Techniques for utterance disambiguation in a human-computer dialogue system

Pontus Lurcock

a thesis submitted for the degree of

Master of Science

at the University of Otago, Dunedin,

New Zealand.

28 January 2005

Abstract

Disambiguating an utterance occurring in a dialogue context is a complex task, which requires input from many different sources of information—some syntactic, some semantic, and some pragmatic.

The central question addressed by this thesis is how to integrate data sources for utterance disambiguation within a bilingual human-computer dialogue system. First, a simple scheme is proposed for classifying disambiguation data sources; then this scheme is used to develop a method for combining data sources in a principled manner. Next, several actual sources of disambiguation data are explored; each is fitted into the previously described implementation framework. In particular, a probabilistic grammar is developed and augmented using novel techniques to increase its performance with respect to the local dialogue context.

In a dialogue system, ambiguities which cannot be resolved automatically can be clarified by asking the user what was meant. This thesis also presents a model of clarification subdialogues which is integrated within the utterance disambiguation framework. This is followed by a brief treatment of how user errors may be accommodated, and how this process can also be fitted—conceptually and in implementation—into the previously described disambiguation framework.

Finally, I describe the details of implementing these techniques within an existing dialogue system, and give examples demonstrating their effectiveness.

Acknowledgements

It is of course obligatory to thank one's supervisor in the acknowledgements section, but my gratitude to to Alistair Knott is entirely sincere. Without use of violence or threats, he has somehow shepherded me through the forging of this thesis at a rate unheard of in the tranquil milieu of the AI back lab. I doubt that I could have completed this dissertation in the allotted time but for Ali's cheerful yet steadfast belief that I was capable of doing so.

I thank Peter Vlugter, main author of the current Te Kaitito architecture, for providing an excellent platform upon which to implement the work of this thesis; he has also been my intrepid guide through many a hair-raising MRS structure.

I thank my parents, for their unstinting support and their forbearance of my globetrotting ways; Peter Petchey, for lending me a beautiful place to live while I was writing up; the garden-shedsters and folks back home, for staying in touch even when I didn't; Bea Hudson, for care, feeding, and ginger beer; and the whole spectacular swarm of sauna-bathers, trampers, tree-huggers, morris-dancers, musicians, capoeiristas, croquet-istas, French-speakers, and fire-jugglers I have encountered in Dunedin, for making my year here so remarkable and enjoyable.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

> . . . ‘Cyclop! now,
> As thou demand’st, I’ll tell thee my name, do thou
> Make good thy hospitable gift to me.
> My name is No-Man; No-Man each degree
> Of friends, as well as parents, call my name.’
>
> . . . For other Cyclops. . . [a]sk’d him, if his fright
> Came from some mortal that his flocks had driven?
> Or if by craft, or might, his death were given?
> He answer’d from his den: ‘By craft, nor might,
> No-Man hath given me death.’ They then said right,
> If no man hurt thee, and thyself alone,
> That which is done to thee by Jove is done;
> And what great Jove inflicts no man can fly.’
>
> —*The Odyssey* (Homer, 1857)

1.1 Ambiguity in language

Ambiguity has a distinguished history as a bane of natural language processing (see, for example, Jurafsky and Martin, 2000, pp. 372–376). The consequences are not always as serious as they were for Polyphemus the Cyclops, whose companions failed to resolve *No-Man* correctly as a proper noun, but disambiguation has nevertheless merited serious attention.

Human disambiguation is so effective that the ambiguity in even seemingly straightforward utterances can be invisible until it becomes a stumbling block for a machine

grammar. The human advantage seems to stem to a large extent from the ability to apply a broad range of semantic-level information about the context, the speaker and the world as the sentence is incrementally parsed (see e.g. Swinney, 1979).

Much recent and current work in machine disambiguation—at least at the syntactic level—has focused, with great success, upon statistical parsing (e.g. Collins, 1997; Charniak, 1997), also called probabilistic or stochastic parsing. A corpus of sentences is annotated with parses known to be correct, and the probabilities of various events (usually rule productions) are calculated for the corpus as a whole. These probabilities are then used to select between different candidate parses of new utterances.

Statistical parsing, however, is not sufficient in every case. Its sensitivity to context is limited: although probabilities can be conditioned on contextual features, trade-offs must be made for computational tractability, and using more complex events makes the training data more sparse. Also, probabilistic parsing is not sensitive to semantic information that can override syntactic statistics. For example,

(1.1) There was a girl with a telescope in the room.
John saw the girl with the telescope.

Here, normal statistical parsing would have trouble: is John seeing with the telescope, or is he seeing a girl-with-telescope? For a human reader, the context largely dispels the ambiguity: we already know of a girl-with-telescope, so it seems reasonable to assume that this is what is being referred to. A sophisticated lexicalized probabilistic grammar, with information that 'seeing with a telescope' is more usual than 'girl with a telescope', might well choose the wrong parse. However, even a relatively simple semantic analysis would reveal that the existence of a girl-with-telescope is already known, vastly increasing the probability that this girl-with-telescope is involved in the correct parse.

My thesis is concerned with the disambiguation task in a particular NLP application: human-computer dialogue. The dialogue system I am working with is the kind that develops deep semantic representations of utterances rather than using robust, surface-oriented, Eliza-style techniques. With these deep representations at our disposal, it seems sensible to make use of this semantic information for disambiguation. But as soon as more than one source of disambiguation data becomes available, we face a problem: how are we to combine the different sources? What are we to do if they disagree? And what are we to do if we cannot come to a reasonable decision? This thesis attempts to answer these questions: it proposes a system for classifying different sources of disambiguation data; it describes a procedure for integrating these sources; it details

several sources and their place in this scheme; and it describes fall-back techniques for use when disambiguation fails.

1.2 The interpretation pipeline

In the field of natural language processing, the **pipeline** is a common metaphor for the interpretation process: raw data comes in at one end, usually in the form of a textual string or an acoustic signal, and is passed through a sequence of processing modules which convert it to progressively deeper semantic forms. Figure 1.1 shows a typical arrangement of this type.

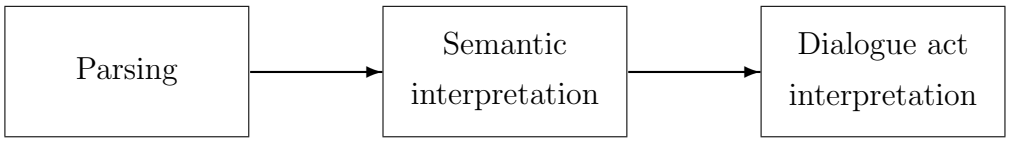

Figure 1.1: A typical interpretation pipeline

The illustrated pipeline is one that might be employed in a text-driven dialogue system. The pipelines for other systems might be different: a speech-driven system might additionally have a 'phonetic interpretation' module at the beginning; a system concerned with continuous text rather than dialogue might lack the final 'dialogue act interpretation' module. But in all cases, the direction of the pipeline is the same: from shallow, surface features like acoustic waveforms to deep semantic features like dialogue acts. In this thesis, the ordering imposed by the interpretation pipeline will be used in a variety of contexts: not only in the initial interpretation, but in data classification, disambiguation, and clarification.

1.3 A pipelined model of ambiguity

In practice, the interpretation process might resemble not so much a pipeline as an anatomical diagram of the circulatory system: just as the aorta divides and subdivides into smaller and smaller arteries, ambiguity can split the interpretation pipeline at any module; these divisions might then be subdivided again by subsequent modules. Figure 1.2 shows how ambiguity can grow geometrically with the number of modules in the pipeline: each interpretation of an ambiguous acoustic signal has several parses,

each of which may have more than one semantic attachment, each of which might be interpreted as more than one dialogue act.

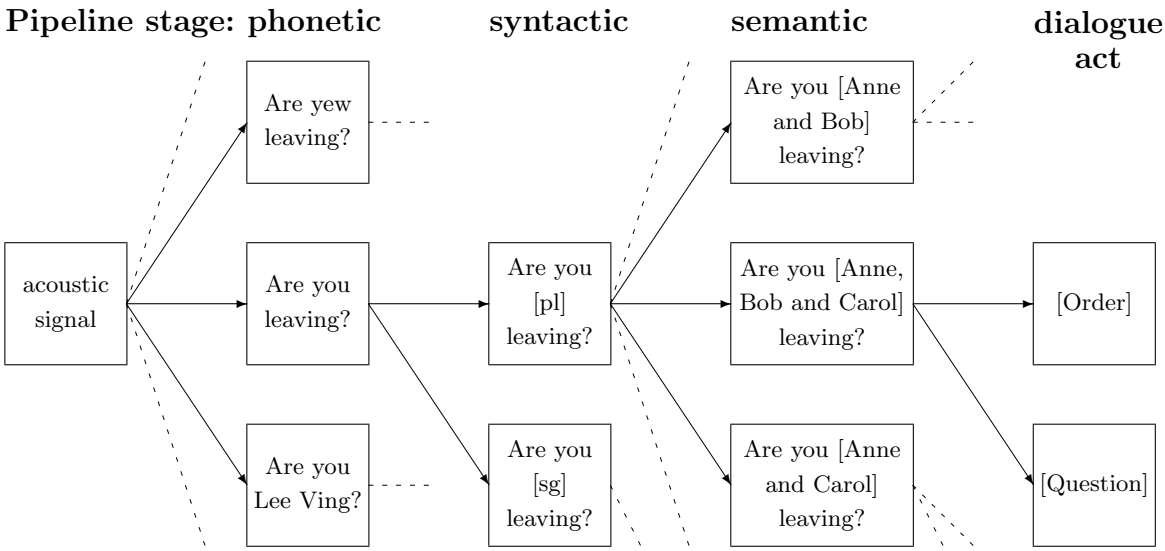

Figure 1.2: Ambiguity explosion in the interpretation pipeline. The central, horizontally aligned row of boxes shows the speaker's intended interpretation, while the branches from the central path show other possible interpretations which must be discarded somehow. Dotted lines indicate further interpretations omitted for brevity.

1.4 Outline of the thesis

Chapter 2 reviews the relevant literature on disambiguation, data integration and explicit clarification, both in general and in dialogue systems. It also introduces Te Kaitito, the bilingual human-machine dialogue system within which the proposed techniques will be implemented.

