agents, but rather what has been established for the sake of the conversation.[11] One may want to design protocols where, for instance, questions are only appropriate follow-ups whenever their answers are not members of the set of FACTS.

Expressive power It is interesting to note that adding a set to a DFA does in fact not increase expressive power, because the range of all possible configurations of the set component could be encoded into a larger DFA. This is the case, because we are working with a *finite* memory alphabet. The set of possible configurations of the blackboard is the power-set of the memory alphabet, i.e. it is also finite. It is therefore possible to transform the original DFA by introducing a new state for every pair of an original state and a configuration of the blackboard. If we arrange the transition function accordingly, we obtain a new DFA (without explicit memory component) that corresponds to the same protocol as the original automaton. However, such a construction would result in an exponential blow-up of the set of states; that is, in practice, a blackboard architecture can have great advantages over a simple DFA-based protocol.

In the literature on argumentation systems, each agent is usually equipped with its own commitment store. Again, while this is a convenient means of representation, working with a DFA with more than one set does also not increase the expressive power of the model, because the range of all possible configurations of the memory components

[11] There may be reasons to postulate some kind of order within the set of facts. See (Ginzburg 1997) for some ideas in this direction.

could be encoded explicitly within a larger DFA.

5.5.6 Protocols with a List

Besides stacks and sets, another important ADT is the *list* data type. Like a stack, a list can be used to store a string of elements of the memory alphabet, but, while a stack only allows access to its top element, in a list all elements are visible and can be manipulated. The most important ADT operations for lists are *insert*(i, x) to insert element x at position i and *delete*(i) to remove the element stored at position i from memory, while the function *retrieve*(i) can be used to check the value of that element (Aho et al. 1983).

Systems providing access to (parts of) the dialogue history explicitly in order to check the legality of an utterance may be modelled as DFA-based protocols together with a finite *list* (by appending utterances to the end of the list as they occur in the dialogue). This architecture allows us to keep track of relevant utterances and the order in which they occur. In particular, a list-based representation enables us to access any of the elements stored in memory at any time, and not just, say, the element inserted last (as for stacks).

An architecture of this sort is used in (Ginzburg and Fernández 2005b) to account for acceptance moves in multi-party dialogue, where the asserter of a proposition p can consider p accepted by the audience only when it has been accepted by all addressees. We model this by means of a list of 'Moves' that keeps track of the utterances that occur in the dialogue. A proposition p is considered accepted if the list of moves contains an $assert(p)$ move followed by acceptance moves by all addressees participating in the dialogue. Thus we need to access several moves stored in memory, and to make sure that the assertion occurred prior to the acceptances, we also need to keep track of the order in which moves occurred.

Keeping track of the full dialogue history encoded as a list gives rise to protocol models less abstract than the ones I have so far discussed, but also more powerful. In fact, according to the thesis underlying the conventionalist approach to communication protocols in multiagent systems research, given the full dialogue history, it should—in principle—always be possible to specify *any* conditions pertaining to the legality of an utterance. Of course, in computational terms, this model is also the most costly one. Storing the entire dialogue history may not always be feasible. Also, simply storing the history without making suitable abstractions (as in the previous examples) will often be too rich a mechanism for designing concise protocols. In fact, many approaches that appeal to lists don't make full use of the expressive power of this ADT.

Expressive power The machine model of a DFA enriched with a finite list is equivalent to a *Turing machine* (Hopcroft et al. 2001, Lewis and Papadimitriou 1998). This follows immediately from the fact that the list can be used to store the tape, while the DFA can be used to implement the control component of a Turing machine. An alternative way of showing this equivalence would be to build on the fact that a list can be used to encode two stacks and a pushdown automaton with two stacks is equivalent to a Turing machine (Lewis and Papadimitriou 1998).

As a final remark, it is well known that Turing machines with multiple tapes can be simulated by a single-tape Turing machine (Lewis and Papadimitriou 1998). Therefore, working with a protocol with several list components would not increase expressive power any further.

5.6 Summary and Conclusions

In the preceding sections, I have presented a variety of interesting features of dialogue as they occur either in natural language interaction or in the context of multiagent systems. These features have given rise to a number of different abstract models for dialogue protocols, based on the machine model of a deterministic finite automaton. I have enriched this basic model with memory components modelled as different abstract data types. In one case, I have also shown that a restriction of the basic model can have useful applications. Table 5.1 gives an overview of the various protocol models I have discussed, together with a selection of representative examples.

I should emphasise that the protocols with memory introduced in this chapter are *abstract* models that are intended to capture characteristic features of particular classes of dialogues. In most cases, the full power of the theoretical model will not be necessary to account for actual human-human dialogues. For instance, pushdown automata, which I have used to model the phenomenon of subdialogues, are only strictly more expressive than simple DFAs if the size of the stack is not bounded. However, one may argue that humans will hardly ever use more than a relatively small number of levels of embeddings. In the case of communicating software agents this bound may be higher, but for practical purposes it seems still very reasonable to work with an upper bound on the number of elements that can be stored in the stack. For all the ADTs that I have discussed in this chapter, if the number of elements that can be stored is bounded, then a DFA equipped with a memory component is not more expressive than a simple DFA. It is worth stressing that this does not disqualify the idea of working with memory-enriched protocols, however. A simple DFA together with an ADT that structures relevant information in an appropriate manner can be much more useful, from both a practical and a theoretical point of view, than a single large and possibly rather cumbersome DFA.

Abstract model	**Examples**
Shallow rules ($\subset$ DFA)	- simple communication protocols in multiagent systems (Endriss et al. 2003)
Finite automaton (DFA)	- simple communication protocols in multiagent systems (Pitt and Mamdani 1999b) - conversational games (Lewin 1998) - finite-state model of grounding (Traum 1994)
DFA + set (= DFA)	- commitment store in argumentation (Hamblin 1970, Amgoud et al. 2000)
DFA + stack (= pushdown automaton)	- (simplified) questions under discussion (Ginzburg 1996) - (some models of) discourse obligations (Kreutel and Matheson 1999)
DFA + stack of sets (= pushdown automaton)	- questions under discussion (Ginzburg 1996)
DFA + list (= Turing machine)	- explicit representation of dialogue history - acceptance moves in multi-party dialogue (Ginzburg and Fernández 2005b)

Table 5.1: Abstract models of dialogue protocols

6 Automatic Classification of Non-Sentential Utterances

In the approach I have presented in this thesis, the main NSU classes in the taxonomy put forward in Chapter 2 are associated with particular utterance types that constrain their resolution. Consequently, in the processing of NSUs a necessary first step towards their resolution is the identification of the right NSU class, which will determine the appropriate resolution procedure.

Although the inter-annotator agreement on the NSU classification task (76% *kappa* score, as seen in Section 2.3.2 of Chapter 2) shows that humans can successfully distinguish between the proposed NSU classes, it is surely too optimistic to assume that this is an easy task for any system that needs to process NSUs automatically. Indeed, because of their fragmentary form and their highly context-dependent meaning, NSUs are often potentially ambiguous, even if humans can intuitively distinguish them in the majority of cases. Consider for instance example (201). An NSU like B's in (201a) can be understood either as a clarification question or as a repeated acknowledgement, depending on whether it is uttered with raising intonation or not. In (201b), on the other hand, the NSU is clearly a short answer, while in (201c) it plays the role of a filler. Yet in the context of (201d) it will most probably be understood as a helpful rejection.

(201) a. A: I left it on the table.
B: ***On the table***

b. A: Where did you leave it?
B: ***On the table***

c. A: I think I put it er. . .
B: ***On the table***

d. A: Should I put it back on the shelf?
B: ***On the table***

In this chapter I concentrate on the task of automatically classifying NSUs, which I approach using machine learning techniques. My aim in doing so is twofold: firstly, I expect to gain interesting insights into the data while achieving a good performance on the classification task; secondly, I seek to develop a classification model whose output can be fed into a system (be it a full dialogue system or, for instance, an automatic dialogue summarisation system) that is then able to resolve the content of NSUs on the basis of the input provided by the classifier.

6.1 Preliminaries

Before turning to the machine learning experiments and the results obtained, I will first point out a methodological simplification adopted with respect to the partitioning of the data, then present the general strategy followed by the feature annotation procedure, and finally give a brief description of the machine learners used.

6.1.1 A Methodological Simplification

The data used in the experiments I report in this chapter was selected from the classified corpus of NSUs following a methodological restriction, which concerns some question-denoting NSUs. I adopt a form-based class 'Sluice', which as mentioned in Section 2.4, cuts across two classes in the taxonomy—Direct Sluice and CE. I classify under 'Sluice' all *wh*-question NSUs, thereby conflating direct and reprise sluices. In the taxonomy of Chapter 2 reprise sluices are classified as CE. In the data set used in these experiments, however, CE only includes clarification fragments which are not bare *wh*-phrases.

The first set of experiments I will present (Section 6.2) addresses the task of classifying all NSU classes on the basis of data selected and partitioned following this restriction. As we shall see, the results obtained are highly positive and provide the ground for a wide coverage NSU classification system. Part of these results have been presented in (Fernández et al. 2005b).

Adopting a form-based class Sluice simplifies the classification task of question-denoting NSUs. However, this methodological simplification seems acceptable only if the distinction between different sluice interpretations can still be recovered. In a second experiment (Section 6.3) I investigate this issue, namely the disambiguation of sluice interpretations, again obtaining very encouraging results. An earlier version of the results pertaining to this second experiment have been presented in (Fernández et al. 2004, 2005a).

6.1.2 ML Toolkits and Algorithms

As mentioned above, I opt for approaching the classification task of NSUs using machine learning techniques. It is perhaps worth stressing however that my focus of attention is not on the methods and algorithms themselves, but on *using* them for the task at hand and, as importantly, to obtain interesting linguistic insights into the data. Nevertheless, a few words on the systems I employ are in order. Here I shall just provide a short description of the systems. For more detailed information on the algorithms I refer the reader to the references cited.

Weka The Weka toolkit, developed at the University of Waikato in New Zealand by Witten and Frank (2000), includes Java implementations of several machine learning algorithms. Many of them are designed for numeric prediction only (like the model tree inducer M5 for instance) and consequently are not appropriate for our classification task. From the toolkit I mainly use the J4.8 decision tree learner, which is an implementation of a slightly revised version of the popular C4.5 algorithm (Quinlan 1993) called Revision 8. One of the main qualities of Weka is its friendly graphical interface. When using J4.8 this allows for the visualisation of the output decision tree, which can be a valuable source of interesting information.

I also use two other simple classifiers included in the Weka toolkit to derive baseline systems—a majority class predictor and a one-rule classifier.

SLIPPER SLIPPER (Simple Learner with Iterative Pruning to Produce Error Reduction) is a rule induction system created by Cohen and Singer (1999). Like Weka's J4.8, it is based on the C4.5 decision tree algorithm, which it improves by using iterative pruning and confidence-rated boosting. The output of SLIPPER is a weighted rule set, in which each rule is associated with a confidence level. I use SLIPPER's option `unordered`, which finds a rule set that separates each class from the remaining classes using growing and pruning techniques. This yields slightly better results than the default setting. To classify an instance x, one computes the sum of the confidences that cover x: if the sum is greater than zero, the positive class is predicted. For each class, the only rule with a negative confidence rating is a single default rule, which predicts membership in the remaining classes.

One advantage of SLIPPER is that it generates transparent, relatively compact rule sets that can provide interesting insights into the data. Here however I will in general concentrate on Weka's decision trees to illustrate the points and will refer to SLIPPER's rules only seldom.

TiMBL TiMBL is a memory-based learning algorithm developed at Tilburg University by Daelemans et al. (2003). As with all memory-based machine learners, TiMBL stores representations of instances from the training set explicitly in memory. In the prediction phase, the similarity between a new test instance and all examples in memory is computed using a distance metric. The system selects the most frequent category within the set of most similar examples (the k-nearest neighbours). As a distance metric I use *modified value difference metric*, which performs better than the default setting (*overlap metric*). In light of recent studies (Daelemans and Hoste 2002), it is likely that the performance of TiMBL could be improved by a more systematic optimisation of its parameters, as e.g. in the experiments presented in (Gabsil and Lemon 2004). Here I only optimise the distance metric parameter and keep the default settings for the number of nearest neighbours ($k = 1$) and feature weighting method (*gain ratio*).

MaxEnt Finally, I experiment with a maximum entropy algorithm developed by Zhang Le (2003). It computes the model with the highest entropy of all models that satisfy the constraints provided by the features. The maximum entropy toolkit I use allows for several options. In the present experiments I use 40 iterations of the default L-BFGS parameter estimation (Malouf 2002).

6.1.3 Features and Feature Annotation

In order to construct the data sets, the NSU instances used in the experiments were annotated with a set of features—one feature set for the NSU classification experiment and a different feature set for the sluicing experiment. As will become clear when these sets of features are described in detail in Sections 6.2.2 and 6.3.2, I opt for using small sets of *meaningful* features, instead of taking large amounts of arbitrary features as is common in some stochastic approaches. I do this with the aim of obtaining a better understanding of the different classes of NSUs, their distribution and their properties.

In all the experiments the feature values are extracted automatically using the PoS information encoded in the BNC. The annotation procedure involves a simple algorithm which employs string searching and pattern matching techniques that exploit the SGML mark-up of the corpus. The BNC was automatically tagged using the CLAWS system developed at Lancaster University (Garside 1987). The $\sim$100 million words in the corpus were annotated according with a set of 57 PoS codes (known as the C5 tag-set) plus 4 codes for punctuation tags. A list of these codes can be found in (Burnard 2000). The BNC PoS annotation process is described in detail in (Leech et al. 1994).

Unfortunately the BNC mark-up does not include any coding of intonation. The features can therefore not use any intonational data, which would presumably be a useful source of information to distinguish, for instance, between question- and proposition denoting NSUs, between Plain Acknowledgement and Plain Affirmative Answer, and between Reprise and Direct sluices.

6.2 Experiment I: NSU Classification

In this section I present the set of experiments that investigate the task of classifying NSUs.

6.2.1 Data

The data used in the first experiment was selected from the corpus of NSUs following some simplifying restrictions. Firstly, I leave aside the 16 instances classified as 'Other' in the corpus study (see Table 2.3 in Chapter 2). Secondly, I restrict the experiments to those NSUs whose antecedent is the immediately preceding utterance. This restriction, which makes the feature annotation task easier, does not pose a significant coverage problem, given that the immediately preceding utterance is the antecedent for the vast majority of NSUs (88%—see Table 2.5 in Chapter 2). The set of all NSUs, excluding those classified as 'Other', whose antecedent is the immediately preceding utterance contains a total of 1123 datapoints. Table 6.1 shows the frequency distribution for the NSU classes partitioned as explained in Section 6.1.1.

Finally, the last restriction adopted concerns the instances classified as Plain Acknowledgement and Check Question. Taking the risk of ending up with a considerably smaller data set, I decided to leave aside these meta-communicative NSU classes given that (i) plain acknowledgements make up more than 50% of the sub-corpus leading to a data set with very skewed distributions; (ii) check questions are realised by the same kind of expressions as plain acknowledgements ("okay","right", etc) and would presumably be captured by the same feature; and (iii) a priori these two classes seem two of the easiest types to identify (a hypothesis that was confirmed after a second experiment—see Section 6.2.6 below). I therefore exclude plain acknowledgements and check questions and concentrate on a more interesting and less skewed data set containing all remaining NSU classes. This makes up a total of 527 data points (1123 – 582 – 15). In Section 6.2.6 I shall compare the results obtained using this restricted data set with those of a second experiment in which plain acknowledgements and check questions are incorporated.

6.2.2 Features

A small set of features, extractable from PoS information, that capture the contextual properties that are relevant for NSU classification was identified. In particular three types of properties that play an important role in the classification task were singled

NSU class	**Total**
Plain Acknowledgement	582
Short Answer	105
Affirmative Answer	100
Repeated Acknowledgement	80
Clarification Ellipsis	66
Rejection	48
Repeated Affirmative Answer	25
Factual Modifier	23
Sluice	20
Helpful Rejection	18
Filler	16
Check Question	15
Bare Modifier Phrase	10
Propositional Modifier	10
Conjunct	5
Total dataset	**1123**

Table 6.1: NSU sub-corpus

out. The first one has to do with semantic, syntactic and lexical properties of the NSUs themselves. The second one refers to the properties of its antecedent utterance. The third concerns relations between the antecedent and the fragment. Table 6.2 shows an overview of the nine features used.

- **NSU features** A set of four features are related to properties of the NSUs. These are `nsu_cont`,`wh_nsu`,`aff_neg` and `lex`. The feature `nsu_cont` is intended to distinguish between question-denoting (`q` value) and proposition-denoting (`p` value) NSUs. The feature `wh_nsu` encodes the presence of a *wh*-phrase in the NSU— it is primarily introduced to identify Sluices. The features `aff_neg` and `lex` signal the appearance of particular lexical items. They include a value `e(mpty)` which allows to encode the absence of the relevant lexical items as well. The values of the feature `aff_neg` indicate the presence of either a *yes* or a *no* word in the NSU. The values of `lex` are invoked by the appearance of modal adverbs (`p_mod`), factual adjectives (`f_mod`), and prepositions (`mod`) and conjunctions (`conj`) in initial positions. These features are expected to be crucial to the identification of Plain/Repeated Affirmative Answer and Plain/Helpful Rejection on the one hand, and Propositional Modifiers, Factual Modifiers, Bare Modifier Phrases and Conjuncts on the other.

feature	**description**	**values**
`nsu_cont`	content of the NSU (either prop or question)	`p,q`
`wh_nsu`	presence of a *wh* word in the NSU	`yes,no`
`aff_neg`	presence of a “yes”/“no” word in the NSU	`yes,no,e(mpty)`
`lex`	presence of different lexical items in the NSU	`p_mod,f_mod,mod,conj,e(mpty)`
`ant_mood`	mood of the antecedent utterance	`decl,n_decl`
`wh_ant`	presence of a *wh* word in the antecedent	`yes,no`
`finished`	(un)finished antecedent	`fin,unf`
`repeat`	repeated words in NSU and antecedent	`0-3`
`parallel`	repeated tag sequences in NSU and antecedent	`0-3`

Table 6.2: NSU features and values

Note that the feature `lex` could be split into four binary features, one for each of its non-empty values. This option, however, leads to virtually the same results. Hence, I opt for a more compact set of features. This also applies to the feature `aff_neg`.

- **Antecedent features** I use the features `ant_mood`,`wh_ant`, and `finished` to encode properties of the antecedent utterance. The first one of these features distinguishes between declarative and non-declarative antecedents. The feature `wh_ant` signals the presence of a *wh*-phrase in the antecedent utterance, which seems to be the best cue for classifying Short Answers. As for the feature `finished`, it should help the learners identify Fillers. The value `unf` is invoked when the antecedent utterance has a hesitant ending (indicated, for instance, by a pause) or when there is no punctuation mark signalling a finished utterance.

- **Similarity features** The last two features, `repeat` and `parallel`, encode similarity relations between the NSU and its antecedent utterance. They are the only numerical features in the feature set. The feature `repeat`, which indicates the appearance of repeated words between NSU and antecedent, is introduced as a clue to identify Repeated Affirmative Answers and Repeated Acknowledgements. The feature `parallel`, on the other hand, is intended to capture the particular parallelism exhibited by Helpful Rejections. It signals the presence of sequences of PoS tags common to the NSU and its antecedent.

Some of the features, like `nsu_cont` and `ant_mood`, for instance, are *high level* features that do not have straightforward correlates in PoS tags. Punctuation tags (that would correspond to intonation patterns in spoken input) help to extract the values of these features, but the correspondence is still not unique. For this reason the automatic feature annotation procedure was evaluated against a small sample of manually annotated data.

The feature values were extracted manually for 52 instances ($\sim$10% of the total) randomly selected from the data set. In comparison with this gold standard, the automatic feature annotation procedure achieves 89% accuracy. Only automatically annotated data is used for the learning experiments.

6.2.3 Baselines

I now turn to examine some baseline systems that will help us to evaluate the classification task. The simplest baseline one can consider is a majority class baseline that always predicts the class with the highest probability in the data. In the restricted data set used for the first experiment this is the class Short Answer. The majority class baseline yields a 6.7% weighted f-score (see Table 6.4 below).

A slightly more interesting baseline can be obtained by using a one-rule classifier. I use the implementation of a one-rule classifier provided in the Weka toolkit. For each feature, the classifier creates a single rule which generates a decision tree where the root is the feature in question and the branches correspond to its different values. The leaves are then associated with the class that occurs most often in the data, for which that value holds. The classifier then chooses the feature which produces the minimum error.

In this case, the feature with the minimum error is `aff_neg`. It produces the one-rule decision tree in Figure 6.1, which yields a 32.5% weighted f-score (see Table 6.3). Plain Affirmative Answer is the class predicted when the NSU contains a *yes*-word; Rejection when it contains a *no*-word; and Short Answer otherwise.

```
aff_neg:
- yes --> AffAns
- no  --> Reject
- e   --> ShortAns
```

Figure 6.1: One-rule tree

Finally, I consider a more substantial baseline that uses the four NSU features. Running Weka's J4.8 decision tree classifier with these features creates a decision tree with four rules, one for each feature used. The tree is shown in Figure 6.2.

```
nsu_cont:
- q --> wh_nsu:
          - yes   --> Sluice
          - no    --> CE
- p --> lex:
          - conj  --> ConjFrag
          - p_mod --> PropMod
          - f_mod --> FactMod
          - mod   --> BareModPh
          - e     --> aff_neg:
                        - yes --> AffAns
                        - no  --> Reject
                        - e   --> ShortAns
```

Figure 6.2: Four-rule tree

The root of the tree corresponds to the feature `nsu_cont`. It makes a first distinction between question-denoting (q branch) and proposition-denoting NSUs (p branch). Not surprisingly, within the q branch the feature wh_nsu is used to distinguish between Sluice and CE. The feature lex is the first node in the p branch. Its different values capture the classes Conjunct, Propositional Modifier, Factual Modifier and Bare Modifier Phrase. The e(mpty) value for this feature takes us to the last, most embedded node of the tree, realised by the feature aff_neg, which creates a sub-tree parallel to the one-rule tree in Figure 6.1. This four-rule baseline yields a 62.33% weighted f-score. Detailed results for the three baselines considered are shown in Tables 6.3, 6.4 and 6.5, respectively.

All results reported (here and in the sequel) were obtained by performing 10-fold cross-validation. They are presented as follows: The tables show the recall, precision and f-measure for each class. To calculate the overall performance of the algorithm, these scores are normalised according to the relative frequency of each class. This is done by multiplying each score by the total of instances of the corresponding class and then dividing by the total number of datapoints in the data set. The weighted overall recall, precision and f-measure, shown in the bottom row of the tables, is then the sum of the corresponding weighted scores. The NSU classes not shown in the tables obtain null scores.

NSU class	**recall**	**prec**	**f1**
ShortAns	100.00	20.10	33.50
Weighted Score	**19.92**	**4.00**	**6.67**

Table 6.3: Majority class baseline

NSU class	**recall**	**prec**	**f1**
ShortAns	95.30	30.10	45.80
AffAns	93.00	75.60	83.40
Reject	100.00	69.60	82.10
Weighted Score	**45.93**	**26.73**	**32.50**

Table 6.4: One-rule baseline

6.2.4 Feature Contribution

As can be seen in Table 6.5, the classes Sluice, CE, Propositional Modifier and Factual Modifier achieve very high f-scores with the four-rule baseline—between 97% and 100%. These results are not improved upon by incorporating additional features nor by using

NSU class	recall	prec	f1
CE	96.97	96.97	96.97
Sluice	100.00	95.24	97.56
ShortAns	94.34	47.39	63.09
AffAns	93.00	81.58	86.92
Reject	100.00	75.00	85.71
PropMod	100.00	100.00	100.00
FactMod	100.00	100.00	100.00
BareModPh	80.00	72.73	76.19
ConjFrag	100.00	71.43	83.33
Weighted Score	**70.40**	**55.92**	**62.33**

Table 6.5: Four-rule baseline

more sophisticated learners, which indicates that NSU features are sufficient indicators to classify these NSU classes. This is in fact not surprising, given that the disambiguation of these categories is tied to the presence of particular lexical items that are relatively easy to identify (*wh*-phrases and certain adverbs and adjectives). Recall however that a further distinction needs to be made between the different interpretations conveyed by sluices, a task I address in the experiment reported in Section 6.3.

There are however four NSU classes that are not predicted at all when only NSU features are used. These are Repeated Affirmative Answer, Helpful Rejection, Repeated Acknowledgement and Filler. Because they are not associated with any leaf in the tree, they yield null scores and therefore don't appear in Table 6.5. Examination of the confusion matrices shows that around 50% of Repeated Affirmative Answers were classified as Plain Affirmative Answers, while the remaining 50%, as well as the overwhelming majority of the other three classes just mentioned, were classified as Short Answer. Acting as the default class, Short Answers achieves the lowest score—63.09% f-score.

In order to determine the contribution of the antecedent features (`ant_mood`, `wh_ant`, `finished`), as a next step these were added to the NSU features used in the four-rule tree. When the antecedent features are incorporated, two additional NSU classes are predicted. These are Repeated Acknowledgement and Filler, which achieve rather positive results—74.8% and 64% f-score, respectively. The full results obtained when NSU and antecedent features are used together are not shown. Besides the addition of these two NSU classes, the results are very similar to those achieved with just NSU features. The tree obtained when the antecedent features are incorporated to the NSU features can be derived by substituting the last node in the tree in Figure 6.2 for the tree in Figure 6.3. As can be seen in Figure 6.3, the features `ant_mood` and `finished` contribute to distinguish Repeated Acknowledgement and Filler from Short Answer, whose f-score consequently raises—from 63.09% to 79%—due to an improvement on precision. Interestingly, the feature `wh_ant` does not have any contribution at this stage (although it will be used by the learners when the similarity features are added.) The general weighted f-score obtained when NSU and antecedent features are combined is 77.87%. A comparison of all weighted f-scores obtained will be shown in the next section, in Table 6.6.

```
aff_neg:
- yes  --> AffAns
- no   --> Reject
- e    --> ant_mood:
            - n_decl  --> ShorAns
            - decl    --> finished:
                           - fin   --> RepAck
                           - unfin --> Filler
```

Figure 6.3: Node on a tree using NSU and antecedent features

The use of NSU features and antecedent features is clearly not enough to account for Repeated Affirmative Answer and Helpful Rejection, which obtain null scores.

6.2.5 ML Results

In this next section I report the results obtained when the similarity features are included, thereby using the full feature set, and the four machine learning algorithms are trained on the data.

Although the classification algorithms implement different machine learning techniques, they all yield very similar results: around an 87% weighted f-score. The maximum entropy model performs best, although the difference between its results and those of the other algorithms is not statistically significant. Detailed recall, precision and f-measure scores are shown in Tables 6.7, 6.8, 6.9 and 6.10 for Weka's J4.8, SLIPPER, TiMBL and MaxEnt, respectively.

As seen in previous sections, the four-rule baseline algorithm that uses only NSU features yields a 62.33% weighted f-score, while the incorporation of antecedent features yields 77.83% weighted f-score. The best result, the 87.75% weighted f-score obtained with the maximal entropy model using all features, shows a 10% improvement over this last result. As promised, a comparison of the scores obtained with the different baselines considered and all learners used is given in Table 6.6.

System	w. f-score
Majority class baseline	6.67
One rule baseline	32.50
Four rule baseline (NSU features)	62.33
NSU and antecedent features	77.83
Full feature set	
SLIPPER	86.35
TiMBL	86.66
J4.8	87.29
MaxEnt	87.75

Table 6.6: Comparison of weighted f-scores

Short Answers achieve high recall scores with the baseline systems (more than 90%). In the three baselines considered, Short Answer acts as the default category. Therefore, even though the recall is high (given that Short Answer is the class with the highest probability), precision tends to be quite low. The precision achieved for Short Answer when only NSU features are used is ∼47%. When antecedent features are incorporated precision goes up to ∼72%. Finally, the addition of similarity features raises the precision for this class to ∼82%. Thus, by using features that help to identify other categories with the machine learners, the precision for Short Answers is improved by around 36%, and the precision of the overall classification system by almost 33%—from 55.90% weighted precision obtained with the four-rule baseline, to the 88.41% achieved with the maximum entropy model using all features.

Indeed, with the addition of the similarity features (`repeat` and `parallel`), the classes Repeated Affirmative Answer and Helpful Rejection are predicted by the learners. Although this contributes to the improvement of precision for Short Answer, the scores yielded by these two categories are lower than the ones achieved with other classes. Repeated Affirmative Answer achieves nevertheless decent f-scores, ranging from 56.96% with SLIPPER to 67.20% with MaxEnt. The feature `wh_ant`, for instance, is used to distinguish Short Answer from Repeated Affirmative Answer. Figure 6.4 shows one of the sub-trees generated by the feature `repeat` when Weka's J4.8 is used with the full feature set.

The class with the lowest scores is clearly Helpful Rejection. TiMBL achieves a 39.92% f-score for this class. The maximal entropy model, however, yields only a 10.37% f-score. Examination of the confusion matrices shows that ∼27% of Helpful Rejections were classified as Plain Rejection, ∼26% as Short Answer, and ∼15% as Repeated Acknowledgement. This indicates that the feature `parallel`, introduced to identify this type of NSUs, is not a good enough cue.

```
repeat:
 - = 0  --> finished:
                - unf -->    Filler
                - fin -->    parallel:
                              - = 0 --> ShortAns
                              - > 0 --> HelpReject
 - > 0  --> ant_mood:
                - decl   --> RepAck
                - n_decl --> repeat:
                             - = 1 -->  wh_ant:
                                         - yes --> ShortAns
                                         - no  --> RepAffAns
                             - > 1 -->  RepAffAns
```

Figure 6.4: Node on a tree using the full feature set

NSU class	**recall**	**prec**	**f1**
CE	97.00	97.00	97.00
Sluice	100.00	95.20	97.60
ShortAns	89.60	82.60	86.00
AffAns	92.00	95.80	93.90
Reject	95.80	80.70	87.60
RepAffAns	68.00	63.00	65.40
RepAck	85.00	89.50	87.20
HelpReject	22.20	33.30	26.70
PropMod	100.00	100.00	100.00
FactMod	100.00	100.00	100.00
BareModPh	80.00	100.00	88.90
ConjFrag	100.00	71.40	83.30
Filler	56.30	100.00	72.00
Weighted Score	**87.62**	**87.68**	**87.29**

Table 6.7: Weka's J4.8

NSU class	**recall**	**prec**	**f1**
CE	93.64	97.22	95.40
Sluice	96.67	91.67	94.10
ShortAns	83.93	82.91	83.41
AffAns	93.13	91.63	92.38
Reject	83.60	100.00	91.06
RepAffAns	53.33	61.11	56.96
RepAck	85.71	89.63	87.62
HelpReject	28.12	20.83	23.94
PropMod	100.00	90.00	94.74
FactMod	100.00	100.00	100.00
BareModPh	100.00	80.56	89.23
ConjFrag	100.00	100.00	100.00
Filler	100.00	62.50	76.92
Weighted Score	**86.21**	**86.49**	**86.35**

Table 6.8: SLIPPER

NSU class	**recall**	**prec**	**f1**
CE	94.37	91.99	93.16
Sluice	94.17	91.67	92.90
ShortAns	88.21	83.00	85.52
AffAns	92.54	94.72	93.62
Reject	95.24	81.99	88.12
RepAffAns	63.89	60.19	61.98
RepAck	86.85	91.09	88.92
HelpReject	35.71	45.24	39.92
PropMod	90.00	100.00	94.74
FactMod	97.22	100.00	98.59
BareModPh	80.56	100.00	89.23
ConjFrag	100.00	100.00	100.00
Filler	48.61	91.67	63.53
Weighted Score	**86.71**	**87.25**	**86.66**

Table 6.9: TiMBL

NSU class	**recall**	**prec**	**f1**
CE	96.11	96.39	96.25
Sluice	100.00	95.83	97.87
ShortAns	89.35	83.59	86.37
AffAns	92.79	97.00	94.85
Reject	100.00	81.13	89.58
RepAffAns	68.52	65.93	67.20
RepAck	84.52	81.99	83.24
HelpReject	5.56	77.78	10.37
PropMod	100.00	100.00	100.00
FactMod	97.50	100.00	98.73
BareModPh	69.44	100.00	81.97
ConjFrag	100.00	100.00	100.00
Filler	62.50	90.62	73.98
Weighted Score	**87.11**	**88.41**	**87.75**

Table 6.10: MaxEnt

6.2.6 Incorporating Plain Acknowledgement and Check Question

As explained in Section 6.2.1, the data set used in the experiments reported in the previous sections excluded the instances classified as Plain Acknowledgement and Check Question in the corpus study. The fact that Plain Acknowledgement is the category with the highest probability in the sub-corpus (making up more than 50% of our total data set—see Table 6.1), and that it does not seem particularly difficult to identify could affect the performance of the learners by inflating the results. Therefore it was left out in order to work with a more balanced data set and to minimise the potential for misleading results. As the expressions used in plain acknowledgements and check questions are very similar and they would in principle be captured by the same feature values, check questions were left out as well. In a second phase the instances classified as Plain Acknowledgement and Check Question were incorporated to measure their effect on the results. In this section I discuss the results obtained and compare them with the ones achieved in the initial experiment.