Chapter 3 forms the nucleus of this thesis. In it, I describe a simple framework for the classification of disambiguation data sources. With reference to this classification, I then outline a general model of how different data sources can be prioritized and integrated; the work of the subsequent chapters takes place within the framework of this model.

Chapter 4 goes on to describe several actual sources of disambiguation data available to the Te Kaitito system, and the practical means by which they may be classified and integrated according to the model described in Chapter 3. One of the most important sources is a statistical parser, augmented in a novel fashion to give it greater sensitivity

to the current dialogue context. In this chapter I also describe sources which exist at the semantic and dialogue act levels.

One of the advantages of a dialogue system is that, if the usual disambiguation procedure cannot make a sufficiently reliable judgement, the user can be queried explicitly to resolve the ambiguity. Chapter 5 presents a method whereby such clarification subdialogues may be incorporated into the disambiguation framework proposed in Chapter 3. Particular attention is paid to disambiguation at the syntactic level using rephrasing: a theoretical framework is established, and some general results derived, before various actual clarification techniques are evaluated according to well-defined metrics. Finally, there is a brief treatment of other syntactic clarification techniques, and of semantic clarification.

Chapter 6 discusses ways in which accommodation can be made for errors in the user's input, and how this accommodation can be seen as part of the disambiguation process and integrated into it.

In Chapter 7, I describe the implementation of the integration architecture for disambiguation data within Te Kaitito, of the modules which provide disambiguation data, and of clarification subdialogues.[1] I also provide dialogue transcripts from sessions with the system showing these systems in operation.

In Chapter 8, I sum up the theoretical and practical results of the thesis, and their usefulness in further work on Te Kaitito and further afield. I conclude with an account of extensions which could be made to my work, and describe its relevance to the future development of Te Kaitito.

Much of the work described in Chapters 3, 4, 5, and 7 has previously been summarized in Lurcock, Vlugter, and Knott (2004).

[1]The error handling described in Chapter 6 is not implemented.

Chapter 2

Literature Survey

> . . . my father had the happiness of reading the oddest books in the universe, and had moreover, in himself, the oddest way of thinking that ever man in it was bless'd with. . .
>
> —Laurence Sterne, *The Life and Opinions of Tristram Shandy* (Sterne, 1912)

In this chapter, I will introduce the main systems and techniques that underlie this thesis. In Section 2.1.1, I describe Te Kaitito, the human-computer dialogue system within which this project is implemented; in Section 2.2 I review some existing approaches to the integration of disambiguation data sources. In Section 2.3 I describe the techniques of probabilistic parsing, with particular attention to those relevant to the syntactic-level disambiguation module implemented in this project. Previous work relating to my treatment of clarification subdialogues (Chapter 5) and error handling (Chapter 6) is reviewed within those chapters.

2.1 The Te Kaitito system

2.1.1 Introduction

Te Kaitito is a human-computer dialogue system designed to teach Māori to speakers of English. Its architecture is described in detail by Knott, Bayard, de Jager, and Wright (2002).

Te Kaitito uses the LKB system (Copestake and Flickinger, 2000) for parsing and generation. LKB makes use of grammars written in an HPSG-style formalism. Te Kaitito can currently use two such grammars: one is the English Resource Grammar (**ERG**) (Flickinger, 2000); the other a small, custom-written Māori-English Grammar,

the **MEG** (Bayard, Knott, and de Jager, 2002). The MEG comprises a set of rules and lexical items covering a set of English sentences along with their Māori translations, so that interpreting a sentence in one language creates a logical form from which its translation into the other language can be generated. Every language-specific rule or word is tagged with an appropriate 'language' feature, preventing the mixing of languages within a sentence. Of course, only the MEG is of any use in the teaching of Māori; the ERG is nevertheless useful during development, and to ensure that Te Kaitito maintains a degree of grammar independence.

The parsing process produces a flat semantic representation in the Minimal Recursion Semantics (MRS) format (Copestake, Flickinger, and Sag, 1999). The MRS form is transformed into a Discourse Representation Structure (DRS) (Kamp, van Genabith, and Reyle, in preparation), which splits an utterance into an assertion and a set of presuppositions such as anaphora and definite NPs. Te Kaitito attempts to resolve these presuppositions with reference to the current dialogue context, then identifies the dialogue act which the resolved utterance performs. Further modules then deal with the task of creating, formulating and delivering a suitable response.

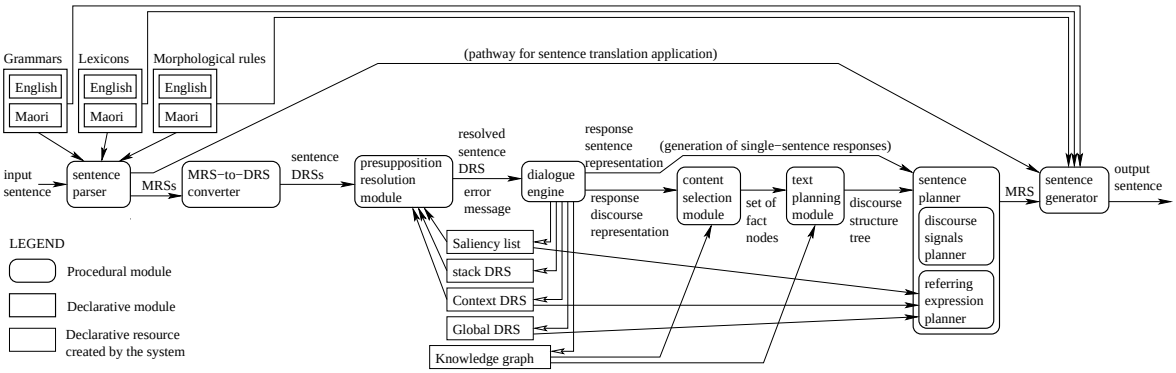

Figure 2.1: Structure of Te Kaitito prior to the addition of my disambiguation framework (from Knott et al., 2002).

As can be seen from Figure 2.1, Te Kaitito's interpretation process exhibits the pipeline structure described in Section 1.2: the user's initial input, supplied as a string of text, is passed through a sequence of modules, undergoing successive transformations. Each module can—in addition to its input along the pipeline—incorporate extra data from the system's current state into its output, producing a richer representation. The pipeline continues beyond the dialogue engine to produce a response, but this thesis is mainly concerned with the interpretation section.

Prior to the commencement of this thesis, Te Kaitito's disambiguation facilities

were quite basic. There were some simple procedural rules which decided whether an indicative sentence was an assertion or the answer to a question by consulting the current dialogue context (described in Section 4.3.1), and syntactic disambiguation simply consisted of choosing a parse at random.

Few aspects of this thesis will make use of the bilingual features of the Māori-English Grammar (the main exception being the unambiguous rephrasings mentioned in Section 5.3). Other aspects of the work require the use of the ERG—chiefly the availability of a large treebank for the training of probabilistic grammars. Most of the examples given will therefore use the ERG.

Te Kaitito has a modular architecture allowing the core processing pipeline to be easily attached to different user interfaces. At present a console-based text interface and a World Wide Web interface are in use, although a three-dimensionally rendered 'talking head' (described in King, Knott, and McCane, 2003) has also been implemented.

2.1.2 Presupposition resolution in Te Kaitito

As mentioned in the previous section, Te Kaitito's interpretation pipeline includes a **presupposition resolution** module. Once an utterance has been broken down into an assertion (the new information imparted by the utterance) and a set of presuppositions (references to the previously existing dialogue context), this module attempts to bind the presuppositions to pre-existing relations in the dialogue context. For instance, in the following example

(2.1)

User	Aaron hits a dog
System	okay
User	the dog chases him
System	okay

the user's second utterance contains two presuppositions: *the dog* and *him*. The system's response indicates that it has successfully bound these—in this case, to the previously mentioned *a dog* and *Aaron*, respectively.

It may be that a presupposition cannot be bound to any known relation. In this case, Te Kaitito makes use of **accommodation**, as described by Lewis (1979): it simply assumes the existence of the required relation. This is a common occurrence in human dialogue: if A says to B 'I have to walk the dog', B will naturally assume the existence of a dog, even if A has never mentioned it before; conversely, if a dog *has* recently featured in the dialogue, B will probably assume that A is referring to this dog rather than

some new, unknown dog. Lewis summarizes the process of accommodation elegantly: 'it's not as easy as you might think to say something that will be unacceptable for lack of required presuppositions. Say something that requires a missing presupposition, and straightaway that presupposition springs into existence, making what you said acceptable after all.'

Te Kaitito operates in much the same way. Indefinite NPs create new entities in the system's store of semantic knowledge. Definite NPs are resolved, if possible, to existing entities; otherwise, an entity is instantly created for the NP to refer to.

This process of presupposition matching and accommodation can be used as a method of disambiguation; the technique was first described by Knott and Vlugter (2003) and was implemented as part of this thesis. It will be described in detail in Section 4.2.

2.1.3 Sample dialogues

To give an impression of the type of interaction possible with Te Kaitito, I show in Figure 2.2 a typical dialogue session.

As the topics of discourse might suggest, the MEG is fairly restricted in its coverage; however, this is not a serious problem for its current application. It is common in language teaching environments to heavily restrict the discourse domain in order to avoid confusing language learners.

Work is currently under way, independently of this project, to allow the system to make dialogue initiatives (Slabbers, in preparation). However, the dialogue shown in Figure 2.2 is is entirely driven by the user; essentially, Te Kaitito acts as a semantic database with a natural language interface. The user can make assertions (Utterance 1) which are incorporated into Te Kaitito's model of the dialogue context, an operation known as grounding (see e.g. Traum, 1999). If the user's utterance is successfully interpreted, an acknowledgement is given (Utterance 2).