To generate the annotated data set an additional value `ack` was added to the feature `aff_neg`. This value is invoked to encode the presence of expressions typically used in plain acknowledgements and/or check questions ("mhm", "right","okay" etc.). The total data set (1123 data points) was automatically annotated with the features modified in this way, and the machine learners were then run on the annotated data.

6.2.6.1 Baselines

Given the high probability of Plain Acknowledgement, a simple majority class baseline gives relatively high results—35.31% weighted f-score. The feature with the minimum error used to derived the one-rule baseline is again `aff_neg`, this time with the new value `ack` as part of its possible values (see Figure 6.5 below). The one-rule baseline yields a weighted f-score of 54.26%.

aff_neg:
```
- ack --> Ack
- yes --> Ack
- no  --> Reject
- e   --> ShortAns
```

Figure 6.5: One-rule tree

The four-rule tree that uses only NSU features goes up to a weighted f-score of 67.99%. In this tree the feature `aff_neg` is now also used to distinguish between CE and Check Question. Figure 6.6 shows the `q` branch of the tree.

```
nsu_cont:
 - q -->  wh_nsu:
            - yes   --> Sluice
            - no    --> aff_neg:
                          - ack   --> CheckQu
                          - yes   --> CheckQu
                          - no    --> CE
                          - e     --> CE
```

Figure 6.6: Node on the four-rule tree

As the last node of the four-rule tree now corresponds to the tree in Figure 6.5, the class Plain Affirmative Answer is not predicted when only NSU features are used.

When antecedent features are incorporated, Plain Affirmative Answers, Repeated Acknowledgements and Fillers are predicted, obtaining very similar scores to the ones achieved in the experiment with the restricted data set. The feature `ant_mood` is now also used to distinguish between Plain Acknowledgement and Plain Affirmative Answer. The last node in the tree is shown in Figure 6.7. The combined use of NSU features and antecedent features yields a weighted f-score of 85.44%.

```
aff_neg:
- ack --> Ack
- yes --> ant_mood:
            - n_decl  --> AffAns
            - decl    --> Ack
- no  --> Reject
- e   --> ant_mood:
            - n_decl  --> ShorAns
            - decl    --> finished:
                            - fin   --> RepAck
                            - unfin --> Filler
```

Figure 6.7: Node on a tree using NSU and antecedent features

6.2.6.2 ML Results

As in the first experiment, when all features are used the results obtained are very similar across learners (around 92% weighted f-score), if slightly lower with Weka's J4.8

(89.53%). Detailed scores for each class are shown in Tables 6.12, 6.13, 6.14 and 6.15. As expected, the class Plain Acknowledgement obtains a high f-score (∼95% with all learners). The f-score for Check Question ranges from 73% yielded by MaxEnt to 90% obtained with SLIPPER. The high score of Plain Acknowledgement combined with its high probability raises the overall performance of the systems almost four points over the results obtained in the previous experiment—from ∼87% to ∼92% weighted f-score. The improvement with respect to the baselines, however, is not as large: we now obtain a 55% improvement over the simple majority class baseline (from 35.31% to 92.21%), while in the experiment with the restricted data set the improvement with respect to the majority class baseline is of 81% (from 6.67% to 87.75% weighted f-score.).

Table 6.11 shows a comparison of all weighted f-scores obtained in this second experiment.

System	**w. f-score**
Majority class baseline	35.31
One rule baseline	53.03
Four rule baseline (NSU features)	67.99
NSU and antecedent features	85.44
Full feature set	
J4.8	89.53
SLIPPER	92.01
TiMBL	92.02
MaxEnt	92.21

Table 6.11: Comparison of weighted f-scores

It is interesting to note that even though the overall performance of the algorithms is slightly higher than before (due to the reasons mentioned above), the scores for some NSU classes are actually lower. The most striking cases are perhaps the classes Helpful Rejection and Conjunct, for which the maximum entropy model now gives null scores (see Table 6.15). We have already pointed out the problems encountered with Helpful Rejection. As for the class Conjunct, although it yields good results with the other learners, the proportion of this class—0.4%, 5 instances only—is now probably too low to obtain reliable results.

A more interesting case is the class Affirmative Answer, which in TiMBL goes down more than 10 points (from 93.61% to 82.42% f-score—see Tables 6.9 and 6.14). The tree in Figure 6.5 provides a clue to the reason for this. When the NSU contains a *yes*-word (second branch of the tree) the class with the highest probability is now Plain Acknowledgement, instead of Plain Affirmative Answer as before (see tree in Figure 6.1). This is due to the fact that, at least in English, expressions like e.g. “yeah” (considered here as *yes*-words) are potentially ambiguous between acknowledgements and affirmative answers.[1] This ambiguity and the problems it entails are also noted by Schlangen (2005), who addresses the problem of identifying NSUs automatically. As he points out, the ambiguity of *yes*-words is one of the difficulties encountered when trying to distinguish between backchannels (plain acknowledgements in our taxonomy) and non-backchannel fragments. This is a tricky problem for Schlangen as his fragment identification procedure does not have access to the context. Although in the present experiments I do use features that capture contextual information, determining whether the antecedent utterance is declarative or interrogative (which one would expect to be the best clue to disambiguate between Plain Acknowledgement and Plain Affirmative Answer) is not always trivial.

[1]Arguably this ambiguity would not arise in French given that, according to Beyssade and Marandin (2005), in French the expressions used to acknowledge an assertion are different from those used in affirmative answers to polar questions.

NSU class	recall	prec	f1
Ack	95.00	96.80	95.90
CheckQu	100.00	83.30	90.90
CE	92.40	95.30	93.80
Sluice	100.00	95.20	97.60
ShortAns	83.00	80.70	81.90
AffAns	86.00	82.70	84.30
Reject	100.00	76.20	86.50
RepAffAns	68.00	65.40	66.70
RepAck	86.30	84.10	85.20
HelpReject	33.30	46.20	38.70
PropMod	60.00	100.00	75.00
FactMod	91.30	100.00	95.50
BareModPh	70.00	100.00	82.40
ConjFrag	100.00	71.40	83.30
Filler	37.50	50.00	42.90
Weighted Score	**89.67**	**89.78**	**89.53**

Table 6.12: Weka's J4.8

NSU class	recall	prec	f1
Ack	96.67	95.71	96.19
CheckQu	86.67	100.00	92.86
CE	96.33	93.75	95.02
Sluice	94.44	100.00	97.14
ShortAns	85.25	84.46	84.85
AffAns	82.79	87.38	85.03
Reject	77.60	100.00	87.39
RepAffAns	67.71	72.71	70.12
RepAck	84.04	92.19	87.93
HelpReject	29.63	18.52	22.79
PropMod	100.00	100.00	100.00
FactMod	100.00	100.00	100.00
BareModPh	83.33	69.44	75.76
ConjFrag	100.00	100.00	100.00
Filler	70.00	56.33	62.43
Weighted Score	**91.57**	**92.70**	**92.01**

Table 6.13: SLIPPER

NSU class	recall	prec	f1
Ack	95.71	95.58	95.64
CheckQu	77.78	71.85	74.70
CE	93.32	94.08	93.70
Sluice	100.00	94.44	97.14
ShortAns	87.79	88.83	88.31
AffAns	85.00	85.12	85.06
Reject	98.33	80.28	88.39
RepAffAns	58.70	55.93	57.28
RepAck	86.11	80.34	83.12
HelpReject	22.67	40.00	28.94
PropMod	100.00	100.00	100.00
FactMod	97.50	100.00	98.73
BareModPh	69.44	83.33	75.76
ConjFrag	100.00	100.00	100.00
Filler	44.33	55.00	49.09
Weighted Score	**91.49**	**90.75**	**91.02**

Table 6.14: TiMBL

NSU class	recall	prec	f1
Ack	95.54	94.59	95.06
CheckQu	63.89	85.19	73.02
CE	92.18	95.01	93.57
Sluice	88.89	94.44	91.58
ShortAns	88.46	84.91	86.65
AffAns	86.83	81.94	84.31
Reject	100.00	78.21	87.77
RepAffAns	69.26	62.28	65.58
RepAck	86.95	77.90	82.18
HelpReject	00.00	00.00	00.00
PropMod	44.44	100.00	61.54
FactMod	93.33	100.00	96.55
BareModP	58.33	100.00	73.68
ConjFrag	00.00	00.00	00.00
Filler	62.59	100.00	76.99
Weighted Score	**91.96**	**93.17**	**91.21**

Table 6.15: MaxEnt

6.3 Experiment II: Sluicing

The experiments on the classification of NSUs presented in previous sections have shown that sluices can successfully be identified—the class Sluice achieves around 97% f-score in all experiments reported. I have already pointed out that, given the form-based definition of this class, high scores were in fact expected.

Now for the classification output to have any impact on the resolution of question-denoting NSUs, more fine-grained distinctions need to be made. In the second corpus study presented in Chapter 2, Section 2.4, I identified four possible readings that can potentially be expressed by bare *wh*-phrases: Direct, Reprise, Clarification and Wh-anaphor. In this section I address the task of distinguishing between these different interpretations, following the same methodology used for the general NSU classification task.

6.3.1 Data

The data set used in the sluicing experiment was selected from the classified corpus of sluices presented in Chapter 2. The corpus study involved an annotation process whereby two samples of sluices were annotated by three coders. To generate the input data for the present experiments, all three-way agreement instances plus those instances where there is agreement between coder 1 and coder 2 (the two coders with the highest agreement) were selected, leaving out cases classified as `unclear`. As the reading Wh-anaphor had only been introduced in the classification of the second sample, 9 instances in the first sample were reclassified using this category and were also included in the data set.[2] The total data set includes 351 datapoints. The distribution of the different sluice interpretations is shown in Table 6.16.

Sluice Interp.	**Total**
Direct	106
Reprise	203
Clarification	24
Wh-anaphor	18
Total data set	**351**

Table 6.16: Sluice sub-corpus

As can be seen in Table 6.16, the classes in the data set have significantly skewed

[2]The reclassified instances in the first sample were those that had motivated the introduction of the Wh-anaphor category for the second sample. Given that there were no disagreements involving this category, this reclassification was straightforward.

distributions. However, as we are now faced with a much smaller data set, we cannot afford to leave out any subset of the data. Hence, the 351 data points are used in the ML experiments with its original distributions.

6.3.2 Features

To annotate the data a set of 11 features was used. An overview of the features and their values is shown in Table 6.17.

Besides the feature `sluice`, which indicates the sluice type, all the other features are concerned with properties of the antecedent utterance. The features `mood` and `polarity` refer to syntactic and semantic properties of the antecedent utterance as a whole. The remaining features, on the other hand, focus on a particular lexical type or construction contained in the antecedent. These features (`quant`, `deictic`, `proper_n`, `pro`, `def_desc`, `wh` and `overt`) are not annotated independently, but conditionally on the sluice type. That is, they will take `yes` as a value if the element or construction in question appears in the antecedent *and* it matches the semantic restrictions imposed by the sluice type. For instance, when a sluice with value `where` for the feature `sluice` is annotated, the feature `deictic`, which encodes the presence of a deictic pronoun, will take value `yes` only if the antecedent utterance contains a *locative* deictic like "here" or "there". Similarly the feature `wh` takes a `yes` value only if there is a *wh*-word in the antecedent that is identical to the sluice type.

Unknown or irrelevant values are indicated by a question mark `?` value. This allows us to express, for instance, that the presence of a proper name is irrelevant to determining the interpretation of say a *when* sluice, while it is crucial when the sluice type is *who*. The feature `overt` takes `no` as value when there is no overt antecedent expression. It takes `yes` when there is an antecedent expression not captured by any other feature, and it is considered irrelevant (`?` value) when there is an antecedent expression defined by another feature.

The 351 data points were automatically annotated with the 11 features shown in Table 6.17. Again, the automatic annotation procedure was evaluated against a manual gold standard, achieving an accuracy of 86%.

6.3.3 Baselines

Because sluices conveying a Reprise reading make up more than 57% of our data set, relatively high results can already be obtained with a majority class baseline that always predicts the class Reprise. This yields 42.4% weighted f-score, as shown in Table 6.18.

The second baseline considered is a one-rule baseline. In this case the feature with the minimum error chosen by the one-rule classifier is `sluice`. The classifier produces

feature	**description**	**values**
`sluice`	type of sluice	`what, why, who,...`
`mood`	mood of the antecedent utterance	`decl, n_decl`
`polarity`	polarity of the antecedent utterance	`pos, neg, ?`
`quant`	presence of a quantified expression	`yes, no, ?`
`deictic`	presence of a deictic pronoun	`yes, no, ?`
`proper_n`	presence of a proper name	`yes, no, ?`
`pro`	presence of a pronoun	`yes, no, ?`
`def_desc`	presence of a definite description	`yes, no, ?`
`wh`	presence of a *wh* word	`yes, no, ?`
`overt`	presence of any other potential ant. expression	`yes, no, ?`

Table 6.17: Sluice features and values

Sluice Interp.	**recall**	**prec**	**f1**
Reprise	100	57.80	73.30
Weighted Score	**57.81**	**33.42**	**42.40**

Table 6.18: Majority class baseline

the one-rule tree in Figure 6.8. The branches of the tree correspond the the sluice types; the interpretation with the highest probability for each type of sluice is then predicted.

By using the feature `sluice` the one-rule tree implements the correlations between sluice type and preferred interpretation that were discussed in Chapter 2, Section 2.4.4. There, I pointed out that these correlations were statistically significant ($\chi^2 = 438.53$, $p \leq 0.001$). We can see now that they are indeed a good rough guide for predicting sluice readings. As shown in Table 6.19, the one-rule baseline dependent on the distribution patterns of the different sluice types yields a 72.73% weighted f-score.

Sluice Interp.	**recall**	**prec**	**f1**
Direct	72.60	67.50	70.00
Reprise	79.30	80.50	79.90
Clarification	100	64.90	78.70
Weighted Score	**73,61**	**71.36**	**72.73**

Table 6.19: One-rule baseline

```
sluice:
 - who    --> Reprise
 - what   --> Clarification
 - why    --> Direct
 - where  --> Reprise
 - when   --> Direct
 - which  --> Reprise
 - whichN --> Reprise
```

Figure 6.8: One-rule tree

6.3.4 ML Results

Finally, the four machine learning algorithms were run on the data set annotated with the eleven features. We obtain results of around 80% weighted f-score, although in this case there are some significant differences amongst the learners. MaxEnt gives the lowest score—73.24% weighted f-score—hardly over the one-rule baseline, and more than 8 points lower than the best results, obtained with Weka's J4.8—81.80% weighted f-score. The size of the data set seems to play a role in these differences, indicating that MaxEnt does not perform so well with small data sets. A summary of weighted f-scores is given in Table 6.20.

System	**w. f-score**
Majority class baseline	42.40
One rule baseline	72.73
MaxEnt	73.24
TiMBL	79.80
SLIPPER	81.62
J4.8	81.80

Table 6.20: Comparison of weighted f-scores

Detailed recall, precision and f-measure results for each learner are shown in Tables 6.21, 6.22, 6.23 and 6.24. The results yield by MaxEnt are almost equivalent to the ones achieved with the one-rule baseline. With the other three learners, the use of contextual features improves the results for Reprise and Direct by around 5 points each with respect to the one-rule baseline. It can be seen however that the results obtained with the one-rule baseline for the Clarification reading are hardly improved upon by any of the learners. In the case of TiMBL the score is in fact lower—72.16 v. 78.70 weighted f-score. This leads us to conclude that the best strategy is to interpret all *what* sluices as conveying a Clarification reading.

The class Wh-anaphora, which not being the majority interpretation for any sluice type was not predicted by the one-rule baseline nor by MaxEnt, now gives positive results with the other three learners. The best result for this class is obtained with Weka's J4.8—80% f-score.

The decision tree generated by Weka's J4.8 algorithm is displayed in Figure 6.9. The root of the tree corresponds to the feature `wh`, which makes a first distinction between Wh-anaphor and the other readings. If the value of this feature is `yes`, the class Wh-anaphor is predicted. A negative value for this feature leads to the feature `sluice`. The class with the highest probability is the only clue used to predict the interpretation of the sluice types `what,where,which` and `whichN` in a way parallel to the one-rule baseline. Additional features are used for `when,why` and `who`. A Direct reading is predicted for a `when` sluice if there is no overt antecedent expression, while a Reprise reading is preferred if the feature `over` takes as value `yes`. For `why` sluices the mood of the antecedent utterance is used to disambiguate between Reprise and Direct: if the antecedent is declarative, the sluice is classified as Direct; if it non-declarative it is interpreted as Reprise. In the classification of `who` sluices three features are taken into account—`quant,pro` and `proper_n`. The basic strategy is as follows: if the antecedent utterance contains a quantifier and no personal pronouns nor proper names appear, the predicted class is Direct, otherwise the sluice is interpreted as Reprise.

6.3.4.1 Feature Contribution

Note that not all features are used in the tree generated by Weka's J4.8. The missing features are `polarity, deictic` and `def_desc`. Although they don't make any contribution to the model generated by the decision tree, examination of the rules generated by SLIPPER shows that they are all used in the rule set induced by this algorithm.

For instance, the feature `polarity` is used in a rule that assigns Direct interpretations to `why` sluices, albeit with a low confidence level (less than 1).

```
Direct not Reprise|Clarification|Wh_anaphor :-
                      sluice = why, polarity = pos, wh = no (+0.91662)
```

The feature `deictic` is used in combination with the sluice type `where` to predict a Reprise reading. In this case the rule is associated with a higher confidence level then the previous rule, even though it does not seem to be very informative as Reprise is the default interpretation for *where* sluices anyway.

Sluice Interp.	**recall**	**prec**	**f1**
direct	71.70	79.20	75.20
reprise	85.70	83.70	84.70
clarification	100.00	68.60	81.40
wh_anaphor	66.70	100.00	80.00
Weighted Score	**81.47**	**82.14**	**81.80**

Table 6.21: Weka's J4.8

Sluice Interp.	**recall**	**prec**	**f1**
direct	81.01	71.99	76.23
reprise	83.85	86.49	85.15
clarification	71.17	94.17	81.07
wh_anaphor	77.78	62.96	69.59
Weighted Score	**81.81**	**81.43**	**81.62**

Table 6.22: SLIPPER

Sluice Interp.	**recall**	**prec**	**f1**
direct	78.72	75.24	76.94
reprise	83.08	83.96	83.52
clarification	75.83	68.83	72.16
wh_anaphor	55.56	77.78	64.81
Weighted Score	**79.85**	**79.98**	**79.80**

Table 6.23: TiMBL

Sluice Interp.	**recall**	**prec**	**f1**
direct	65.22	75.56	70.01
reprise	85.74	76.38	80.79
clarification	89.17	70.33	78.64
wh_anaphor	0.00	0.00	0.00
Weighted Score	**75.38**	**76.93**	**73.24**

Table 6.24: MaxEnt

```
wh:
- yes --> Wh_anaphor
- no  --> sluice:
           - what   --> Clarification
           - where  --> Reprise
           - which  --> Reprise
           - whichN --> Reprise
           - when   --> overt:
                         - yes    --> Reprise
                         - no     --> Direct
           - why    --> ant_mood
                         - decl   --> Direct
                         - n_decl --> Reprise
           - who    --> quant:
                         - yes --> pro:
                                    - yes --> Reprise
                                    - no  --> proper_n:
                                               - yes --> Reprise
                                               - no  --> Direct
                         - no  --> Reprise
```

Figure 6.9: Weka's J4.8 tree

```
Reprise not Direct|Clarification|Wh_anaphor :-
                   sluice = where, deictic = yes (+1.77533)
```

A perhaps more informative rule could be the following, where the absence of deictic pronouns favours Direct interpretations over Reprise ones. The confidence level of the rule however is significantly low.

```
Direct not Reprise|Clarification|Wh_anaphor :-
                                  deictic = yes (+0.139897)
```

The feature `def_desc` was mainly introduced to captured potential antecedents for *which* and *whichN* sluices. However, given the overwhelming majority of the Reprise reading for this kind of sluices, the feature does not seem to play any role in this respect. It is nevertheless used by SLIPPER in combination with other features:

```
Reprise not Direct|Clarification|Wh_anaphor :-
       quant = no, def_desc = yes, mood = n_decl (+1.24173)
```

Despite the fact that all features are used in the rules generated by SLIPPER, the contribution of the features `polarity`, `deictic` and `def_desc` does not seem to be very significant. When they are eliminated from the feature set, SLIPPER yields very similar results to the ones obtained with the full set of features—81.22% weighted f-score v. the 81.66% obtained before. TiMBL on the other hand goes down a couple of points—from 79.80% to 77.32% weighted f-score. No variation is of course observed with MaxEnt, which seems to be using just the sluice type as a clue for classification.

6.4 Automatic NSU Resolution: SHARDS

As mentioned at the outset of this chapter, one of the aims of employing statistical techniques to disambiguate amongst NSUs was to develop a classification model whose output could be used by a dialogue system. This would provide the system with the right NSU class, which in the present approach determines the appropriate resolution procedure. Obviously, for this to be of any use, the system fed by the output of the classifier needs to be equipped with the capability of resolving NSUs on the basis of this information. A system designed with the aim of resolving NSUs is SHARDS (Ginzburg et al. 2001, Fernández et al. in press), which implements an approach in line with the one I have presented in this thesis.

SHARDS is an implemented system which provides a procedure for computing the interpretation of some NSUs in dialogue. The system comprises two main components: an HPSG-based grammar and a resolution procedure. The SHARDS grammar is an implemented version of the grammar proposed by Ginzburg and Sag (2001), and is encoded in ProFIT (Erbach 1995). It is is able to parse NSUs and to assign them an underspecified semantic representation.

The last sentence parsed by the grammar is stored in memory and is used by the second component of the system to resolve the underspecified content of NSUs. The resolution component creates a model of dialogue context that provides a set of possible questions under discussion and a set of salient utterances. Once an NSU has been parsed, the resolution component instantiates the values of the features MAX-QUD and SAL-UTT and combines them with the underspecified semantic representation of the NSU to fully resolve its content.

The baseline system handles short answers, direct and reprise sluices, as well as plain affirmative answers to polar questions. SHARDS however has been extended to cover several types of clarification requests, and has been used as a part of the information-state-based dialogue system CLARIE (Purver 2004b), which implements the theory of grounding and clarification proposed by Ginzburg and Cooper (2004).

6.5 Summary and Conclusions

This chapter has addressed the task of automatically classifying NSUs employing machine learning techniques. I have presented two experiments: one concerned with the classification of all NSU classes, and another one focussed on the disambiguation of sluice interpretations. In both cases, the results obtained are decidedly positive and provide the basis for a wide-coverage NSU classification model that can be used to assist the resolution of NSUs in a dialogue processing system.

The experiments have also provided interesting insights into the data, for instance by showing that some NSUs that according to the approach of Chapter 4 require more complex resolution mechanisms—like Helpful Rejection and Repeated Acknowledgement—are also NSU classes that are more difficult to disambiguate automatically; while [+ SA] NSUs like Plain Acknowledgement, Plain Affirmative Answer, Plain Rejection and Propositional Modifier, since they usually have a strong lexical component are rather easy to spot. As for sluices, the experiments have confirmed that the identified correlations between interpretation and sluice type are a good guide to predict sluice reading.

To train the machine learners I have used a small set of features that capture either properties of the NSUs themselves, of their antecedents, or of the relation between antecedent and NSU. The positive results obtained suggest that the features employed offer a reasonable basis for machine learning acquisition of the NSU taxonomy. However, some features like `parallel`, introduced to account for Helpful Rejection, require considerable improvement. A possibility in this direction could be to use similar techniques to the ones employed e.g. in (Poesio et al. 2004, Schlangen 2005) to compute semantic similarity to derive a notion of semantic contrast that would complement this structural feature. This is however an issue that I leave to future investigation.

7 Conclusions

In this chapter, I briefly summarise the main contributions made in this thesis and provide an outlook on possible ways in which the work could be further developed.

7.1 Summary of Contributions

This thesis has been concerned with the analysis of non-sentential utterances in dialogue. I have employed experimental, symbolic and statistical methods to provide an account that is grounded on empirical data, is theoretically motivated, and is amenable to computational processing. As the last section of each chapter offers an overview of its content, I shall not recapitulate this here. Instead, I devote this section to identifying some of the main contributions of this work. These are the following:

- The first contribution of the thesis concerns the development of a comprehensive, corpus-based taxonomy of NSU classes as they occur in conversation. The empirical study on which the taxonomy is based also offers novel data about the distance between NSUs and their antecedents.
- As a second contribution, I have shown that Type Theory with Records can be used to formalise the main classes of NSUs identified. In previous work, this kind of task has typically been addressed using the formalism of HPSG. Here I have demonstrated that Type Theory with Records offers a valid alternative to the grammatical characterisation of utterances in general, which has the advantage of combining *sign*-like structures with dynamic representations.
- A third contribution of the thesis relates to the proposed hierarchy of abstract models of dialogue protocols. The hierarchy can be seen as providing an abstract characterisation of the minimal computational requirements needed to deal with different structural features of dialogue interaction. This offers a novel link between dialogue dynamics and formal language theory.

- The fourth main contribution of this thesis concerns the application of machine learning techniques to the problem of automatically identifying NSU classes. This has resulted in a wide-coverage classification model that can be used to boost the resolution of NSUs in dialogue processing.

7.2 Further Work

In Chapter 4, I have formalised the main classes of NSUs. However, some classes like Bare Modifier Phrase, Factual Modifier, Conjunct or Filler were not given an analysis. Perhaps the most natural direction for future work would be to extend the TTR analysis to cover these classes as well. For classes like Bare Modifier Phrase, this would involve extending the grammatical analysis to incorporate a proper treatment of modification. The class Filler, however, seems to require incremental processing and therefore would call for more radical extensions.

Regarding the abstract models for dialogue protocols introduced in Chapter 5, there are a variety of possible avenues of future research. The most obvious would be to try to identify further dialogue phenomena that require protocol models not covered by the analysis given so far. For instance, to analyse some NSU classes I have made use of accommodation. The question is whether this calls for an extension of the data type used (in order to be able to introduce elements into the memory component in retrospective), or whether this simply requires a more sophisticated use of an existing model. Another interesting issue is the interaction between protocols and other aspects of dialogue modelling. For instance, the choice of a particular protocol model may restrict the possible dialogue strategies. Vice versa, the need for a particular strategy may influence the data type used to model the protocol.

The classification task addressed in Chapter 6 is only a first step towards the automatic resolution of NSUs. To actually resolve NSUs, obviously the classifier has to be combined with a module that takes care of this task on the basis of the NSU class identified. In this respect, a possible route to take could be to integrate the classification model presented in this thesis with the information state-based dialogue system CLARIE (Purver 2004b), which is based on similar theoretical assumptions. One of the main aspects involved in the integration of a classifier with the resolution module would be the extraction of the feature values from the information state of the system, as well as the instantiation of the contextual variables with information from the feature values.

The input fed to the classification model I have presented is a vector of features associated with an utterance that has already been singled out as an NSU. Hence, a related point that needs further elaboration is the automatic identification of NSUs as opposed to other kinds of utterances in dialogue. As mentioned earlier, some work in this direction has recently been carried out by Schlangen (2005).

Finally, it is worth stressing that in this thesis I have only looked at NSUs from a processing perspective. Nothing has been said about the generation of NSUs in dialogue, which clearly is an interesting and important topic as well. Stina Ericsson's thesis addresses NSUs from this perspective (Ericsson 2005).

Appendix A
Generalising *PABAK* to n Categories

This appendix provides a generalisation of the $PABAK$ formula for n categories and m coders developed in collaboration with Ulle Endriss.

In (Byrt et al. 1993), $PABAK$ is computed with the same formula to compute $kappa$, but using as $P(E)$ an "average" chance agreement $P(E)_{av}$, where any category is taken to be equally chosen by all coders. In the case of two coders choosing between two categories, $PABAK$ equals $2P(A) - 1$, given that: (i) The probability p of a coder choosing one of the two categories is the total proportion divided by the number of categories, i.e. $p = 1/2 = 0.5$, (ii) the probability of the 2 coders agreeing on a category is $p^2 = 0.5^2 = 0.25$, and (iii) the probability of the 2 coders agreeing on any of the 2 categories equals $2 \cdot p^2$. Hence $P(E)_{av} = 2 \cdot 0.25 = 0.5$, and therefore

$$PABAK = \frac{P(A) - 0.5}{1 - 0.5} = 2P(A) - 1$$

The generalisation of $P(E)_{av}$ for m coders and n categories equals to $1/n$, given that: (i) The probability p of a coder choosing one of n categories is $1/n$, (ii) the probability of any pair of coders agreeing on a category is $(1/n)^2$, and (iii) the probability of any pair of coders agreeing on any category equals $n \cdot (1/n)^2$. Hence $P(E)_{av} = n \cdot (1/n)^2 = 1/n$. Therefore we obtain

$$PABAK = \frac{P(A) - \frac{1}{n}}{1 - \frac{1}{n}}$$

Appendix B
The TTR System: Formal Definitions

This appendix provides the formal definitions of Type Theory with Records. I introduce the general hierarchy of types and give precise definitions of records, record types, and families of record types. The appendix ends with a summary of the basic boolean operations on propositions.

B.1 The TTR System

In order to be able to treat types as objects in our domain, following Cooper (2006a) I introduce a stratified set **Type** of sets **Type**^{n} of types of order n. The stratification is achieved by stating that **Type**$^{n+1}$ is the set of types whose inhabitants are types of order n. **Type**^{0} is then the set of types whose inhabitants are not types. This allows us to treat types as first class citizens while avoiding paradoxes like $T : T$. Here I define the system of TTR in terms of stratified types. In the main text, however, I ignore the subscripts whenever there is no danger of confusion.

The system of Type Theory with Records is a set

$$\textbf{TTR System} = \{\textbf{Type}, \textbf{BasType}, \textbf{ProofType}, \textbf{M}\}$$

where

- $\textbf{Type} = \bigcup_{n \in \mathbb{N}} \textbf{Type}^n$ is a family of sets indexed by the naturals numbers such that for any $n \in \mathbb{N}$:
  - $\textbf{Type}^n$ is a set of types of order n.
  - $\textbf{RecType}^n$ is a set of record types ρ of order n, where record types are sets of pairs $\langle l, T\rangle$ defined with respect to a set of labels L, such that $l \in L$ and $T \in \textbf{Type}^n$.
  - If $T \in \textbf{RecType}^n$ then $T \in \textbf{Type}^n$.