The user can add new assertions (Utterance 6) about previously mentioned entities by using definite NPs, and make queries (Utterance 3) to which the system responds using its current knowledge. The bilingual structure of the MEG (described in Section 2.1.1) allows the system to make responses in either English or Māori with equal ease; here it is configured to generate every possible response in both languages (Utterance 4). If Te Kaitito cannot fully interpret an utterance made by the user (here, Utterance 7), it can initiate (Utterance 8) a **clarification subdialogue**. Chapter 5 discusses these clarification subdialogues in detail; the subdialogue shown here is initiated

Speaker	*Utterance*	
User	a blue dog barked	(1)
System	Ok	(2)
User	a red dog walked	
System	Ok	
User	which dog walked	(3)
System	the red dog it was the red dog nā te kurī whero	(4)
User	Ok	(5)
User	the blue dog chased a green cat	(6)
System	Ok	
User	a weasel barked	
System	Ok	
User	a big cat chased the weasel	
System	Ok	
User	the cat barked	
System	Ok	
User	the red dog chased the weasel	
System	Ok	
User	the blue dog walked	
System	Ok	

Speaker	*Utterance*	
User	the dog ate the weasel	(7)
System	which dog ate the weasel	(8)
User	the red dog	
System	Ok	
System	Ok	
User	the blue dog chased a slow monkey	
System	Ok	
User	a loud monkey ate the blue dog	
System	Ok	
User	which monkey did the blue dog chase	
System	the slow monkey it was the slow monkey nā te makimaki pōturi	
User	Ok	
User	what chased the monkey	
System	the blue dog it was the blue dog nā te kurī kikorangi	

Figure 2.2: A dialogue session with Te Kaitito; adapted from de Jager, Knott, and Bayard (2002).

with a semantic-level clarification question, as described in Section 5.5.

This interaction model has utility for CALL: the user and the system can discuss a situation in Māori with the possibility of falling back to English if the user fails to understand an utterance.

2.1.4 Ambiguity in Te Kaitito

Ambiguity can arise in any of Te Kaitito's interpretation modules: a sentence can have multiple parses; each parse can have multiple ways to resolve presuppositions with respect to the dialogue context; and each resolved parse might be interpretable

as more than one dialogue act. This thesis is concerned mainly with resolution of ambiguity arising at the syntactic level; however, performing this task will involve using data from all stages of the processing pipeline.

2.2 Existing approaches to integrating disambiguation data

The idea of integrating information from disparate sources during sentence interpretation is certainly not a new one. Most large-scale systems perform an integration of some kind. It is less common to specifically develop a dedicated architecture for this kind of synthesis. In this section, I consider some of the work that has been done in this area. Once I have presented (in Chapter 3) my architecture for integration of disambiguation data in Te Kaitito, I will briefly compare it (in Section 3.2.5) to the schemes described below.

2.2.1 Interpretation as abduction

Hobbs, Stickel, Appelt, and Martin (1993) describe a remarkably broad and uniform method for sentence interpretation—including disambiguation—by the technique of abductive reasoning. In this framework, a sentence is seen a a consequent which may be inferred from a set of logical clauses. The system will probably already know some of these clauses to be true. Others will need to be accommodated or assumed: these correspond to the new information imparted by the sentence. A cost is associated with each type of assumption, and the system works backwards from the sentence to find a set of assumptions with minimal total cost, which is taken to be the correct interpretation. Essentially, abduction seeks to find a minimal explanation of how a sentence can be true. This framework allows uniform incorporation of syntactic, semantic, and pragmatic information. The chief focus of the paper is on pragmatics, aided by the construction of a large base of world and domain knowledge.

The interpretation process consists of deriving a logical form for the sentence, using facts in a knowledge base and axioms in a grammar. Often, the knowledge base will be insufficient to prove the sentence; in that case, one or more predicates must be *assumed* to be true. These predicates are the new information imparted by the sentence. (This division is analogous to Te Kaitito's use of DRS to separate an utterance's assertional and presuppositional content, as described in Section 2.1.1).

As an example, Hobbs et al. describe the interpretation of the sentence

(2.2) The Boston office called.

which corresponds to proving

$$(\exists x, y, z, e)\, call(e, x) \wedge person(x) \wedge rel(x, y) \wedge office(y) \wedge Boston(z) \wedge nn(z, y)$$

('there was a calling event e by a person x related by some relation *rel* to an office y, which is related by some relation *nn* to z, which is Boston').

I will first give the axioms necessary to prove this, and then the knowledge-base facts to which they must be applied.

$$\begin{aligned}(\forall w_1, w_2, y, p, e, x)\, & np(w_1, y) \wedge verb(w_2, p) \wedge \\ & p'(e, x)^{\$3} \wedge rel(x, y)^{\$20} \wedge Req(p, x)^{\$10} \supset s(w_1\, w_2, e)\end{aligned} \tag{2.3}$$

This can probably be most succinctly be explained using a term-by-term English gloss (ignoring, for the moment, the superscripts attached to some of the terms).

$np(w_1, y)$ $\wedge$ $verb(w_2, p)$ $\wedge$

If w_1 is a NP denoting y, and w_2 is a verb denoting a predicate p, and

$\wedge p'(e, x)^{\$3}$ $\wedge$ $rel(x, y)^{\$20}$

e is the eventuality of p holding for x, and x is somehow related to y,

$\wedge$ $Req(p, x)^{\$10}$ $\supset$

and x satisfies the requirements placed on it by p, then

$s(w_1\, w_2, e)$

$w_1\, w_2$ is a sentence expressing the eventuality e.

Interpreting the NP also requires the application of another axiom:

$$\begin{aligned}(\forall w_1, w_2, q, r, y, z)\, & det(the) \wedge noun(w_1, r) \wedge noun(w_2, q) \\ & \wedge r(z)^{\$5} \wedge q(y)^{\$10} \wedge nn(z, y)^{\$20} \supset np(the\, w_1\, w_2, y)\end{aligned} \tag{2.4}$$

('If *the* is a determiner, and w_1 and w_2 are nouns denoting the predicates r and q respectively, and r and q hold for the entities z and y respectively, and there is some implicit relation *nn* between z and y, then *the* $w_1\, w_2$ is a NP referring to the entity y'.)

The superscripted dollar amounts indicate the cost of assuming a conjunct to be true if it is not in the knowledge base. This cost corresponds to an intuitive notion of

how unlikely the assumption is; thus, it is easy (\$3) to assume that the information carried by the verb is true, but hard (\$20) to assume the existence of a 'Boston office' if the knowledge base doesn't encompass any office which is in any way related to Boston.

These axioms can then be combined with facts in the knowledge base:

$Boston(B_1)$	B_1 is the city of Boston.
$office(O_1) \wedge in(O_1, B_1)$	O_1 is an office in Boston (B_1).
$person(J_1)$	John (J_1) is a person.
$work\text{-}for(J_1, O_1)$	John (J_1) works for the office O_1.
$(\forall y, z) in(y, z) \supset nn(z, y)$	If y is in z, then a compound nominal can be formed from z and y.
$(\forall x, y) work\text{-}for(x, y) \supset rel(x, y)$	If x works for y, then y can be coerced into x.
$(\forall x) person(x) \supset Req(call, x)$	If x is a person, then x can call.

The process of interpretation consists of finding the minimum total cost which allows the sentence's logical expression to be true; the predicates which must be assumed correspond to the new information imparted by the sentence. In this case, all the antecedents of the axioms can be proved from the knowledge base except for $p'(e, x)$ (the information that someone called), which can be assumed for a cost of \$3, and incorporated into the knowledge base as a new fact.

The breadth of of this approach to interpretation and disambiguation comes from the fact that both syntactic and pragmatic data are placed within the same framework. Axiom 2.3 above is used to simultaneously derive the syntactic form of the sentence and its meaning; w_2 *is a verb* is placed on an equal footing with *a person can make a call*. The effectiveness of the method depends, of course, on choosing suitable values for the costs associated with assumptions, and on combining them in a suitable manner.

2.2.2 Eliminative parsing with graded constraints

Menzel and Schröder (1999) describes another method for utterance interpretation by combining information from a variety of data sources. The technique differs somewhat both from that of Hobbs et al. (1993) and from that devised in this thesis: it makes use of constraint-based, eliminative parsing, starting with a representation encoding every possible parse structure, and performing the rest of the parsing process as a disambiguation. This approach allows simultaneous use of both syntactic and semantic constraints in attempting to find a most likely interpretation. The system is, like the Te Kaitito system described in this thesis, a dialogue-based language learning environment,

and the focus is on the correct detection and diagnosis of errors while maintaining the robustness of the parser.

I reproduce here an example given by Menzel and Schröder of their eliminative parsing method. Both the grammar and the constraints are here extremely simplified for illustrative purposes. Suppose that we wish to parse the German sentence

(2.5) *Der Mann besichtigt den Markplatz.*
The man visits the marketplace.

using a grammar which contains only three relations: DET (the modification of a noun by a determiner), SUBJ (the modification of a finite verb by the head of a NP as the subject), and DOBJ (the modification of a finite verb by the head of a NP as the direct object). We begin with a completely ambiguous representation (Figure 2.3) in which every word modifies every other word using every possible relation.

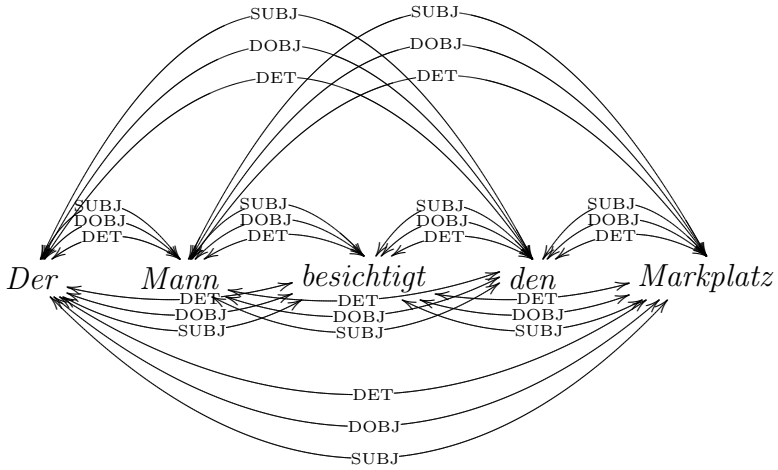

Figure 2.3: Eliminative parsing: the initial set of all possible relations. Each of the double-headed arrows represents *two* relations of the type corresponding to the arrow's label, one going in each direction.

This somewhat disconcerting structure can be radically pruned by the application of three unary constraints: 'a determiner modifies a noun to its right with DET'; 'a noun can modify a finite verb as either the subject or the direct object'; and 'a finite verb modifies nothing'. Figure 2.4 shows the result.