- **BasType** is a set of basic types, including at least types $\mathit{Ind}, \mathit{Time}, \mathit{Bool}$.
- **ProofType** is a set of proof types. It is defined with respect to a pair $\langle \mathit{Pred}, V \rangle$, where *Pred* is a set of predicates and V a function that assigns to each $P \in \mathit{Pred}$ a finite tuple of types $T \in \mathbf{Type}$.
- $\mathbf{M} = \{D, F, v\}$ is a model where
  - D is a non-empty set of elements.
  - F is a function such that:

    for each $T \in \mathbf{BasType}$, $F(T) \subseteq D$

    for each $T \in \mathbf{ProofType}$, $F(T) \subseteq D$
  - v is a mapping from a set of symbols $\Sigma = \{a_1, \cdots, a_n\}$ to elements in D.

  F determines the inhabitants of the basic types by assigning to each $T \in \mathbf{BasType}$ a set of elements in D. If $T \in \mathbf{BasType}$, then $a : T$ iff $v(a) \in F(T)$.

  The elements of $\mathbf{ProofType}$ are determined by F in the following way: If $P \in \mathit{Pred}$, $V(P) = \langle T_1, \ldots, T_n \rangle$ and $v(a_1), \cdots, v(a_n) \in D$ are such that $a_1 : T_1, \ldots, a_n : T_n$, then $P(a_1, \cdots, a_n) \in \mathbf{ProofType}$. If $T \in \mathbf{ProofType}$, then $a : T$ iff $v(a) \in F(T)$.

B.1.1 The Hierarchy of Types

The hierarchy of types is constructed as follows:

- $\mathbf{Type}^0 = \mathbf{BasType} \cup \mathbf{ProofType}$, and nothing else.

For each $n > 0$, $\mathbf{Type}^n$ is inductively defined as follows:

- If $T \in \mathbf{Type}^n$ then $T \in \mathbf{Type}^{n+1}$.
- If $T_1, T_2 \in \mathbf{Type}^n$, then $(T_1 \rightarrow T_2) \in \mathbf{Type}^n$.
- If $T_1, T_2 \in \mathbf{Type}^n$, then $(T_1 \wedge T_2) \in \mathbf{Type}^n$.
- If $T_1, T_2 \in \mathbf{Type}^n$, then $(T_1 \vee T_2) \in \mathbf{Type}^n$.
- If $T \in \mathbf{Type}^n$, then $\neg T \in \mathbf{Type}^n$.
- No other types belong to $\mathbf{Type}^n$.
- If $T \in \mathbf{Type}^n$ and $T : T'$, then $T' \in \mathbf{Type}^{n+1}$.

- Let $\mathit{ProofType} \in \mathbf{Type}^1$ be the type of proof types; that is $T : \mathit{ProofType}$ iff $T \in \mathbf{ProofType}$.

- Let $\mathit{Type}^n \in \mathbf{Type}^{n+1}$ be the type of types of order n; that is $T : \mathit{Type}^n$ iff $T \in \mathbf{Type}^n$.

- Let $\mathit{RecType}^n \in \mathbf{Type}^{n+1}$ be the type of record types of order n; that is $T : \mathit{RecType}^n$ iff $T \in \mathbf{RecType}^n$.

B.1.2 Records, Record Types and Families of Record Types

- The inhabitants of record types are records, that is sets of pairs $\langle l, a\rangle$ of labels l and objects a. A record $r = \{\langle l_1, a_1\rangle, \cdots, \langle l_n, a_n\rangle\}$ is of type $\rho = \{\langle l_1, T_1\rangle, \cdots, \langle l_k, T_k\rangle\}$ iff for each field $\langle l_i, T_i\rangle \in \rho$ there is a field $\langle l_i, a_i\rangle \in r$ such that $a_i : T_i$.

- Let Rec^n be the type of records of type ρ such that $\rho : \mathit{RecType}^n$. The empty record $r = \{\ \}$ also denoted by [ ] is of type Rec^0. For any $n > 0$, if $T \in \mathbf{RecType}^n$ and $r : T$, then $r : \mathit{Rec}^n$.

- The empty record type $\rho = \{\ \}$ is also denoted by $[\ ] \in \mathbf{RecType}^0$. For each $n > 0$, $\mathbf{RecType}^n$ is inductively defined as follows: If $\rho \in \mathbf{RecType}^n$, l is a label not occurring in ρ, and $T \in \mathbf{Type}^n$, then $\rho \cup \{\langle l, T\rangle\} \in \mathbf{RecType}^n$. A record r is of type $\rho \cup \{\langle l, T\rangle\}$ iff $r : \rho$, $\langle l, a\rangle$ is a field in r and $a : T$.

- *Dependent record types* are a particular kind of record types. Dependent record types are record types ρ such that some types appearing in ρ are *families of types*. A family of types over a type T is a function $\mathcal{F}$ of type $(T \rightarrow \mathit{Type}^n)$ from variables $x : T$ to types $T' : \mathit{Type}^n$ dependent on x, such that $\mathcal{F}(a) = T'_{[x \rightarrow a]}$ for any $a : T$. We extend the subtyping relation to families of types as follows:

  Definition 7 (Subtyping of families of types) *Let $\mathcal{F}_1 : (T_1 \rightarrow \mathit{Type}^n)$ and $\mathcal{F}_2 : (T_2 \rightarrow \mathit{Type}^n)$ be families of types. Then $\mathcal{F}_1 \sqsubseteq \mathcal{F}_2$ iff $T_1 \sqsubseteq T_2$ and $\mathcal{F}_1(a) \sqsubseteq \mathcal{F}_2(a)$ for any element $a : T_1$.*

  If $\rho \in \mathbf{RecType}^n$, $\mathcal{F}$ is a family of types over ρ, and l is a label not occurring in ρ, then $\rho \cup \{\langle l, \mathcal{F}\rangle\} \in \mathbf{RecType}^n$ is a dependent record type.

- *Families of record types* are a particular kind of families of types. A family of record types is a family of types $\mathcal{F}$ over T such that $\mathcal{F} : (T \rightarrow \mathit{RecType}^n)$.

B.2 Boolean Operations on Propositions

- **Negation**

  Given a proposition $p = \begin{bmatrix} \mathrm{s} & = & r \\ \mathrm{s}_T & = & \rho \end{bmatrix}$ its negation is $q = \begin{bmatrix} \mathrm{s} & = & r \\ \mathrm{s}_T & = & \neg\rho \end{bmatrix}$. If p is true then q is false.

- **Conjuction**

  Given propositions $p = \begin{bmatrix} \mathrm{s} & = & r_1 \\ \mathrm{s}_\mathrm{T} & = & \rho_1 \end{bmatrix}$ and $q = \begin{bmatrix} \mathrm{s} & = & r_2 \\ \mathrm{s}_\mathrm{T} & = & \rho_2 \end{bmatrix}$ their conjunction $\{p, q\}$ is $\begin{bmatrix} \mathrm{s} & = & \langle r_1, r_2 \rangle \\ \mathrm{s}_\mathrm{T} & = & \rho_1 \wedge \rho_2 \end{bmatrix}$. $\{p, q\}$ is true iff p and q are true.

- **Disjunction**

  Given propositions $p = \begin{bmatrix} \mathrm{s} & = & r \\ \mathrm{s}_\mathrm{T} & = & \rho_1 \end{bmatrix}$ and $q = \begin{bmatrix} \mathrm{s} & = & r \\ \mathrm{s}_\mathrm{T} & = & \rho_2 \end{bmatrix}$ their disjunction $\{p, q\}$ is $\begin{bmatrix} \mathrm{s} & = & r \\ \mathrm{s}_\mathrm{T} & = & \rho_1 \vee \rho_2 \end{bmatrix}$. $\{p, q\}$ is true iff either p or q is true.

Appendix C
A Grammar Fragment

This appendix contains a detailed grammar fragment for the utterance representations put forward in Chapter 4. I start by introducing a small lexicon, to then give the compositional principles by means of which basic phrases and simple clausal types are constructed.

C.1 Preliminaries

The grammar will be defined as a set of constructions, in the sense of Sag (1997). Lexical items, phrases and clauses are represented as families of information state types. Recall that these are functions $\lambda(r : IS_1).IS_2$ where $IS_1, IS_2 \sqsubseteq IS$. Let us first zoom into the internal structure of the *IS* type in order to better grasp some notational abbreviations I shall use in the definition of the grammar. Information states are records of the following type:

(202) $$IS =_{def} \begin{bmatrix} \text{facts} & : & \begin{bmatrix} \text{s} & : & \mathit{Rec} \\ \text{s}_T & : & \mathit{RecType} \end{bmatrix} \\ \text{qud} & : & \left\langle \begin{bmatrix} \text{que} : \mathit{Question} \\ \text{top} : \langle \mathit{Sign} \rangle \end{bmatrix} \right\rangle \\ \text{utt} & : & \begin{bmatrix} \text{s} & : & \mathit{Rec} \\ \text{s}_T & : & \begin{bmatrix} \text{z} & : & \begin{bmatrix} \text{phon} & : & \mathit{Phon} \\ \text{syn} & : & \mathit{Cat} \\ \text{sem} & : & \mathit{Type} \\ \text{quant} & : & \mathit{RecType} \end{bmatrix} \\ \text{a} & : & \mathit{Ind} \\ \text{c} & : & \mathit{Utter}(\text{a}, \text{z}) \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

The type *Sign* corresponds to the record type labelled z within utt. The label quant will be used to handle existential quantifiers, of which I will offer an extremely simple account that will be limited to indefinite NPs. The content of definite NPs will be the value of their label quants, which will be later on incorporated into the content of clauses containing these phrases.

As the representation of families of *IS* types can be rather cumbersome, in the definition of the grammatical types I shall use some notational shortcuts, most of which were already introduced in Chapter 4. They can be summarised as follows:

- I will generally ignore those fields, in both domain and range types of the function, that are not relevant for current purposes. When there is no need to zoom into either the domain or the range types, I will use the general type *IS*.
- Within the range type of the function, I will mostly be concerned with the sign associated with each construction—i.e. the value of label z within the situation type of the locutionary proposition in utt. Consequently, I will whenever possible use the label $\mathrm{utt.z}$ instead of utt to avoid clutter and directly refer to that sign.
- As for qud, usually I will be concerned with the first element in the qud list. Therefore I will commonly employ the label qud_1 instead of qud to refer to that first element.
- Within the domain type, often I will need to refer to contextual parameters that will typically be part of the situation type of the propositional content of facts. Thus, for convenience of notation, I will usually refer to them directly under facts, i.e. ignoring the intermediate label s_T.

C.2 Words

I start by defining a small lexicon containing proper names and common nouns, *wh*-words, the singular indefinite article, intransitive and transitive verbs, and the auxiliary verb *do*.

Proper Names The type of proper names is shown in (203), while (204) shows the lexical entry for the name *Mia*.

(203) $p\_name =_{def}$

$$\lambda\left(r : \left[\ \mathrm{facts} : \begin{bmatrix} \mathrm{x} : \mathit{Ind} \\ \mathrm{c} : \mathit{named}(\mathrm{x}, \mathit{Name}) \end{bmatrix}\ \right]\right) . \left[\ \mathrm{utt.z} : \begin{bmatrix} \mathrm{syn} = \mathrm{N} : \mathit{Cat} \\ \mathrm{sem} : [\mathrm{y} = r.\mathrm{x} : \mathit{Ind}] \end{bmatrix}\ \right]$$

(204) $$\lambda\left(r : \left[\ \mathrm{facts} : \begin{bmatrix} \mathrm{x} : \mathit{Ind} \\ \mathrm{c} : \mathit{named}(\mathrm{x}, \mathit{Mia}) \end{bmatrix}\ \right]\right) . \left[\ \mathrm{utt.z} : \begin{bmatrix} \mathrm{phon} : \mathtt{mia} \\ \mathrm{syn} = \mathrm{N} : \mathit{Cat} \\ \mathrm{sem} : [\mathrm{y} = r.\mathrm{x} : \mathit{Ind}] \end{bmatrix}\ \right]$$

Being referential expressions, proper names denote individuals, whose existence is taken for granted. This is indicated by requiring their content to be of type *Ind*, and constraining the current information state to include the appropriate individual—in (204), for instance, an individual x named *Mia*. Identity between the contextually provided individual and the denotation of the name is enforced by means of a manifest field.

Common Nouns The type of common nouns is given in (205). In a classical montogovian style, the content of common nouns is modelled as a function from individuals to proof types. An example for the noun *bike* is shown in (206). As the content of the noun is in principle not dependent on context, the domain type is unconstrained i.e. specified as being the general type of information states *IS*.

(205) $cn =_{def}$

$$\lambda\,(r : \mathit{IS})\,.\left[\ \text{utt.z} : \left[\begin{array}{l}\text{syn} = \text{N} : \mathit{Cat}\\ \text{sem} : \left[\ \text{c} : \mathit{Ind} \rightarrow \mathit{ProofType}\ \right]\end{array}\right]\ \right]$$

(206) $$\lambda\,(r : \mathit{IS})\,.\left[\ \text{utt.z} : \left[\begin{array}{l}\text{phon} : \texttt{bike}\\ \text{syn} = \text{N} : \mathit{Cat}\\ \text{sem} : \left[\ \text{c} : \lambda(\text{x} : \mathit{Ind}).\mathit{Bike}(\text{x})\ \right]\end{array}\right]\ \right]$$

Wh Words The type in (207) is the type of interrogative *wh*-words. Interrogative *wh*-words like e.g. *who* and *what* are associated with *wh*-restrictors, i.e. records which introduce an individual and restrict it appropriately, as seen in Section 4.2.2.1 of Chapter 4. (208) shows the lexical entry for *who*.

(207) $wh =_{def}$

$$\lambda\,(r : \mathit{IS})\,.\left[\begin{array}{lcl}\text{qud}_1 & : & \left[\ \text{top} = \langle\text{utt.z}\rangle : \langle\mathit{Sign}\rangle\ \right]\\ \text{utt.z} : & & \left[\begin{array}{lcl}\text{syn} = \text{N} : \mathit{Cat} & & \\ \text{sem} & : & T_{wh}\end{array}\right]\end{array}\right]$$

(208) $$\lambda\,(r : \mathit{IS})\,.\left[\begin{array}{lcl}\text{qud}_1 & : & \left[\ \text{top} = \langle\text{utt.z}\rangle : \langle\mathit{Sign}\rangle\ \right]\\ \text{utt.z} : & & \left[\begin{array}{lcl}\text{phon} & : & \texttt{who}\\ \text{syn} = \text{N} : \mathit{Cat} & & \\ \text{sem} & : & \left[\begin{array}{l}\text{x} : \mathit{Ind}\\ \text{c} : \mathit{person}(\text{x})\end{array}\right]\end{array}\right]\end{array}\right]$$

Indefinite Article I adopt a simple view of singular indefinite articles as functions that map a property p (the denotation of common nouns) to a record type introducing an individual that is constrained to bear p. This will be enough to ensure existential quantification of the individual in question.[1]

(209) $ind\_det =_{def}$

$$\lambda\,(r : IS)\,.\left[\ \text{utt.z} : \begin{bmatrix} \text{phon} & : & \texttt{a} \\ \text{syn} = \text{DET} & : & Cat \\ \text{sem} & : & \lambda(r_1 : [\text{p} : Ind \rightarrow ProofType]).\begin{bmatrix} \text{x} : Ind \\ \text{c} : r_1.\text{p@x} \end{bmatrix} \end{bmatrix}\ \right]$$

Verbs The types of intransitive and transitive verbs are shown in (210) and (211), respectively. (212) and (213) give examples of particular lexical entries. The content of an intransitive verb like *procrastinate* is a function that maps an individual to a proof type that is inhabited if that individual procrastinates. A transitive verb like *find* follows the same pattern, but with an extra argument for its object.