The structure is still ambiguous, but far less so. Two binary constraints can complete the disambiguation: 'a word form can not be modified twice with the same label', and 'nominative case is required for the complete subject phrase'. Figure 2.5 shows the fully disambiguated sentence.

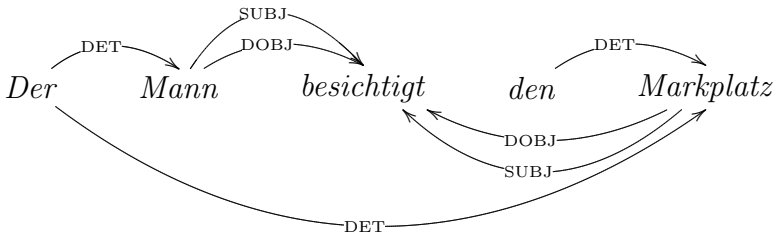

Figure 2.4: Remaining relations after application of unary constraints.

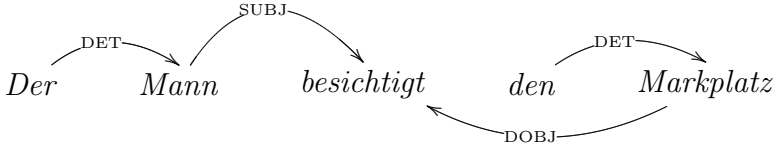

Figure 2.5: Fully disambiguated parse

Like the axioms of Hobbs et al., the constraints used in this interpretation process need not just be syntactic. Menzel and Schröder give a diagram of the relations, in the realms of syntax, semantics, and domain knowledge, constraining the interpretation of the sentence

(2.6) *Der Parkplatz liegt neben der Kirche.*
The car park lies beside the church.

Figure 2.6 shows the relations. The 'domain' knowledge relates to the particular situation under discussion; thus, in this case, the relation represents the system's pre-programmed knowledge that, in this case, the car park really *does* lie beside the church.

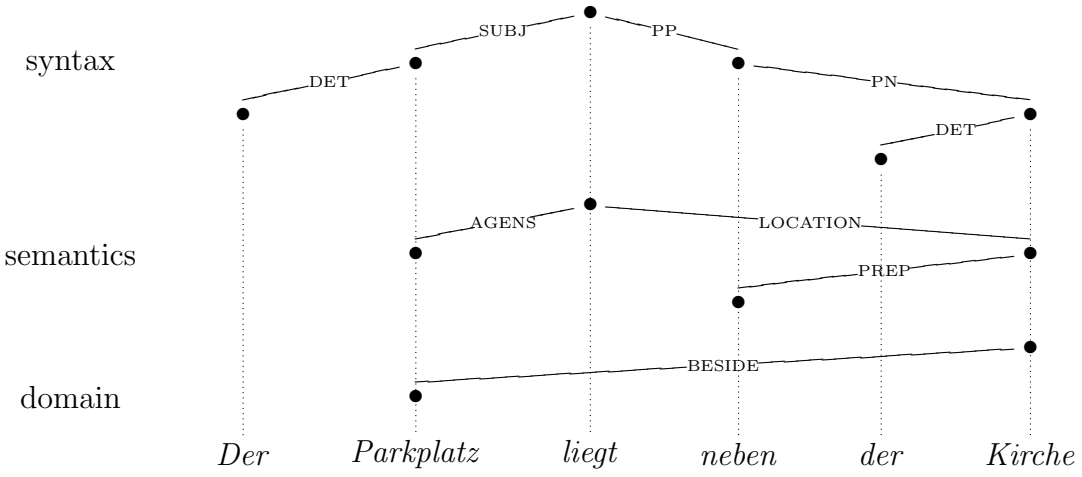

Figure 2.6: Constraints at multiple levels

The structures shows in Figure 2.6 are mapped to each other by constraints and disambiguated simultaneously. The constraints in this system are *graded*: each constraint

is associated with a weight representing the severity of violating it. It is possible that some constraints in a partial interpretation will conflict with each other. Disambiguation is then a process of seeking an interpretation which, in Menzel and Schröder's words, 'violates only as few and as weak constraints as possible'.

There are clear parallels with the work of Hobbs et al. In the abductive model, there may be too few facts to prove a sentence's logical form, in which case a minimal-cost set of facts is *assumed* and added to those available for the sentence's interpretation. In the eliminative model, there may be too many constraints to interpret a sentence consistently, in which case a minimal-cost set of constraints is *discarded*. The processes may, to an extent, be seen as mirror images of one another.

2.2.3 Other approaches

Rosenfeld (1996) also tackles the problem of integrating disparate sources of disambiguation data: he uses a Maximum-Entropy framework to allow the integration of any data source which can be described in terms of statistics on a text. However, this does not allow the incorporation of higher semantic levels of information, as explored in this thesis.

2.3 Probabilistic parsing

A 'traditional' grammar makes a stark binary decision about a sentence: either it is a valid sentence of the language defined by the grammar, or it is not. This makes disambiguation very hard: ideally, a grammar would be defined so as to return exactly one parse for any valid input, and exactly zero parses for any invalid input. In practice, this proves impossible: even humans, with a wealth of higher-level information at their disposal, are occasionally incapable of reliably choosing a single correct syntactic parse of a sentence. Natural language is too inherently ambiguous to be definable by such strict set inclusion. Since it is usually considered less harmful to generate spurious interpretations of sentences than to miss out valid interpretations, grammars have tended to overgenerate rather than undergenerate. The problem then becomes how to select the best or most likely of several possible parses.

Statistical, or probabilistic, parsing techniques have proved very successful in this task (see Abney, 1996; Manning and Schütze, 1999). Probabilistic parsing generally consists of annotating every grammatical rule in some way with a probability corresponding to the frequency of that rule's use. During the parsing of a sentence, the

probabilities of individual rules are combined in such a way as to calculate the overall probability of that parse occurring. Ideally, a parse produced by such a grammar can be understood in relation to an imaginary collection of every possible (parsed) sentence in the grammar's language, in which the number of instances of a sentence is proportional to the frequency of its use. The probability assigned by the grammar should, then, be the probability that a randomly selected sentence from this collection will correspond exactly to the parse under consideration. It is thus likely to be a minuscule number for any realistic grammar; nevertheless, the probabilities for alternative parses can be compared and the most likely one selected.

Such a collection is, of course, impossible to assemble and, being infinite, would in any case be impossible to make use of. But remarkably effective implementations can be produced by inferring probabilities from relatively small treebanks of parsed sentences. Data sparsity is always a problem, but it can be alleviated by the use of backoff (see e.g. Collins, 1997) and smoothing (a survey of smoothing techniques is given in Chen and Goodman, 1996).

In practice, implementations vary widely: the underlying grammar formalism, the amount of contextual conditioning on the probabilities, and the resources available to infer the probabilities all affect the design of a practical parser.

2.3.1 Probabilistic context-free grammars

One of the simplest forms of probabilistic grammar, both in conception and implementation, is a probabilistic context-free grammar (PCFG). This technique appears first to have been described by Booth (1969), which is also one of the earliest descriptions of any probabilistic parsing technique. Each production rule is assigned a number corresponding to the probability of the production occurring given that its head occurs. Raw probabilities can thus be easily obtained by counting instances of a production and dividing by the number of occurrences of the non-terminal symbol at its head. Since the grammar is context-free, the overall probability of a parse can be calculated simply by multiplying the probabilities of all the rules used in producing it. In practice, the probabilities are so low that they can become inconvenient for human programmers to read and difficult for computers to process accurately due to arithmetic underflows. Probabilities are therefore usually stored as logarithms and combined by addition rather than multiplication.

2.3.2 Lexicalized probabilistic grammars

In its simplest form, a PCFG works only with rules and parts of speech and makes no use of data about individual words covered by the grammar. This approach can lead to problems. Consider, as a minimal example, a very simple PCFG for interpreting a restricted class of NPs. It has only two rules:

NP → Adjective Noun
NP → Noun Noun

Suppose that we have inferred a probability for each of these rules from corpus of disambiguated NPs. The problem quickly becomes apparent: whichever production is more common in the corpus will produce a higher probability for its associated rule, which will then be used in *every* case. This grammar could never assign different structures to 'ground coffee' and 'ground frost'. Similar problems affect larger PCFGs: the two sentences 'I ordered meatballs in gravy' and 'I ordered meatballs in desperation' must be given the same PP attachment by any non-lexicalized PCFG, leading to a preposterous interpretation for one of the sentences.

In an attempt to overcome problems such as these, most modern grammars (e.g. Charniak, 1997) make use of lexical dependency information. In the case of a PCFG, this can be thought of as replacing every non-lexicalized rule with a huge set of lexicalized rules such as

1. NP(coffee) → Adjective(ground) Noun(coffee)
2. NP(coffee) → Noun(ground) Noun(coffee)
3. NP(frost) → Adjective(ground) Noun(frost)
4. NP(frost) → Noun(ground) Noun(frost)

We might then hope, with a sufficiently large parsed corpus, to be able to infer high probabilities for rules 1 and 4, and low probabilities for rules 2 and 3. For relatively common collocations such as 'ground frost', this might be feasible; for grammars of realistic size it is impossible. Even to disambiguate the hugely restricted set of sentences 'I ordered *NP* in *NP*', we would need a corpus containing productions such as

VP(ordered) → VB(ordered) NP(meatballs) PP(in)
VP(ordered) → VB(ordered) NP(zabaglione) PP(in)
VP(ordered) → VB(ordered) NP(supplies) PP(in)
VP(ordered) → VB(ordered) NP(rivets) PP(in)
⋮ ⋮ ⋮

for *every possible* direct object of *ordered*. There are several techniques for overcoming

this data sparsity problem; for example, backing off to an unlexicalized grammar (see e.g. Collins, 1997), clustering word senses (see e.g. Lakeland, in preparation), and even consulting the World Wide Web (Keller and Lapata, 2003).

2.3.3 Probabilistic head-driven phrase structure grammars

Head-driven Phrase Structure Grammar (HPSG), the formalism used by the system described in this thesis, is a typed feature structure grammar described in detail by Pollard and Sag (1994). The application of statistical methods to attribute-value grammars in general, and HPSG in particular, is relatively recent. One of the earliest attempts is by Brew (1995), who first describes a procedure for probabilistically augmenting a formalism he refers to as 'HPSG$^-$'—which lacks the vital feature of re-entrancy. A parameter estimation procedure for HPSG$^-$, isomorphic to that for a PCFG, is described. Brew goes on to sketch means by which this procedure might be extended to take account of re-entrancy, but does not expand upon them. Abney (1997) gives a thorough treatment of stochastic attribute-value grammars. He argues that the PCFG-style empirical relative frequency estimates proposed by Brew will not generally produce a maximum likelihood estimate, and instead advocates log-linear modelling using Markov random fields. He proposes using the Improved Iterative Scaling (IIS) algorithm of Della Pietra, Della Pietra, and Lafferty (1995) with Metropolis-Hastings random sampling to choose features and set weights; however, he expresses reservations about the computational efficiency of the random sampling techniques. Johnson, Geman, Canon, Chi, and Riezler (1999) echo these reservations, and describe two computationally tractable alternatives to Abney's feature estimator. They use these estimators to set parameters for a stochastic version of Lexical-Functional Grammar.

Toutanova, Manning, Shieber, Flickinger, and Oepen (2002) describe an implementation synthesizing much of this work: they construct several PCFG models for HPSG, and compare them with log-linear models that use the same sets of features. Their grammars are trained on parsed sentences from the Redwoods treebank (Oepen, Flickinger, Toutanova, and Manning, 2002), which is also used for the work in this thesis. I will make further reference reference to Toutanova et al. (2002) and the Redwoods treebank in discussing the design (Section 4.1) and implementation (Section 7.3) of a probabilistic grammar for Te Kaitito.

Chapter 3

Integrating disambiguation data

> You will have observed that in the life of every scientist there comes a moment when, having grasped some principle, he develops its consequences and broadens its applications or, as people say, he builds a system. At such times his courage and strength increase. He goes over what he knows and finishes acquiring the knowledge that he lacked. He considers every notion from all its aspects, which he brings together and classifies. And if he is unable to establish his own system or even to convince himself that it really exists, at least when he abandons it he is more knowledgeable than he was before he conceived of it, and he salvages from it some truths which had not been known before.
>
> —Count Jan Potocki (1761–1815), *The Manuscript Found in Saragossa* (Potocki, 1995)

3.1 Classifying and prioritizing disambiguation data

3.1.1 A taxonomy of data sources

In a dialogue system, disambiguation data can be gleaned from a wide variety of sources. As an aid to discussion, classification and integration of these disparate sources, I propose the simple taxonomy shown in Table 3.1. In my explanation of this taxonomy I will use, as a running example, the following variant of a classic ambiguous sentence:

(3.1) The fruit flies like a banana.

This might be interpreted as a statement about the aerodynamic properties of some fruit, or as an observation on the dietary preferences of a group of insects.

		representation level		
		syntactic	*semantic*	*dialogue act*
domain	*world*	general corpus statistics	world knowledge axioms	general dialogue structure axioms
	speaker	language model of the user	axioms about user	dialogue axioms about user
	dialogue context	statistics about recent context	context-matching operations	axioms about current dialogue genre

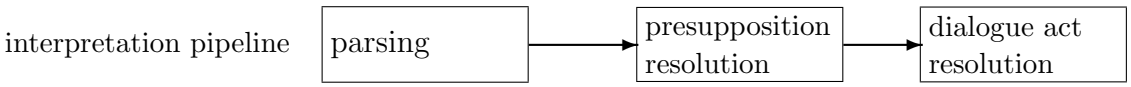

Table 3.1: A taxonomy of disambiguation data sources; the shaded cells correspond to sources not used in the work described by this thesis.

In this taxonomy, sources are classified along two orthogonal dimensions. The first I refer to as **level**. This might be defined roughly as 'semantic depth', and corresponds to the form in which an utterance is represented at each stage in the processing pipeline—for example, a string of characters, or a set of logical predicates. For the purposes of this thesis, I will be using three levels:

Syntactic information sources operate on the parse structure of an utterance. A statistical grammar is probably the most common example. In this case, a lexicalized probabilistic grammar might prefer the *flies-as-noun* reading of Example 3.1, based on a high incidence of the compound noun phrase 'fruit flies' and a low incidence of the verb phrase 'fruit flies' in its training data-set.

Semantic information sources make use of the logical form of an utterance. In the case of the example, a semantic-level disambiguation source would be one which took into account the semantic meaning of 'the fruit' and 'the fruit flies'. A base of world knowledge and an inference engine might be able to show that fruit does not fly much, but flies do tend to like fruit. Te Kaitito does not support much world knowledge or reasoning of this type, but it can make judgements

about semantic interpretations based on the binding of referents within the local dialogue context (see Section 2.1.2). In this case, local semantic disambiguation would prefer *flies-as-verb* if there were some fruit under discussion to which 'the fruit' could refer, but no flies to which 'the fruit flies' could refer.

Dialogue act information sources make use of knowledge about how dialogues tend to be structured, and about the relationships between utterances considered as dialogue acts. These might take the form of axioms about the discourse relations which are likely to hold between an incoming utterance and its immediate context, similar to those presented by Lascarides and Asher (1991). Suppose that Example 3.1 was immediately preceded in the dialogue by the utterance 'What do the fruit flies like?'. Combining the knowledge that the previous dialogue act was a question with the knowledge that, in normal dialogue, the response to a question is usually a pertinent answer, we could derive a preference for the *flies-as-noun* reading.

Further levels could be envisaged in other situations: for example, a system allowing speech input might also need to consider a phonetic level below the syntactic level. This thesis will not deal with the phonetic level; however, Chapter 6 describes the addition of a character-string level at the start of the pipeline. Note that an information source at a particular level always makes use of information at the lower levels as well—for example, a semantic-level representation is always derived from a particular syntactic-level representation. Thus the classification might more accurately be said to reflect the *highest* representational level considered.

A similar classification can be applied to the level at which ambiguity actually arises when interpreting a particular utterance. That is, we might say that a syntactic ambiguity is one generated while parsing a sentence, and a semantic ambiguity is one generated when semantically interpreting a particular parse of a sentence. This thesis is mostly concerned with resolution of ambiguity *arising* at the syntactic level. However, a central consideration of this work is that to resolve such ambiguities, we can *make use of* information from other stages of the interpretation pipeline—that is, at semantic and dialogue act levels. Developing an architecture which supports this kind of look-ahead is a key goal of this thesis. Thus, 'representational level' will usually refer to the level from which disambiguation data is being inferred, rather than the level at which an ambiguity arises.

The second dimension of classification might be termed **domain**. This refers, roughly, to a body or area of knowledge which can guide the system in disambiguation. In this thesis, I will be making reference to three domains:

World refers to general world knowledge. At a syntactic level, this might take the form of statistics inferred from a treebank of hand-parsed sentences, showing that *flies* is more commonly a verb than a noun. At a semantic level, we might have a knowledge base containing an axiom (in the style of Hobbs et al., 1993) about what flies tend to eat. At the dialogue act level, the *world* domain would encompass general dialogue axioms such as those described by Lascarides and Asher (1991).

Speaker knowledge is concerned with the particular speaker whose utterance is being disambiguated. In the example, we may know that the speaker is a geneticist working with fruit flies. At a syntactic level, a corpus of their utterances might thus show a statistical preponderance of *fly-as-noun* occurrences, providing useful information about the likely part of speech of *fly*. At a semantic level, we might know or be able to deduce that the speaker is more likely to speak about topics related to flies. At the dialogue act level, we might have axioms specific to certain speakers—for example, 'Anne changes the subject a lot' or 'Michael seldom answers a question directly'.

Dialogue context refers to knowledge about the situation in which the utterance was made. Suppose that the conversation up to this point has concerned fruit flies. Syntactically, a corpus of recent utterances drawn from the dialogue itself would show a strong tendency for the *fly-as-noun* reading. Semantically, 'the fruit flies' might be interpretable as a reference to some previously mentioned fruit flies, whereas 'the fruit' would not be resolvable to any previously mentioned fruit. At the dialogue act level we could have axioms about the specific genre of the current discourse. Compare, for example, an extended narration with a task-oriented dialogue; for the narration, we might expect a higher proportion of the speaker's questions to be rhetorical.

As with levels, further domains are possible—if a dialogue's topic is predefined, a subject-specific corpus could be consulted for more specialized information; if the system's users are known to be from a particular region, a corpus of utterances in the regional dialect could be used.

3.1.2 Prioritizing data sources

Merely classifying data sources does not tell us how to combine them. When all the sources are in agreement, there is no problem: we disambiguate according to the consensus. But what are we to do if, say, we are talking to a geneticist about a food fight, and the contextual-level and speaker-level sources disagree? Or if *fly-as-verb* is common in the recent context, but nobody has mentioned any fruit to which 'the fruit' could refer?

Prioritization with respect to level has an important effect on implementation efficiency: since the construction of a representation at each level is dependent on data from the previous level, it is more efficient to stop considering a candidate interpretation at a low level. The question is how much of the interpretation tree shown in Figure 1.2 on page 4 we must construct before we can begin to discard unlikely interpretations. If syntactic-level information always overrides semantic, we can prune most of the branches of the ambiguity tree before they begin to grow.

I will now attempt to answer these questions. As the basis for the work in this thesis, I propose the following prioritization rules:

1. 'Higher' levels should take precedence over 'lower' levels. Specifically, dialogue act information should take precedence over semantic information, which should take precedence over syntactic information.

2. Narrower domains should take precedence over broader domains. Specifically, contextual knowledge should take precedence over speaker-specific knowledge, which should take precedence over global knowledge.

I will now attempt to justify the adoption of these rules.

Level prioritization

I claim that higher levels should be given precedence. This intuition seems to be supported by an archetypal mechanism found in puns, which creates a semantic context to force a syntactically unusual interpretation of a common phrase. For example:

(3.2) A ghost walks into a bar. 'Sorry,' says the barman, 'we don't serve spirits'.