(210) $int\_v =_{def}$

$$\lambda\,(r : IS)\,.\left[\ \text{utt} : \begin{bmatrix} \text{syn} = \text{V} : Cat \\ \text{sem} \ :\ \lambda\,(r_1 : [\text{x} : Ind])\,.\,[\ \text{c} : ProofType] \end{bmatrix}\ \right]$$

(211) $tra\_v =_{def}$

$$\lambda\,(r : IS)\,.\left[\ \text{utt.z} : \begin{bmatrix} \text{syn} = \text{V} : Cat \\ \text{sem} \ :\ \lambda\,(r_1 : [\text{x} : Ind])\,. \\ \qquad (\lambda\,(r_2 : [\text{y} : Ind])\,.\,[\ \text{c} : ProofType]) \end{bmatrix}\ \right]$$

(212) $$\lambda\,(r : IS)\,.\left[\ \text{utt.z} : \begin{bmatrix} \text{phon} & : & \texttt{procrastinate} \\ \text{syn} = \text{V} & : & Cat \\ \text{sem} & : & \lambda\,(r_1 : [\text{x} : Ind])\,.\left[\ \text{c} : Procrastinate(r_1.\text{x})\ \right] \end{bmatrix}\ \right]$$

(213) $$\lambda\,(r : IS)\,.\left[\ \text{utt.z} : \begin{bmatrix} \text{phon} & : & \texttt{find} \\ \text{syn} = \text{V} & : & Cat \\ \text{sem} & : & \lambda\,(r_1 : [\text{x} : Ind])\,. \\ & & \Big(\lambda\,(r_2 : [\text{y} : Ind])\,.\left[\ \text{c} : Find(r_1.\text{x}, r_2.\text{y})\ \right]\Big) \end{bmatrix}\ \right]$$

[1]Recall that by the definition of record typehood, record types are truth-bearing objects, which leads to an effect of existential quantification (cf. Chapter 4, Section 4.1.1.2).

The auxiliary verb *do* simply denotes the identity function, which I represent with the type Id:

(214) $aux\_do =_{def}$

$$\lambda\,(r : IS)\,.\left[\begin{array}{l} \text{utt.z} : \left[\begin{array}{lcl} \text{phon} & : & \texttt{do} \\ \text{syn} = \text{V} & : & Cat \\ \text{sem} & : & Id \end{array}\right] \end{array}\right]$$

C.3 Phrases

Like words, phrases are also families of IS types, built up from those specified by their immediate constituents. Phrasal functions have the following general form:

(215) $\lambda(r : IS).\left[\begin{array}{lcl} \text{utt.z} & : & Phrase \end{array}\right]$

The type $Phrase$ is a subtype of $Sign$. As can be seen in (216), phrasal signs have two additional labels: a label slash that will be used to construct canonical *wh*-interrogatives as extraction constructions, and a label dtrs whose value is the list of signs associated with the immediate constituents of a phrase. I shall label the signs within the list of daughter with labels $\mathrm{d_1, d_2, \ldots, d_n}$.

(216) $$Phrase \quad =_{def} \left[\begin{array}{lcl} \text{phon} & : & Phon \\ \text{syn} & : & Cat \\ \text{sem} & : & Type \\ \text{quant} & : & RecType \\ \text{slash} & : & Type \\ \text{dtrs} & : & \langle Sign \rangle \end{array}\right]$$

Besides the idiosyncrasies introduced by each construction, to which I will turn to in a minute, some basic constrains governing the construction of phrases can be singled out. These apply to any phrasal IS family σ_0 with immediate constituents $\sigma_1, ..., \sigma_n$.

Let $\sigma = \lambda(r : T_1).T_2$ be an IS family. I shall use $Dom(\sigma)$ to refer to T_1 and $Rng(\sigma)$ to refer to T_2, where $T_1, T_2 \sqsubseteq IS$. If π is a path of labels $l_1, ..., l_n$ in T_1, then $Dom(\sigma).\pi$ denotes the type labelled by l_n in T_1. If π is a path of labels in T_2, then $Rng(\sigma).\pi$ denotes the type labelled by l_n in T_2.

- **Contextual amalgamation:** $Dom(u_0) \sqsubseteq Dom(\sigma_1), ..., Dom(\sigma_n)$

  The domain type of a phrasal IS family σ_0 is a subtype of the domain type of each of its immediate constituents. Essentially this means that phrases inherit the contextual requirements of their daughters, being on top of this able to introduce new requirements of their own.

- **quant amalgamation:** $\mathit{Rng}(\sigma_0).\mathrm{utt.z.quant} = \bigcup_{i=1}^{n} \mathit{Rng}(\sigma_i).\mathrm{utt.z.quant}$

  The type of the field labelled quant in the range type of a phrasal *IS* family σ_0 is the union of the record types labelled quant in the range type of its daughters.

- **Top amalgamation:** $\mathit{Rng}(\sigma_0).\mathrm{qud_1.top} = \langle \mathit{Sign} \rangle \bigoplus_{i=1}^{n} \mathit{Rng}(\sigma_i).\mathrm{qud_1.top}$

  The value of top within the maximal question under discussion of the range type of a phrasal *IS* family is the concatenation of some list of signs and the top values of the range type of its daughters. Again this means that phrases inherit topical sub-utterance from their daughters as well as being able to introduce topical sub-utterances themselves.

- **Phrasal structure:** $\mathit{Rng}(\sigma_0).\mathrm{utt.z.dtrs} = \langle \mathit{Sign} \rangle_{\langle \mathrm{d_1}:\mathit{Rng}(\sigma_1).\mathrm{utt.z},\ldots,\mathrm{d_n}:\mathit{Rng}(\sigma_n).\mathrm{utt.z}\rangle}$

  The value of the field labelled dtrs in the sign within the value of utt in the range type of a phrasal *IS* family σ_0 is a singleton type whose only inhabitant is a list whose members $\mathrm{d_1}, \ldots, \mathrm{d_n}$ are signs as specified in the range type of σ_0's daughters.

  If $\sigma_1 : T$ is an immediate constituent of σ_0, then within the list of daughters of σ_0 I will use the notation $T.\mathrm{z}$ instead of $\mathit{Rng}(\sigma_1).\mathrm{utt.z}$ to denote the type of the sign in the range type of σ_1. Again, this will become clear in a minute when phrases are defined.

C.3.1 Non-clausal Phrases

In this section I shall define simple NPs and VPs.

Noun Phrases The general type of NPs is shown in (217).

(217) $np =_{def}$

$$\lambda\,(r : \mathit{IS})\,.\left[\ \mathrm{utt.z} \quad : \quad \begin{bmatrix} \mathrm{syn} = \mathrm{NP} : \mathit{Cat} \\ \mathrm{sem} = [\mathrm{x} : \mathit{Ind}] \end{bmatrix}\ \right]$$

Beside the NPs consisting of a proper name or a *wh*-word, I consider indefinite NPs of the form *"a N"*. While proper name NPs are referential and require the presence in context of a suitable referent, indefinites are existentially quantified and introduce a topical sub-utterance in qud. The *IS* family in (218) is the type of indefinite NPs. Note that by **phrasal structure**, the value of dtrs is a list containing the signs associated with *IS* families of type *ind_det* and *cn*, which are therefore the immediate constituents of this phrasal type.

(218) np_ind :

$$\lambda\,(r : \mathit{IS})\,.\begin{bmatrix} \text{qud}_1 & : & \begin{bmatrix} \text{top} = \langle \text{utt.z}\rangle : \langle \mathit{Sign}\rangle \end{bmatrix} \\ \text{utt.z} & : & \begin{bmatrix} \text{syn} = \text{NP} : \mathit{Cat} \\ \text{sem} \;\;:\;\; \text{d}_1.\text{sem@d}_2.\text{sem} \\ \text{quant} = \text{d}_1.\text{sem@d}_2.\text{sem} : \mathit{RecType} \\ \text{dtrs} = \langle \text{d}_1 : \mathit{ind\_det}.\text{z},\ \text{d}_2 : \mathit{cn}.\text{z}\rangle : \langle \mathit{Sign}\rangle \end{bmatrix} \end{bmatrix}$$

There are a few aspects that should be noted about this type. First, the content of the indefinite NP arises by applying the content of the indefinite article (its first daughter) to the content of the noun (its second daughter). Second, the value of the label quant is the same as the content of the NP. And third, the indefinite NP becomes a topical sub-utterance within qud.

Let us see how this works with an example. Consider the indefinite NP *a bike*, whose immediate constituents are the indefinite article *a* and the noun *bike* as defined in (209) and (206), respectively. The types of the daughters are shown in detail in (220) and (221).

(219) $$\lambda\,(r : \mathit{IS})\,.\begin{bmatrix} \text{qud}_1 & : & \begin{bmatrix} \text{top} = \langle \text{utt.z}\rangle : \langle \mathit{Sign}\rangle \end{bmatrix} \\ \text{utt.z} & : & \begin{bmatrix} \text{phon} \;\;:\;\; \texttt{a bike} \\ \text{syn} = \text{NP} : \mathit{Cat} \\ \text{sem} \;\;:\;\; \text{d}_1.\text{sem@d}_2.\text{sem} \\ \text{quant} = \text{d}_1.\text{sem@d}_2.\text{sem} : \mathit{RecType} \\ \text{dtrs} = \langle \text{d}_1 : \mathit{a}.\text{z}, \text{d}_2 : \mathit{bike}.\text{z}\rangle : \langle \mathit{Sign}\rangle \end{bmatrix} \end{bmatrix}$$

(220) $$\text{d}_1 : \begin{bmatrix} \text{syn} = \text{DET} : \mathit{Cat} \\ \text{sem} : \lambda(r_1 : [\text{p} : \mathit{Type}]).\begin{bmatrix} \text{x} : \mathit{Ind} \\ \text{c} : r_1.\text{p@x} \end{bmatrix} \end{bmatrix}$$

(221) $$\text{d}_2 : \begin{bmatrix} \text{syn} = \text{N} : \mathit{Cat} \\ \text{sem} : \begin{bmatrix} \text{c} : \lambda(\text{y} : \mathit{Ind}).\mathit{Bike}(\text{y}) \end{bmatrix} \end{bmatrix}$$

The content of the NP (as well as the value of quant) arises by the application of $\text{d}_1.\text{sem}$ to $\text{d}_2.\text{sem}$, which has as output the following record type:

(222) $$\begin{bmatrix} \text{x} : \mathit{Ind} \\ \text{c} : \lambda(\text{y} : \mathit{Ind}).\mathit{Bike}(\text{y})\text{@x} \end{bmatrix}$$

Finally, after application of the function denoted by the noun to an individual x as specified in the field labelled c above, we obtain the record type in (223), which corresponds to the content of the indefinite NP.

(223) $\begin{bmatrix} \text{x} & : & \mathit{Ind} \\ \text{c} & : & \mathit{Bike}(\text{x}) \end{bmatrix}$

Verb Phrases (224) shows the general type of VPs, which is the same as that of intransitive VPs. The type of transitive VPs (subsumed by (224)) is given in (225).

(224) $\mathit{vp} =_{def}$

$$\lambda\,(r : \mathit{IS})\,.\begin{bmatrix} \text{utt.z} : \begin{bmatrix} \text{syn} = \text{VP} : \mathit{Cat} \\ \text{sem} \;\; : \;\; \lambda\,(r_1 : [\text{x} : \mathit{Ind}])\,.\,[\,\text{c} : \mathit{ProofType}] \end{bmatrix} \end{bmatrix}$$

(225) $\mathit{vp\_tra} =_{def}$

$$\lambda\,(r : \mathit{IS})\,.\begin{bmatrix} \text{utt.z} : \begin{bmatrix} \text{syn} = \text{VP} : \mathit{Cat} \\ \text{sem} : \text{d}_1.\text{sem@d}_2.\text{sem} \\ \text{dtrs} = \langle \text{d}_1 : \mathit{v\_tra}.\text{z},\; \text{d}_2 : \mathit{np}.\text{z}\rangle : \langle \mathit{Sign}\rangle \end{bmatrix} \end{bmatrix}$$

In the most standard way, the content of a transitive VP arises by applying the denotation of the verb to that of the object. Let us see how the VP *find a bike* would be built up from the *IS* families corresponding to the verb *find* (213) and the indefinite NP *a bike* (219).

(226)
$$\lambda\,(r : \mathit{IS})\,.\begin{bmatrix} \text{qud}_1 : \begin{bmatrix} \text{top} = \langle \text{d}_2\rangle : \langle \mathit{Sign}\rangle \end{bmatrix} \\ \text{utt.z} : \begin{bmatrix} \text{phon} \;\; : \;\; \texttt{find a bike} \\ \text{syn} = \text{VP} : \mathit{Cat} \\ \text{sem} : \lambda(r_2 : [\text{y} : \mathit{Ind}]).\begin{bmatrix} \text{c} : \mathit{Find}(r_2.\text{y}, \text{d}_2.\text{sem.x}) \end{bmatrix} \\ \text{quant} = \begin{bmatrix} \text{x} & : & \mathit{Ind} \\ \text{c} & : & \mathit{Bike}(\text{x}) \end{bmatrix} : \mathit{RecType} \\ \text{dtrs} = \langle \text{d}_1 : \mathit{find}.\text{z}, \text{d}_2 : \mathit{a\_bike}.\text{z}\rangle : \langle \mathit{Sign}\rangle \end{bmatrix} \end{bmatrix}$$

(227)
$$\text{d}_1 : \begin{bmatrix} \text{phon} \;\; : \;\; \texttt{find} \\ \text{syn} = \text{V} : \mathit{Cat} \\ \text{sem} : \;\; \lambda(r_1 : [\text{x} : \mathit{Ind}]). \\ \qquad\qquad \lambda(r_2 : [\text{y} : \mathit{Ind}]).\begin{bmatrix} \text{c} : \mathit{Find}(r_1.\text{x}, r_2.\text{y}) \end{bmatrix} \end{bmatrix}$$

(228)
$$\text{d2} : \begin{bmatrix} \text{phon} \;\; : \;\; \texttt{a bike} \\ \text{syn} = \text{NP} : \mathit{Cat} \\ \text{sem} \;\; : \;\; \begin{bmatrix} \text{x} & : & \mathit{Ind} \\ \text{c} & : & \mathit{Bike}(\text{x}) \end{bmatrix} \\ \text{quant} = \begin{bmatrix} \text{x} & : & \mathit{Ind} \\ \text{c} & : & \mathit{Bike}(\text{x}) \end{bmatrix} : \mathit{RecType} \end{bmatrix}$$

Applying $\mathrm{d}_1.\mathrm{sem}$ to $\mathrm{d}_2.\mathrm{sem}$ yields the following function, which is the content of the VP:

(229) $\lambda(r_2 : [y : \mathit{Ind}]).\left[\ \mathrm{c} : \mathit{Find}(r_2.\mathrm{y}, \mathrm{d}_2.\mathrm{sem.x})\ \right]$

By **top amalgamation** and **quant amalgamation**, the values of top and quant are the same as those of the object.

C.3.2 Clauses

In this section I will give the types for simple declarative clauses and basic polar and *wh*-interrogatives.

Declaratives The content of a simple declarative sentence consisting of an NP and a VP is a proposition whose situation type arises by applying the content of the VP to that of the NP, and uniting this with any existential quantifiers.

(230) $\mathit{declarative} =_{def}$

$$\lambda\left(r : \left[\ \mathrm{facts} : [\mathrm{s} : \mathit{Rec}]\ \right]\right). \left[\ \mathrm{utt.z} : \left[\begin{array}{l} \mathrm{syn} = \mathrm{S} : \mathit{Cat} \\ \mathrm{sem} = \left[\begin{array}{l} \mathrm{s}_1 = r.\mathrm{facts.s} : \mathit{Rec} \\ \mathrm{s}_T = \mathrm{d}_2.\mathrm{sem@d}_1.\mathrm{sem} \cup \mathrm{quant} : \mathit{RecType} \end{array}\right] : \mathit{Prop} \\ \mathrm{slash} : \mathit{Rec}_0 \\ \mathrm{quant} : \mathit{RecType} \\ \mathrm{dtrs} = \langle \mathrm{d}_1 : \mathit{np}.\mathrm{z}, \mathrm{d}_2 : \mathit{vp}.\mathrm{z} \rangle : \langle \mathit{Sign} \rangle \end{array}\right]\ \right]$$

Polar Interrogatives Polar interrogatives consist of the auxiliary verb *do*, an NP, and a VP. Their content is a polar question. The situation type of the propositional core is constructed by applying the content of the VP to that of the NP, and then applying the identity function denoted by the auxiliary. Finally, any quantifiers are also united with the situation type.