Here, the listener must disambiguate between two readings of the phrase 'serve spirits': either (1) *dispense strong alcohol* or (2) *cater to wraiths*. The conflict between syntactic and semantic levels of disambiguation could hardly be clearer: any general corpus would

contain a vast preponderance of reading 1, and a lexicalized statistical grammar inferred from a general treebank could hardly fail to choose this reading. A human listener will probably have heard the phrase 'serve spirits' in sense 1 a large number of times, and in sense 2 not at all.[1] Within the joke itself, the semantic support for reading 2 is remarkably slender: the single word *ghost*. But the application of semantic reasoning to equate *ghost* with *spirit* is nevertheless able, in a human listener, to override the vast syntactic-level preference for reading 1.

Dialogue act information is also capable of exerting an extremely strong influence. Consider the sentence

(3.3) Matt cooks lunch.

which, outside any dialogue context, seems entirely unambiguous. However, a vastly implausible interpretation can be primed by manipulating the utterance's classification as a dialogue act:

(3.4) A: What do matt cooks do when they get hungry around midday?
B: Matt cooks lunch.

The first utterance is a question; in a normally structured dialogue, this creates an expectation that the second utterance will form an answer to this question. Thus the reading is changed from *noun-verb-noun* to *adjective-noun-verb*. Note that the utterance's classification as a dialogue act (specifically, as an answer to a question) overrides both syntactic information which might be inferred by a probabilistic grammar (for example, the high probability of *lunch* being a noun and *cook* being a verb) and global semantic information (for example, the fact that cooks are people and people are not usually matt).

Domain prioritization

Narrower, more context-specific domains seem to take higher precedence than more global domains in human disambiguation. If Example 3.1 were uttered by someone we know to be studying the aerodynamic properties of foodstuffs, we might well be predisposed to assume the *flies-as-verb* reading, even if it contradicts more general world-knowledge about the eating habits of flies and the rarity of flying fruit. This preference might again be overridden by data from the immediately preceding context:

[1] Unless, of course, the listener has heard the joke before—which, given its vintage, is far from impossible. I ignore this possibility here.

if the last ten utterances have all concerned fruit flies, a human would tend to prefer the *flies-as-noun* reading, regardless of any broader trend in the speaker's utterances.

Examples 3.2 and 3.4 can also be seen to support this kind of domain prioritization: in both cases, the unusual interpretation of the utterance is cued by features in the immediately local domain, which override the global information that would normally lead to the 'standard' interpretation.

3.2 An architecture for the integration of disambiguation data

We can now make use of these rules in an attempt to outline a procedure for disambiguation. Unfortunately, they are not quite sufficient. I have not attempted to formulate a prioritization rule which takes into account a data source's position on both axes (level and domain) simultaneously. What should we do if, for example, very local syntactic knowledge conflicts with global dialogue act knowledge?

Fortunately, in this case, a reasonable disambiguation procedure can be produced without fully resolving this question. As shown in Table 3.1, I will not be considering the full space of disambiguation data sources. Dialogue act knowledge is only used in the global domain, and semantic knowledge only in the local domain. The only difficulty is in the conflict between contextual syntactic information and global dialogue-act information.

In this case I have made a pragmatic decision to prioritize the dialogue-act information. This is in part due to the implementation of syntactic disambiguation: as will be described in Section 4.1.2, syntactic world and syntactic contextual sources are evaluated in a way which makes it hard to consider them separately. In any case, the framework (as described below) allows for some leeway in the judgement of each disambiguation module, so it is hoped that this will suffice.

3.2.1 The inverted pruning framework

The algorithm I propose for disambiguating an utterance can be described as follows. We first generate a complete set of possible **interpretations**. I use the term 'interpretation' to refer to a complete, unambiguous reading of a sentence, encompassing a syntactic parse structure, a set of bindings of referents to the dialogue context, and a determination of the dialogue act represented by the utterance. These interpretations

need not, of course, be realized separately in the implementation: for example, different dialogue-act interpretations which share a single semantic interpretation can all contain pointers to the same semantic-level representation. However, it can be convenient to discuss interpretations as if they were distinct structures.

These interpretations pass through a series of **filter modules**, ordered more or less by descending level; these modules make up a disambiguation pipeline with a structure corresponding to the initial interpretation pipeline, but running in the opposite direction. Each cell in Table 3.1 on page 22 corresponds to one or more of these modules. The individual modules will be described in detail in Chapter 4. Each module scores the parses according to some metric appropriate to the level, throws out those which fail to meet a certain **threshold**, and passes the rest on down the disambiguation pipeline. The threshold is simply a cut-off point expressed in the metric used by a particular filter module; it might be fixed, or dynamically calculated from the distribution of the scores of a particular utterance. The interpretations are first assessed as dialogue acts, and all those which pass this filter are then assessed according their semantic attributes. Those which are sufficiently plausible at a semantic level are then assessed as syntactic structures. Higher representational levels are thus accorded precedence because they have the chance to weed out implausible parses before they even become visible to the lower levels. In effect, the lower levels act as tie-breakers for the higher levels. I refer to this procedure as **inverted pruning** because the filter modules are applied in the reverse order to that in which the interpretations are constructed.

3.2.2 An example of inverted pruning

Consider the following (rather contrived) dialogue fragment, designed to create a context that cues a particular interpretation of the final utterance:

(3.5)
A: The big fruit flies ate the fresh fruit
B: OK
A: I threw the the mouldy fruit at the small fruit flies
B: How does the fruit fly?
A: The fruit flies like a banana

The last utterance is ambiguous at three levels: firstly, it can be parsed as a sentence about fruit or a sentence about flies. Secondly, each of the parses has two possibilities for semantic attachment: fresh versus mouldy fruit in one case, big versus small flies in the other. Thirdly, each semantic-level representation could be interpreted as either

an answer to the question asked by B, or a new assertion, apropos of nothing—leaving B's question unanswered. I will now show how the inverted pruning algorithm deals with these ambiguities.

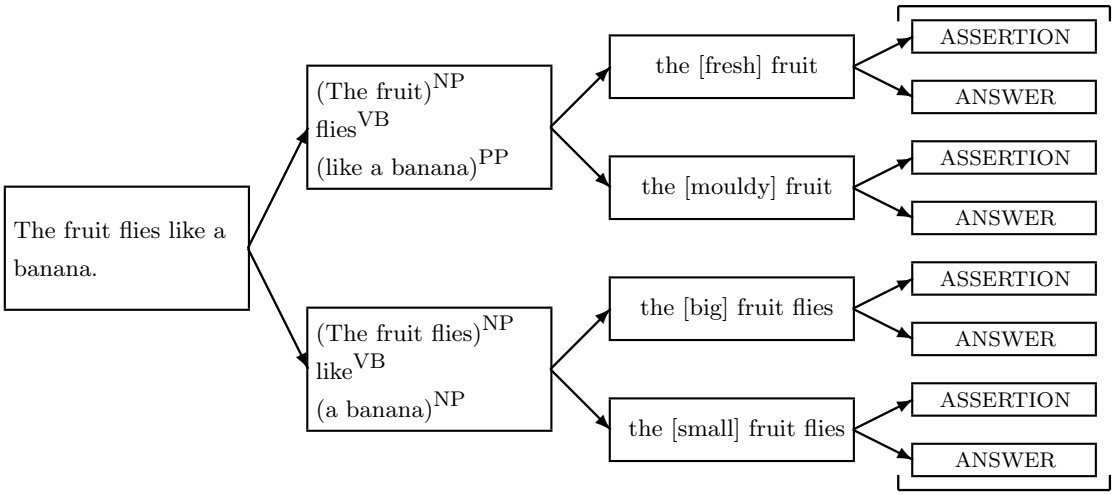

Figure 3.1: A complete set of interpretations. The brackets mark the dialogue act interpretations, which are the first to be considered.

Figure 3.1 shows the fully ambiguous interpretation tree which might be generated from this utterance (cf. the initial state of Menzel and Schröder's eliminative parser shown in Figure 2.3). First, the dialogue acts of the full set of complete interpretations are considered. Since the utterance has occurred in response to the question 'How does the fruit fly?', the dialogue act module chooses all interpretations which form an answer to this question, and discards the rest. Note that by discarding *all* possible speech act interpretations of both possible semantic interpretations of the second parse, we have eliminated this parse without even considering its syntactic merits. This is an example of the 'look-ahead' approach to disambiguation.

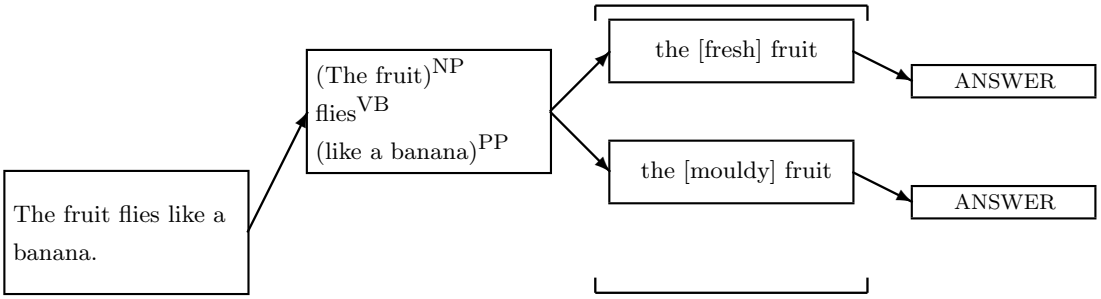

Figure 3.2: Interpretations remaining after dialogue-act filtering. Semantic-level interpretations are now being evaluated.

The two remaining interpretations are passed on to a semantic disambiguation module (see Figure 3.2). There are two *fruit*s in the current dialogue context, and the disambiguation module must decide which of them is being referred to. It picks the mouldy fruit, because it has been mentioned more recently than the fresh fruit (see Section 4.2.3 for details on how Te Kaitito does this in practice).

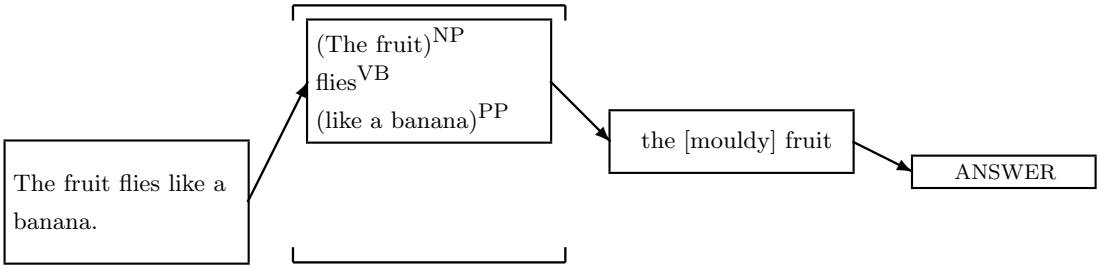

Figure 3.3: The single interpretation remaining after semantic filtering. Syntactic-level disambiguation data could now be consulted, but in this case the utterance has already been fully disambiguated.

The remaining interpretation is passed to a syntactic disambiguation module (see Figure 3.3), which can do nothing but pass it on, since it is the only one left. Note that the syntactic structure has been disambiguated without any reference to syntactic data. Even if there is a huge syntactic preference for the *flies-as-noun* reading, it makes no difference in this case: the syntactic disambiguation module is never consulted.

Inverted pruning involves consulting data sources strictly in isolation rather than attempting to consider them all simultaneously by some technique such as weighted linear combination. There are a number of advantages to this sequential approach. It avoids the difficulty of trying to mathematically combine numerical scores from widely differing sources. For example, the accommodation counter (see Section 4.2.1) will produce integral scores ranging between zero and around five. The stochastic grammar (described in Section 4.1) produces negative real numbers, usually in the range of around -5 to -40, but often with very small variations between different parses of the same sentence. A mathematical formula for combining interpretation scores would need somehow to map these scores into a common domain before they could be compared or combined. In contrast, inverted pruning only requires independent tuning of the threshold value for each module: the numerical scores for interpretations do not interact at all.

This modularity confers other advantages: modules can easily be reordered, added, and removed, without having to make any changes to the combination process. And changes to the scoring system within a module do not require re-tuning of any combi-

nation parameters.

3.2.3 The interpretation-disambiguation-clarification pipeline

If the disambiguation process is successful, a single interpretation will emerge from the end of the pipeline. Modules explicitly avoid throwing out all possible interpretations, regardless of their threshold, so we are assured of having at least one interpretation left. However, we cannot guarantee *only* one interpretation. It could be that none of the modules had sufficient information to whittle the available interpretations down to one. In this case, a dialogue system has an advantage over non-interactive language interpreters: it can explicitly ask the user for clarification.

Clarification questions are discussed in Chapter 5. They certainly count as sources of disambiguation data, and can be classified along the 'level' axis of the described taxonomy. (It is difficult, however, to give them a meaningful classification along the 'domain' axis.) However they cannot be integrated into the disambiguation pipeline in the ordinary way: firstly, they are a technique of last resort—obviously, if it is possible to disambiguate without inconveniencing the user, we would prefer to do so. Secondly, they must be asked in order of *increasing* semantic depth, whereas data sources for automatic disambiguation should be consulted in *decreasing* order. These considerations will be more fully explained and justified in Chapter 5.

The complete interpretation process—consisting of initial interpretation, automatic disambiguation, and (if necessary) explicit clarification, can thus be seen to have the structure of a *boustrophedon*[2]—that is, progressing from left to right (interpretation), then right to left (disambiguation), then left to right again (clarification). Figure 3.4 gives a diagrammatic representation of this structure. Note that the bottom two rows of this figure (automatic disambiguation and explicit clarification) correspond in their entirety to a *single* module in Te Kaitito's overall pipeline as shown in Figure 2.1.

3.2.4 The consequences of full look-ahead for disambiguation

In terms of implementation efficiency, this scheme might seem a little disappointing: we must follow every branch of the pipeline right to the end before we can decide whether it constitutes a valid interpretation.

[2]The term *boustrophedon*, referring literally to the route followed by an ox ploughing a field, is more commonly applied to writing systems where lines are written alternately left-to-right and right-to-left.

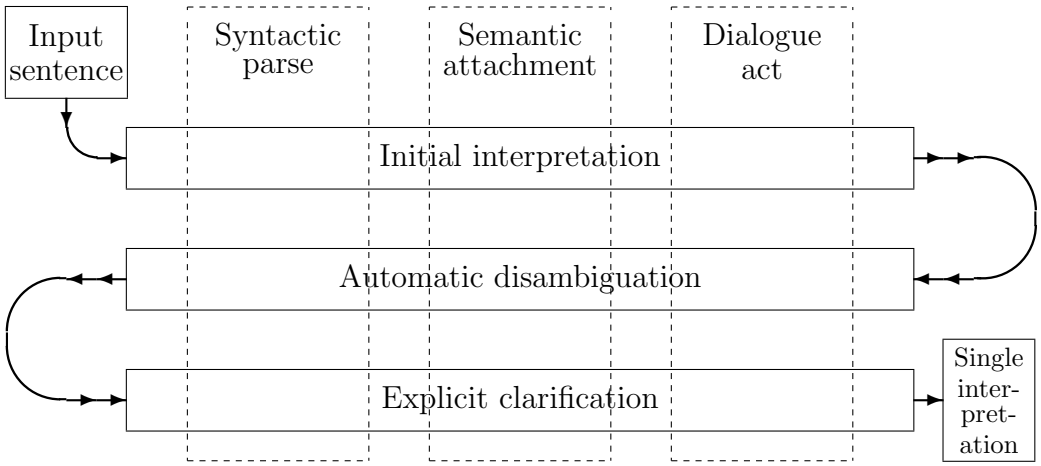

Figure 3.4: The boustrophedal structure of the interpretation process

There are also advantages, however; for example, the availability of semantic information for every syntactic structure allows for the possibility of augmenting clarification questions with contextual information, as described in Section 5.4.1.

If efficiency proves to be a problem, we can still apply the data sources in a more conventional order by pruning *some* interpretations during the initial process of constructing interpretations. For instance, if we are using a hugely ambiguous grammar which returns a thousand parses for a sentence, we might rank them them as soon as they are generated and only pass, say, the top hundred on through the interpretation pipeline. In this way we can improve efficiency without sacrificing all the advantages of look-ahead disambiguation, although we might get incorrect interpretations of a few pathological cases. In the system I am working on, this level of ambiguity is not manifested at any level, so this kind of pre-emptive pruning is not carried out.[3]

3.2.5 Comparison to other approaches

This thesis takes an approach somewhat different from that of Hobbs et al.: information sources are procedurally prioritized rather than contributing to a global numerical score, and there is no large base of 'common sense' world knowledge. However, there are also points of strong similarity: in particular, the accommodation of unknown referents

[3]In fact, the version of the Redwoods treebank used in the implementation of my probabilistic grammar *does* exhibit large-scale ambiguity for a small proportion of sentences; however, the latest version of Te Kaitito uses a newer and less ambiguous version of the ERG, and the MEG is not overly ambiguous.

(described in Section 4.2.1) and the accommodation of errors by input perturbation (described in Chapter 6) are similar to the cost-weighted assumptions made during abduction.

In terms of the framework described by Menzel and Schröder (1999), the disambiguation procedure proposed here can be seen as an application of constraints, but in my scheme they are applied sequentially rather than simultaneously.

Note that both Hobbs et al. and Menzel and Schröder speak in terms of 'costs' or 'weights' assigned to the making of assumptions or the retraction of constraints, but they give few details as to how these weights should be set. Menzel and Schröder appeal to the intuitive notion of 'how serious one considers a constraint violation', and write that their parser seeks an interpretation which violates 'as few and as weak constraints as possible'. This raises the question of what to do when faced with the choice of violating as few constraints as possible *or* as weak constraints as possible. Is it better to violate two weak constraints or one strong one? And how are the weights to be combined when multiple constraints are considered?

Hobbs et al. specify that their costs are combined additively, but similar questions arise: how can one be sure, when choosing costs, that, say, an interpretation requiring a \$10 assumption and a \$4 assumption will *always* be preferable to one requiring three \$5 assumptions? Addressing such concerns, Hobbs et al. write:

> The problem of how to combine symbolic and numeric schemes in the most effective way, exploiting the expressive power of the first and the evaluative power of the second, is one of the most significant problems that faces researchers in artificial intelligence today. The abduction scheme we have presented attempts just this. However, our numeric component is highly *ad hoc* at the present time.

They go on to discuss various approaches to assigning and combining numerical costs; however, they do not give a concrete recommendation for a principle on which to base these assignments or combinations.

It seems that any implementation of simultaneous multi-level disambiguation using numerical weights must necessarily be somewhat ad hoc. A search strategy for finding acceptable weights is clearly needed, but the search space is large and complex: if we have a set of n weighted constraints (or assumable axioms), and if any subset of those might occur in a sentence, then changing the value of just a single weight has the potential to alter the interpretations of 2^{n-1} sentences.

It was in order to avoid this degree of complexity that I designed a more tightly constrained, more procedural algorithm. Certainly there are still parameters which

must be tuned empirically—the exact implementations of the ranking metrics, and the threshold values for keeping or pruning interpretations—but these are local to each module, and do not result in a combinatorial explosion of potential interactions. Implementation thus becomes far more straightforward. This strategy might be seen as more dangerous than the simultaneous approaches of Hobbs et al. and Menzel and Schröder: disambiguation data can be thrown away without ever being consulted—for example, the syntactic data of the example given in Section 3.2.2—so perhaps there is a risk of missing a valid interpretation. However, the prioritization rules proposed in Section 3.1.2 are designed to minimize the risk of this happening.

There is one area in which I do use linearly weighted rather than procedural combination techniques: this is *within* the syntactic disambiguation module, in integrating the world, speaker, and context domains. Further details of this will be given in Sections 4.1.2 and 7.3.2.

Chapter 4

Disambiguation modules

So he went the words to gather,
That the spells he might discover,
And a field he spread with reindeer,
Loaded benches high with squirrels,
Many words he thus discovered,
But they all were useless to him.

—*Kalevala*, Runo XVI, ll. 141-146 (Lönnrot, 1907)

Having established a framework for the classification of disambiguation data, I will now outline some actual data sources around which the filter modules described in Section 3.2.1 can be constructed. The modules are all described with a view to practical implementation within Te Kaitito, so the level of description is fairly concrete (although the implementation details are deferred to Chapter 7).

The descriptions of the filter modules are arranged with respect to modules' *levels*, as defined on page 22. The focus is on the highest representation level from which the modules draw their information, rather than the level at which the information is applied. Thus, for example, presuppositional weight is described under 'semantic modules' although it is only used to disambiguate between syntactic parses.