(231) $\mathit{polar\_int} =_{def}$

$$\lambda\left(r : \left[\ \mathrm{facts} : [\mathrm{s} : \mathit{Rec}]\ \right]\right). \left[\ \mathrm{utt.z} : \left[\begin{array}{l} \mathrm{syn} = \mathrm{S} : \mathit{Cat} \\ \mathrm{sem} = \lambda(r_1 : [\,]).\left[\begin{array}{l} \mathrm{s}_1 = r.\mathrm{facts.s} : \mathit{Rec} \\ \mathrm{s}_T = \mathrm{d}_1@(\mathrm{d}_3.\mathrm{sem@d}_2.\mathrm{sem}) \cup \mathrm{quant} : \mathit{RecType} \end{array}\right] : Q \\ \mathrm{slash} : \mathit{Rec}_0 \\ \mathrm{quant} : \mathit{RecType} \\ \mathrm{dtrs} = \langle \mathrm{d}_1 : \mathit{aux\_do}.\mathrm{z}, \mathrm{d}_2 : \mathit{np}.\mathrm{z}, \mathrm{d}_3 : \mathit{vp}.\mathrm{z} \rangle : \langle \mathit{Sign} \rangle \end{array}\right]\ \right]$$

Wh-interrogatives For simplicity's sake, I only provide a type for subject *wh*-interrogatives. To construct them, we need an additional declarative type that creates a proposition out of a VP. This is achieved by applying the function denoted by the VP to the value of slash.

(232) $non\_subj\_decl =_{def}$

$$\lambda\left(r : \left[\ \text{facts} : [\text{s} : \textit{Rec}]\ \right]\right) .
\left[\ \text{utt.z} : \left[\begin{array}{l} \text{syn} = \text{S} : \textit{Cat} \\ \text{sem} = \left[\begin{array}{l} \text{s}_1 = r.\text{facts.s} : \textit{Rec} \\ \text{s}_T = \text{d}_1.\text{sem@slash} \cup \text{quant} : \textit{RecType} \end{array}\right] : \textit{Prop} \\ \text{slash} : \textit{Type} \\ \text{dtrs} = \langle \text{d}_1 : \textit{vp}.\text{z} \rangle : \langle \textit{Sign} \rangle \end{array}\right]\ \right]$$

We can now construct subject *wh*-interrogatives treating the *wh*-phrase as a filler whose content is identified with the slash value of a non-subject declarative. The content of this construction is a question whose domain is the content of the filler *wh*-phrase and whose range is the propositional content of the non-subject declarative.

(233) $wh\_int =_{def}$

$$\lambda\left(r : \left[\ \text{facts} : [\text{s} : \textit{Rec}]\ \right]\right) .
\left[\ \text{utt.z} : \left[\begin{array}{l} \text{syn} = \text{S} : \textit{Cat} \\ \text{sem} = \lambda(\text{d}_1.\text{sem}).\text{d}_2.\text{sem} : \textit{Question} \\ \text{slash} : \textit{Rec}_0 \\ \text{quant} : \textit{Rec}_0 \\ \text{dtrs} = \left\langle \begin{array}{l} \text{d}_1 : [\text{sem} = \text{d}_2.\text{slash} : T_{wh}], \\ \text{d}_2 : [\text{sem} : [\text{s}_1 = r.\text{facts.s} : \textit{Rec}]] \end{array} \right\rangle : \langle \textit{Sign} \rangle \end{array}\right]\ \right]$$

C.4 Root Utterances

Finally, root utterances are defined as *IS* families of type *root*. The value of utt is of type *IllocProp*. The content of the sign is constrained to be of type *Message* and the value of slash to be the empty record.

(234) $root =_{def}$

$$\lambda\left(r : \begin{bmatrix} \text{facts} : \begin{bmatrix} \text{s} : \textit{Rec} \\ \text{a} : \textit{Ind} \\ \text{b} : \textit{Ind} \end{bmatrix} \\ \text{qud} : \left\langle \begin{bmatrix} \text{que} : \textit{Question} \\ \text{top} : \textit{Sign} \end{bmatrix} \right\rangle \\ \text{utt} : \textit{IllocProp} \end{bmatrix}\right).$$

$$\begin{bmatrix} \text{facts} \sqsubseteq r.\text{facts} : \textit{Prop} \\ \text{qud} : \left\langle \begin{bmatrix} \text{que} : \textit{Question} \\ \text{top} : \textit{Sign} \end{bmatrix} \right\rangle \\ \text{utt} : \begin{bmatrix} \text{s}_1 = r.\text{facts.s} : \textit{Rec} \\ \text{s}_T \;:\; \begin{bmatrix} \text{z} & : & \begin{bmatrix} \text{syn} = \text{S} : \textit{Cat} \\ \text{sem} : \textit{Message} \\ \text{slash} : \textit{Rec}_0 \end{bmatrix} \\ \text{c}_1 & : & \textit{Utter}(r.\text{facts.a}, \text{z}) \\ \text{c}_2 & : & \textit{IllocRel}(r.\text{facts.a}, r.\text{facts.b}, \text{z.sem}) \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Bibliography

René Ahn and Hans-Peter Kolb. Discourse representation meets constructive mathematics. In *Papers from the Second Symposium on Logic and Language*. Akadémia Kiadó, Budapest, 1990.

Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *Data Structures and Algorithms*. Adison-Wesley, 1983.

James Allen and Mark Core. *DAMSL: Dialogue Act Markup in Several Layers*, 1997.

Leila Amgoud, Nicolas Maudet, and Simon Parsons. Modelling Dialogues using Argumentation. In *Proceedings of 4th International Conference on MultiAgent Systems (ICMAS-2000)*. IEEE Press, 2000.

Nicholas Asher. *Reference to Abstract Objects in Discourse*. Studies in Linguistics and Philosophy. Kluwer Academc Publisher, Dordrecht, 1993.

Nicholas Asher. Varieties of Discourse Structure in Dialogue. In *Proceedings of the 2nd Workshop on the Semantics and Pragmatics of Dialogue (Twendial)*, 1998.

Nicholas Asher and Alex Lascarides. *Logics of Conversation*. Cambridge University Press, 2003.

Ellen Barton. *Nonsentential Constituents*. John Benjamins, Amsterdam, 1990.

Jon Barwise and John Etchemendy. Information, infons and inference. In Robin Cooper and John Perry Kuniaki Mukai, editors, *Situation Theory and its Applications*, number 22 in CSLI Lecture Notes, pages 33–78. CSLI Publications, Stanford, 1990.

Jon Barwise and John Perry. *Situations and Attitudes*. MIT Press, 1983.

Gustavo Betarte and Alvaro Tasistro. Extension of Martin Löf type theory with record types and subtyping. In G. Sambin and J. Smith, editors, *25 Years of Constructive Type Theory*. Oxford University Press, 1998.

Claire Beyssade and Jean-Marie Marandin. Contour Meaning and Dialogue Structure. Ms presented at the workshop Dialogue Modelling and Grammar, Paris, France, 2005.

Peter Bohlin, Johan Bos, Staffan Larsson, Ian Lewin, Colin Matheson, and David Milward. Survey of existing interactive systems. Technical report, The TRINDI project, Deliverable D1.3, 1999.

Johan Bos, Ewan Klein, Oliver Lemon, and Tetsushi Oka. DIPPER: Description and Formalisation of an Information-State Update Dialogue System Architecture. In *Proceedings of the 4th SIGdial Workshop on Discourse and Dialogue*, Sapporo, Japan, 2003.

Johan Bresnan. Locative Inversion and the Architecture of Universal Grammar. *Language*, 1(70):72–131, 1994.

Lou Burnard. *Reference Guide for the British National Corpus (World Edition)*. Oxford Universtity Computing Services, 2000. Available from `ftp://sable.ox.ac.uk/pub/ota/BNC/`.

Ted Byrt, Janet Bishop, and John B. Carlin. Bias, Prevalence and Kappa. *Journal of Clinical Epidemiology*, 46(5):423–429, 1993.

Sandra Carberry. *Plan Recognition in Natural Language Dialogue*. MIT Press, Cambridge, Massachusetts, 1990.

Jean Carletta. Assessing agreement on classification tasks: the kappa statistics. *Computational Linguistics*, 2(22):249–255, 1996.

Lauri Carlson. *Dialogue Games*. Synthese Language Library. D. Reidel, 1983.

Sandy Chung, William Ladusaw, and James McCloskey. Sluicing and logical form. *Natural Language Semantics*, 3:239–282, 1995.

Domenic V. Cicchetti and Alvan R. Feinstein. High agreement but low kappa: I. The problems of two paradoxes. *Journal of Clinical Epidemiology*, 43(6):543–548, 1990.

Herbert H. Clark. *Using Language*. Cambridge University Press, Cambridge, 1996.

Herbert H. Clark and Edward F. Schaefer. Contributing to discourse. *Cognitive Science*, 13:259–294, 1989.

Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20:37–46, 1960.

Philip R. Cohen and Hector J. Levesque. Rational interaction as the basis for communication. In P. Cohen, J. Morgana, and M. Pollack, editors, *Intentions in Communication*. MIT Press, 1990.

William Cohen and Yoram Singer. A simple, fast, and effective rule learner. In *Proceedings of the 16th National Conference on Artificial Intelligence (AAAI-99)*, 1999.

Marco Colombetti. A Commitment-based Approach to Agent Speech Acts and Conversations. In *Workshop on Agent Languages and Conversation Policies*, 2000.

Robin Cooper. Information states, attitudes and dependent record types. In L.Cavedon, P. Blackburn, N. Braisby, and A. Shimojima, editors, *Logic, Language and Computation*, volume 3, pages 85–106. CSLI Publications, 2000.

Robin Cooper. Records and Records Types in Semantic Theory. *Journal of Logic and Computation*, 15(2):99–112, 2005.

Robin Cooper. Austinian Truth, Attitudes and Type Theory. *Research on Language and Computation*, 3:333–362, 2006a.

Robin Cooper. A Type Theoretic Approach to Information State Update in Issue-based Dialogue Management. In Larry Moss, editor, *Jon Barwise Memorial Volume*. Indiana University Press, 2006b. To appear.

Robin Cooper and Jonathan Ginzburg. Using dependent record types in clarification ellipsis,. In *Proceedings of the sixth Workshop on the Semantics and Pragmatics of Dialogue (Edilog)*, pages 45–52. Edinburgh University, 2002.

Ann Copestake, Dan Flickinger, Ivan Sag, and Carl Pollard. Minimal recursion semantics: An introduction. Technical report, Stanford University, Stanford, CA, 1999.

Thierry Coquand, Randy Pollack, and Makoto Takeyama. A logical framework with dependently typed records. *Fundamenta Informaticae*, 20:1–22, 2004.

Haskell Brooks Curry and Robert Feys. *Combinatory Logic*, volume 1. North Holland, Amsterdam, 1958.

Walter Daelemans and Véronique Hoste. Evaluation of machine learning methods for natural language processing tasks. In *In Proceedings of the third International Conference on Language Resources and Evaluation (LREC-02)*, pages 755–760, 2002.

Walter Daelemans, Jakub Zavrel, Ko van der Sloot, and Antal van den Bosch. TiMBL: Tilburg Memory Based Learner, v. 5.0, Reference Guide. Technical Report ILK-0310, University of Tilburg, 2003.

Mary Dalrymple, Fernando Pereira, and Stuart Shieber. Ellipsis and Higher Order Unification. *Linguistics and Philosophy*, 14:399–452, 1991.

Paul Dekker. Topical restriction and answerhood. In *Proceedings of the Sinn und Bedeutung Workshop*, Konstanz, 2003a.

Paul Dekker. Questions in context. In R. Asatiani, K. Balogh, G. Chikoidze, P. Dekker, and D. de Jongh, editors, *Proceedings of the 5th International Tbilisi Symposium on Language, Logic and Computation*, Tbilisi, Gerogia, 2003b.

Barbara Di Eugenio and Michael Glass. The kappa statistic: A second look. *Computational Linguistics*, 30(1):95–101, 2004.

Ulrich Endriss, Nicolas Maudet, Fariba Sadri, and Francesca Toni. Protocol Conformance for Logic-based Agents. In G. Gottlob and T. Walsh, editors, *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-2003)*, pages 679–684. Morgan Kaufmann Publishers, 2003.

Elisabeth Engdahl, Steffan Larsson, and Stina Ericsson. Focus-ground articulation and parallelism in a dynamic model of dialogue. Technical report, Trindi: Task Oriented Instructional Dialogue, 2000. Accessible from http://www.ling.gu.se/research/projects/trindi.

Gregor Erbach. ProFIT: Prolog with features, inheritance and templates. In *Proceedings of the Seventh European Conference of the Association for Computational Linguistics*, pages 180–187, 1995.

Stina Ericsson. *Information Enriched Constituents in Dialogue*. PhD thesis, Göteborg University, 2005.

Nicholas Fay, Simon Garrod, and Jean Carletta. Group discussion as interactive dialogue or serial monologue: The influence of group size. *Psychological Science*, pages 481–486, 2000.

Raquel Fernández. A Dynamic Logic Formalisation of Inquiry-Oriented Dialogues. In *Proceedings of the 6th CLUK Colloquium*, pages 17–24, Edinburgh, UK, 2003a.

Raquel Fernández. A Dynamic Logic Formalisation of the Dialogue Gameboard. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics. Student Research Workshop*, pages 17–24, Budapest, Hungary, 2003b.

Raquel Fernández. The Dynamics of Utterances: Grounding and Update in Type Theory with Records. In *Proceedings of the Workshop on Discourse Domains and Information Structure (ESSLLI 2005)*, Edinburgh, UK, 2005.

Raquel Fernández and Ulle Endriss. Abstract Models for Dialogue Protocols: A Preliminary Report. Technical Report TR-03-03, Department of Computer Science, King's College London, July 2003a.

Raquel Fernández and Ulle Endriss. Towards a Hierarchy of Abstract Models for Dialogue Protocols. In R. Asatiani, K. Balogh, G. Chikoidze, P. Dekker, and D. de Jongh, editors, *Proceedings of the 5th International Tbilisi Symposium on Language, Logic and Computation*, pages 75–82. ILLC, October 2003b.

Raquel Fernández and Jonathan Ginzburg. Non-sentential utterances: A corpus study. *Traitement automatique des languages. Dialogue*, 43(2):13–42, 2002.

Raquel Fernández and Matthew Purver. Information State Update: Semantics or Pragmatics? In *Proceedings of the 8th Workshop on the Semantics and Pragmatics of Dialogue (Catalog)*, pages 20–27, Barcelona, Spain, July 2004.

Raquel Fernández, Jonathan Ginzburg, and Shalom Lappin. Classifying Ellipsis in Dialogue: A Machine Learning Approach. In *Proceedings of the 20th International Conference on Computational Linguistics, COLING 2004*, pages 240–246, Geneva, Switzerland, 2004.

Raquel Fernández, Jonathan Ginzburg, and Shalom Lappin. Automatic bare sluice disambiguation in dialogue. In *Proceedings of the 6th International Workshop of Computational Semantics (IWCS-6)*, pages 115–127, Tilburg, The Netherlands, 2005a.

Raquel Fernández, Jonathan Ginzburg, and Shalom Lappin. Using Machine Learning for Non-Sentential Utterance Classification. In *Proceedings of the 6th SIGdial Workshop on Discourse and Dialogue*, Lisbon, Portugal, 2005b.

Raquel Fernández, Jonathan Ginzburg, Howard Gregory, and Shalom Lappin. SHARDS: Fragment resolution in dialogue. In H. Bunt and R. Muskens, editors, *Computing Meaning*, volume 3. Kluwer, in press.

Tim Fernando. Conservative generalized quantifiers and presupposition. In *Proceedings of the 11th Annual Semantics and Linguistic Theory Conference*, pages 172–191. NYU/Cornell, 2001.

Dan Flickinger, Ann Copestake, and Ivan Sag. HPSG analysis of English. In W. Wahlster and R. Karger, editors, *Verbmobil: Foundations of Speech-to-Speech Translation*, pages 254–263. Springer Verlag, Berlin, Heidelberg and New York, 2000.

FIPA. *Communicative Act Library Specification*. Foundation for Intelligent Physical Agents (FIPA), 2002. http://www.fipa.org/specs/fipa00037/.

Malte Gabsil and Oliver Lemon. Combining acoustic and pragmatic features to predict recognition performance in spoken dialogue systems. In *In Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, Barcelona, Spain, 2004.

Claire Gardent. Unifying parallels. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, Maryland, US, 1999.

Claire Gardent and Michel Kohlhase. Computing parallelism in discourse. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, Japan, 1997.

Claire Gardent, Michael Kohlhase, and Noor van Leusen. Corrections and higher order unification. In *Proceedings of KONVENS*, Bielefeld, Germany, 1996.

Roger Garside. The CLAWS word-tagging system. In R. Garside, G. Leech, and G. Sampson, editors, *The computational analysis of English: A corpus-based approach*, pages 30–41. Longman, Harlow, 1987.

Jonathan Ginzburg. *Semantics and Interaction in Dialogue*. CSLI Publications and University of Chicago Press, Stanford: California, forthcoming. Draft chapters available from http://www.dcs.kcl.ac.uk/staff/ginzburg.

Jonathan Ginzburg. Interrogatives: Questions, facts, and dialogue. In Shalom Lappin, editor, *Handbook of Contemporary Semantic Theory*. Blackwell, Oxford, 1996.

Jonathan Ginzburg. Structural mismatch in dialogue. In *Proceedings of first Workshop on the Semantics and Pragmatics of Dialogue (MunDial)*. Universität München, 1997.

Jonathan Ginzburg. Abstraction and ontology: questions as propositional abstracts in type theory with records. *Journal of Logic and Computation*, 2(15):113–118, 2005.

Jonathan Ginzburg. Ellipsis resolution with syntactic presuppositions. In H. Bunt and R. Muskens, editors, *Computing Meaning: Current Issues in Computational Semantics*. Kluwer, 1999.

Jonathan Ginzburg and Robin Cooper. Resolving Ellipsis in Clarification. In *Proceedings of the 39th Meeting of the Association for Computational Linguistics*, pages 236–243, Toulouse, 2001.

Jonathan Ginzburg and Robin Cooper. Clarification, Ellipsis, and the Nature of Contextual Updates. *Linguistics and Philosophy*, 27(3):297–366, 2004.

Jonathan Ginzburg and Raquel Fernández. Scaling up from dialogue to multilogue: some principles and benchmarks. In *Proceedings of the 43rd Meeting of the Association for Computational Linguistics*, Michigan, 2005a. Ann Arbor.

Jonathan Ginzburg and Raquel Fernández. Action at a distance: The difference between dialogue and multilogue. In C. Gardent and B. Gaiffe, editors, *Proceedings of the 9th Workshop on the Semantics and Pragmatics of Dialogue (Dialor)*, pages 85–92, Nancy, France, 2005b.

Jonathan Ginzburg and Dimitra Kolliakou. Unexpected asymmetries in the acquisition of non-sentential utterances, April 2004. Paper presented at the Child Language Research Forum, Standford.

Jonathan Ginzburg and Ivan Sag. *Interrogative Investigations*. CSLI Publications, Stanford, California, 2001.

Jonathan Ginzburg, Howard Gregory, and Shalom Lappin. SHARDS: Fragment resolution in dialogue. In H. Bunt, I. van der Suis, and E. Thijsse, editors, *Proceedings of the Fourth International Workshop on Computational Semantics (IWCS)*, 2001.

Robert Goldblatt. *Logics of Time and Computation*. Lecture Notes. CSLI Publications, 1992.

Jeroen Groenendijk and Martin Stokhof. Dynamic Predicate Logic. *Linguistics and Philosophy*, 14(1):39–100, 1991.

Jeroen Groenendijk and Martin Stokhof. Questions. In J. van Benthem and A. ten Meulen, editors, *Handbook of Logic and Language*, pages 1055–1124. MIT Press, 1997.

Barbara Grosz and Candy Sidner. Plans for discourse. In P. Cohen, J. Morgana, and M. Pollack, editors, *Intentions in Communication*. MIT Press, 1990.

Charles L. Hamblin. *Fallacies*. Methuen, London, 1970.

David Harel, Dexter Kozen, and Jerzy Tiuryn. *Dynamic Logic*. Foundations of Computing Series. The MIT Press, 2000.

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

John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 2nd edition, 2001.

David Israel and John Perry. What is information? In Phillip Hanson, editor, *Information, Language and Cognition*, pages 1–19. University of British Columbia Press, Vancouver, 1990.

Andrew J. I. Jones and Xavier Parent. Conventional Signalling Acts and Conversation. In F. Dignum, editor, *Advances in Agent Communication*, volume 2922 of *LNAI*. Springer-Verlag, 2004.

Hans Kamp and Uwe Reyle. *From Discourse to Logic*. Kluwer Academic Publishers, 1993.

David Kaplan. Demonstratives: An essay on the semantics, logic, metaphysics, and epistemology of demonstratives and other indexicals. In *Themes from Kaplan*, pages 481–614. Oxford University Press, 1989.

Ruth Kempson, Willfried Meyer-Viol, and Dov Gabbay. *Dynamic Syntax. The Flow of Language Understanding*. Blackwell, 2000.

Emiel Krahmer and Paul Piwek. Presupposition projection as proof construction. In *Computing Meaning: Current Issues in Computational Semantics*, Studies in Linguistics & Philosophy series. Kluwer Academic Publisher, 1998.

Jörn Kreutel and Colin Matheson. Modelling questions and assertions in dialogue using obligations. In *Proceedings of the 3rd Workshop on the Sematics and Pragmatics of Dialogue (Atrmstelog 99)*, Amsterdam, 1999.

Manfred Krifka. For a structured account of questions and answers. In C. Smith, editor, *Proceedings of the Workshop on Spoken and Written Text*. University of Texas at Austin, 1999.

Staffan Larsson. Interactive communication management in an issue-based dialogue system. In *Proceedings of the 7th Workshop on the Semantics and Pragmatics of Dialogue (DiaBruck)*, 2003.

Staffan Larsson. *Issue-based Dialogue Management*. PhD thesis, Göteborg University, 2002.

Staffan Larsson and David Traum. Information State and Dialogue Management in the TRINDI Dialogue Move Engine. *Natural Language Engineering*, pages 323–340, 2000. Special Issue on Best Practice in Spoken Language Dialogue Systems Engineering.

Staffan Larsson, Peter Ljunglöf, Robin Cooper, Elisabet Engdahl, and Stina Ericsson. GoDiS: An Accommodating Dialogue System. In *Proceedings of ANLP/NAACL-2000 Workshop on Conversational Systems*, pages 7–10. Association for Computational Linguistics, 2000.

Zhang Le. Maximum Entropy Modeling Toolkit for Python and C++, 2003. http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html.

Geoffrey Leech, Roger Garside, and Michael Bryant. The large-scale grammatical tagging of text: experience with the British National Corpus. In N. Oostdijk and P. de Haan, editors, *Corpus-based Research into Language*, pages 47–63. Rodopi, Amsterdam, 1994.

Stephen C. Levinson. *Pragmatics*. Cambridge University Press, Cambridge, 1983.

Ian Lewin. The Autoroute dialogue demonstrator. Technical Report CRC-073, SRI International, Cambridge Computer Science Research Centre, 1998.

David Lewis. Score keeping in a language game. *Journal of Philosophical Logic*, 8:339–359, 1979.

Harry R. Lewis and Christos H. Papadimitriou. *Elements of the Theory of Computation*. Prentice Hall, 2nd edition, 1998.

Robert Malouf. A comparision of algorithm for maximum entropy parameter estimation. In *Proceedings of the Sixth Conference on Natural Language Learning*, pages 49–55, 2002.

Colin Matheson, Massimo Poesio, and David Traum. Modelling grounding and discourse obligations using update rules. In *Proceedings of the first Annual Meeting of the North American Chapter of the Association for Computational Linguistics*, Seattle, 2000.

Jason Merchant. *The syntax of silence: Sluicing, islands, and the theory of ellipsis*. Oxford University Press, Oxford, 2001.

Jason Merchant. Fragments and ellipsis. *Linguistics and Philosphy*, 6(27):661–738, 2004.

Jason Merchant. *The Syntax of Silence: Sluicing, Islands, and Identity in Ellipsis*. PhD thesis, University of California, Santa Cruz, 1999.

David Milward. Dynamic dependency grammar. *Linguistics and Philosophy*, 17:561–605, 1994.

Richard Montague. Pragmatics. In Richmond Thomason, editor, *Formal Philosophy*. New Haven, Yale UP, 1974.

Jerry Morgan. Sentence fragments and the notion 'sentence'. In *Issues in Linguistics*, pages 719–751. University of Illinois Press, Urbana, 1973.

Jerry Morgan. Sentence fragments revisited. In B. Music, R. Graczyk, and C. Wiltshire, editors, *Parasession on Language in Context*, volume 25 of *CLS*, pages 228–241. Chicago Linguistic Society, 1989.

Reinhart Muskens. A compositional discourse representation theory. In *Proceedings of the 9th Amsterdam Colloquium*, pages 467–486, 1994.

Simon Parsons, Nick Jennings, and Carles Sierra. Agents that reason and negotiate by arguing. *Journal of Logic and Computation*, 8(3):261–292, 1998.

Jeremy Pitt and Abe Mamdani. A Protocol-based Semantics for an Agent Communication Language. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI-1999)*, pages 486–491. Morgan Kaufmann Publishers, 1999a.

Jeremy Pitt and Abe Mamdani. Communication Protocols in Multi-Agent Systems. In *Workshop on Specifying and Implementing Conversation Policies*, 1999b.

Paul Piwek. *Logic, Information and Conversation*. PhD thesis, Eindhoven University of Technology, 1998.

Massimo Poesio and Reinhard Muskens. The dynamics of discourse situations. In P. Dekker and M. Stokhof, editors, *Proceedings of the 11th Amsterdam Colloquium*, 1997.

Massimo Poesio and David Traum. Towards an axiomatization of dialogue acts. In J. Hulstijn and A. Nijholt, editors, *Proceedings of the Twente Workshop on the Formal Semantics and Pragmatics of Dialogue*, pages 207–222, Enschede, May 1998.

Massimo Poesio and David Traum. Conversational actions and discourse situations. *Computational Intelligence*, 13(3):309–347, 1997.

Massimo Poesio, Rahul Mehta, Axel Maroudas, and Janet Hitzeman. Learning to resolve bridging references. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 144–151, Barcelona, Spain, 2004.

Richard Power. The organization of purposeful dialogues. *Linguistics*, 17:107–152, 1979.

Stephen G. Pulman. Higher Order Unification and the Interpretation of Focus. *Linguistics and Philosophy*, 20:73–115, 1997.

Matthew Purver. CLARIE: the Clarification Engine. In J. Ginzburg and E. Vallduví, editors, *Proceedings of the 8th Workshop on the Semantics and Pragmatics of Dialogue (Catalog)*, pages 77–84, Barcelona, 2004a.

Matthew Purver. SCoRE: A tool for searching the BNC. Technical Report TR-01-07, Department of Computer Science, King's College London, 2001.

Matthew Purver. *The Theory and Use of Clarification Requests in Dialogue*. PhD thesis, King's College, University of London, 2004b.

Matthew Purver and Raquel Fernández. Utterances as Update Instructions. In *Proceedings of the 7th Workshop on the Semantics and Pragmatics of Dialogue (DiaBruck)*, pages 115–122, Saarbrücken, Germany, September 2003.

Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Francisco, 1993.

Arne Ranta. *Type-theoretical grammar*. Clarendon Press, Oxford, 1994.

Craige Roberts. Information structure in discourse. In *Ohio Working Papers in Semantics*. Ohio State University, Columbus, 1996.

Kepa Rodríguez and David Schlangen. Form, intonation and function of clarification requests in German task oriented spoken dialogues. In *Proceedings of the 8th Workshop on the Semantics and Pragmatics of Dialogue (Catalog)*, 2004.

John Ross. Guess who. In *Proceeding of the 5th annual Meeting of the Chicago Linguistics Society*, pages 252–286, Chicago, 1969.

David Sadek. Dialogue acts are rational plans. In *Proceedings of the ESCA/ETR Workshop on Multi-modal Dialogue*, pages 1–29, Maratea, Italy, 1991.

Ivan A. Sag. English relative clause constructions. *Journal of Linguistics*, 33:431–484, 1997.

Emanuel A. Schegloff and Harvey Sacks. Opening up closings. *Semiotica*, 4(7):289–327, 1973.

David Schlangen. Towards finding and fixing fragments: Using ML to identify non-sentential utterances and their antecedents in multi-party dialogue. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, USA, 2005. Ann Arbor.

David Schlangen. Causes and strategies for requestiong clarification in dialogue. In *Proceedings of the 4th SIGdial Workshop on Discourse and Dialogue*, 2004.

David Schlangen. *A Coherence-Based Approach to the Interpretation of Non-Sentential Utterances in Dialogue*. PhD thesis, University of Edinburgh, Scotland, 2003.

David Schlangen and Alex Lascarides. Resolving fragments using discourse information. In *Proceedings of the 6th International Workshop on the Semantics and Pragmatics of Dialogue (Edilog)*, pages 161–168, Edinburgh, 2002.

David Schlangen and Alex Lascarides. The interpretation of non-sentential utterances in dialogue. In *Proceedings of the 4th SIGdial Workshop on Discourse and Dialogue*, 2003.

Sidney Siegel and John N. Castellan. *Nonparametric Statistics for the Behavioral Sciences*. McGraw-Hill, 1988.

Munindar P. Singh. Agent Communication Languages: Rethinking the Principles. *IEEE Computer*, 31(12):40–47, 1998.

Robert Stalnaker. Assertion. *Syntax and Semantics*, 9, 1978. New York Academic Press.

Tim Stowell. *Origins of Phrase Structure*. PhD thesis, MIT, 1981.

Paul Taylor, Simon King, Stephen Isard, and Helen Wright. Intonation and dialogue context as constraints for speech recognition. *Language and Speech*, 41(3):493–512, 1998.

David Traum. *A Computational Theory of Grounding in Natural Language Conversation*. PhD thesis, University of Rochester, Department of Computer Science, Rochester, NY, 1994.

David Traum. Semantics and pragmatics of questions and answers for dialogue agents. In *Proceedings of the International Workshop on Computational Semantics (IWCS'03)*, pages 380–394, Tilburg, The Netherlands, 2003.

David Traum and James Allen. Discourse obligations in dialogue processing. In *Proceedings of the 32nd annual meeting of the Association for Computational Linguistics*, 1994.

David Traum, Johan Bos, Robin Cooper, Staffan Larsson, Ian Lewin, Colin Matheson, and Massimo Poesio. A model of dialogue moves and information state revision. Technical report, 1999. The TRINDI project, Deliverable D2.1.

Bonnie Webber. *A Formal Approach to Discourse Anaphora*. PhD thesis, Hardvard University, 1978.

Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools with Java implementations*. Morgan Kaufmann, San Francisco, 2000. http://www.cs.waikato.ac.nz/ml/weka.

Helen Wright, Massimo Poesio, and Stephen Isard. Using high level dialogue information for dialogue act recognition using prosody features. In *Proceedings of the ESCA Workshop on Prosody and Dialogue*, Eindhoven, The Netherlands, 1999.