First, the only syntactic-level module—the probabilistic grammar—will be described (Section 4.1). Next I will discuss two semantic-level modules based on presupposition-resolution techniques (Section 4.2), and a saliency-based module for disambiguation of referents (Section 4.2.3). Finally, in Section 4.3, I will present a somewhat miscellaneous collection of modules operating at the dialogue act level.

4.1 The syntactic-level filter module

The only syntactic-level module presented here is a stochastic parse ranker using the probabilistic parsing techniques discussed in Section 2.3. Here I will mainly discuss the innovations developed for this thesis; their implementation will be described in more detail in Section 7.3.

4.1.1 Integrating a stochastic HPSG grammar with Te Kaitito

Te Kaitito, as mentioned in Section 2.1.1, uses an HPSG-style grammar which operates by unification rather than the simple rewriting rules of a CFG. It is therefore not possible to apply probabilities directly to productions in the same way. However, the LKB parser does produce a **derivation tree** for each possible parse, which can be treated in more or less the same way as the parse tree of a CFG. The derivation tree is essentially a tree-structured record of the rule applications which result in a successful parse of the sentence. Brew (1995) shows that, with very little modification, PCFG-style training and parse ranking can be applied to HPSG derivation trees.

As described in Section 2.3.3, Abney (1997) shows that the context-dependency of attribute-value grammars such as HPSG makes empirical relative frequency estimates[1] (as used for PCFGs) sub-optimal for training. Nevertheless, they give adequate performance in many cases[2] and their simplicity makes them attractive. I have two main reasons for choosing to use empirical relative frequency estimates. Firstly, the modifications made to improve the grammar's sensitivity to dialogue context (described in the next section) make it desirable to start with a simple probability model. Secondly, the time-consuming process of implementing and testing a sophisticated log-linear model would conflict with the more immediate goal of integrating the grammar both with the rest of the disambiguation pipeline and with the Te Kaitito system as a whole.

In this thesis, then, the main concern is not to build the best possible probabilistic grammar; it is to integrate a probabilistic grammar thoroughly with all the components of a dialogue system. This integration takes place in two complementary ways: the use of dialogue context to improve the performance of the probabilistic grammar in novel ways (as described in Section 4.1.2); and the use of the probabilistic grammar to

[1] Abney coins the term 'empirical relative frequency' to describe the probability estimates calculated for a standard PCFG.

[2] Toutanova et al. (2002) and Toutanova, Manning, Oepen, and Flickinger (2003) give comparisons of PCFGs with log-linear models for HPSG.

improve the performance of a number of other dialogue system modules. The primary motivation and purpose of the probabilistic grammar is certainly disambiguation of incoming utterances. However, once in place, it is readily available for other purposes. For example, Te Kaitito creates its utterances as MRS forms which are turned into text strings by LKB's generation module. LKB often generates more than one syntactic form (phrasing) of the supplied semantic content. Prior to my implementation of the probabilistic grammar, Te Kaitito arbitrarily picked one of these phrasings, leading to utterances which, while syntactically valid, can be rather oddly formulated (for example, 'The tree in the garden the dog chases the cat up.' rather than 'The dog chases the cat up the tree in the garden.'). This provides an obvious application for a probabilistic grammar: it can be used to select the most 'natural' of a set of generated utterances. The grammar can also be used to select, in similar fashion, the most natural rephrasings during a clarification subdialogue (see Section 5.3.3).

Ranking generated utterances stochastically also allows us to augment the grammar with special forms for use only during clarification, without the danger that they will be used in normal dialogue; Section 5.3.7 describes this technique.

More interestingly, a stochastic grammar incorporating the innovations described below can also be used to monitor the progress of a language learner (see Section 4.1.4).

4.1.2 Contextually augmented probabilistic parsing

In a traditional probabilistic grammar, rule probabilities are static: they are inferred offline from a treebank and do not change during the parsing process. This limits the sensitivity which a probabilistic grammar can have to local context. One way of addressing this is is to use features which incorporate information about the recent context, but this technique is limited by two factors: the computational tractability of using increasingly complex features, and the availability of training data. The more complex a feature, the less well-attested it will be in a corpus, requiring heavier use of smoothing and backoff.

I have explored a simple but novel way of making a statistical grammar more sensitive to context: in addition to maintaining the rule counts necessary for a traditional probabilistic grammar (henceforth referred to as **global** parameters) we also maintain a set of **local** parameters.

During the course of the dialogue, uses of grammar rules in utterances interpreted by the system are counted by the local grammar in much the same way as the global grammar is trained on a treebank beforehand. In effect, the unfolding dialogue acts as

a dynamic ancillary treebank which can be used to adjust the probabilistic grammar's parameters. The local grammar is intended to reflect the current dialogue context, so damping is applied at every dialogue turn to pull the probabilities back towards those given by the global parameters. Note that, as with a treebank, we can always be fairly certain of having a correct parse since we can fall back on clarification questions (described in Chapter 5) if disambiguation fails.

4.1.3 Advantages of contextually augmented probabilistic parsing

A contextually-augmented parsing scheme offers several advantages over a traditional 'static' probabilistic grammar. Disambiguation should become more reliable: sentences are disambiguated with more reference to the features seen in the recent dialogue, rather than just those in a corpus which might reflect usage in a somewhat different domain. For example, my global probabilistic grammar is trained on the Redwoods treebank (introduced in Section 2.3.3 and further described in Section 7.3). Redwoods chiefly covers the domain of business appointment scheduling, which is not a major concern in most of Te Kaitito's dialogues.

Data sparsity is an endemic problem in statistical parsing: no matter how large the corpus, there will be cases it doesn't cover. Smoothing and backoff are usually used to compensate for a lack of data, but a bias towards the rules represented in the local context also helps to overcome this problem: the utterance being disambiguated is likely to be closely related to recent utterances, so a grammar reflecting them is likely to have good coverage of the current utterance.

The locally observed features are not used in isolation: they are only used to modify temporarily the fixed parameters of the global grammar. Thus, if the global grammar has poor coverage in some area, the local grammar might help; and if the local grammar doesn't contain any pertinent data for a particular utterance, the performance will at least be no worse than for the original, global grammar.

It has been observed (e.g. by Wang and Hoffman, 2004) that the first occurrence of an ambiguous term in a body of text generally provides the best context for determining the sense of an ambiguous word. Disambiguation algorithms often take advantage of this by taking into account the sense of the first occurrence when disambiguating subsequent occurrences of the same word, or even by disambiguating *only* the first occurrence and assuming that all subsequent occurrences have the same sense (see

e.g. Mohammad and Pedersen, 2004). However, there is a risk associated with this technique: if the first occurrence is disambiguated incorrectly, there is little or no hope of rectifying it on the basis of subsequent occurrences.

The contextually augmented grammar I propose produces a similar effect. Once an ambiguous construction has been successfully disambiguated once, the grammar's parameters adapt to increase the probability of resolving the ambiguity in the same way if it occurs again. But in this case the risk of propagating an incorrect disambiguation is greatly reduced: in an interactive dialogue system, we can always fall back on explicit clarification (as described in Chapter 5) if disambiguation produces insufficiently conclusive results. Thus we can be more confident of performing the first disambiguation correctly—although we may require help from the user to do so.

4.1.4 Idiolect-sensitive parsing

In a very similar fashion to context-sensitive parsing, we can augment the probabilistic grammar to be sensitive to a particular user's idiolect. In this case we maintain one ancillary database for the utterances of each individual user, train it only on that user's utterances, and consult it (in conjunction with the global and local databases) only when that user is the speaker. Again, some damping is necessary: user idiolects are susceptible to change, especially in a CALL environment where learners' language use may be expected to become more sophisticated with time.

Maintaining statistics on the user's idiolect can be useful for other purposes: in a CALL system, it is important to monitor the learner's language use for gaps in linguistic knowledge. A per-user treebank could allow this to be done by more elegant means than explicit testing: the statistical database can be consulted to see if any grammatical rules have incidences significantly below those in the global database. When more structured, graded dialogues are added to Te Kaitito, this could allow the system to judge when a learner has sufficient understanding for it to end the dialogue.

The statistics could also be used to rectify gaps in the user's knowledge: when the system generates a response, the generator often produces more than one syntactic form. The probabilistic grammar can be used to select the most 'natural' of these; however, in teaching mode, more unusual use can be made of it. The generated derivation trees can be ranked with a heavier bias towards the idiolect database, and the *lowest*-scoring parse presented to the user. In this way, the system could gear its style of expression in such a way that the user sees more examples of grammatical constructions with which they are unfamiliar.

Work along these lines is currently being conducted by Slabbers (in preparation): the probabilistic parsing code I developed for this thesis is being used in the development of a teaching system which assesses a student user's knowledge of syntactic rules.

4.1.5 Inverted scoring

The Redwoods treebank data do not just include correct parse structures for sentences; they include every possible parse, along with data showing which parses are preferred by the annotators.

The availability of so many incorrect parses raises an interesting possibility: could they be used to train a probabilistic grammar to recognize *incorrect* interpretations of a sentence? The training process would be identical; only the input data would differ. Using this inverted parse ranker in conjunction with a more conventional probabilistic grammar might improve performance; after all, the incorrect parses in the treebank contain information not available to the normal type of probabilistic grammar.

4.2 Semantic-level filter modules

Te Kaitito's main semantic-level disambiguation facilities use the mechanisms of presupposition resolution and accommodation (operating in the contextual domain of the taxonomy described in Section 3.1.1). The features described and implemented here were first described in Knott and Vlugter (2003).

The principle behind Te Kaitito's semantic disambiguation is that the most likely interpretation is the one which can be most easily incorporated into the dialogue context.

4.2.1 Accommodation

If Te Kaitito is unable to bind a presupposition to the current dialogue context, then—as described in Section 2.1.2—it will **accommodate** the presupposition: that is, it will add the relation to the current dialogue context as if it had been explicitly stated. Thus, for example, if no dogs have previously been mentioned, *the dog barks* produces exactly the same effect as *a dog barks*: a dog relation and a barking relation are added to the current context. (If a dog *has* been previously mentioned, the definite form will

attach the barking relation to it, whereas the indefinite form will instantiate a new dog).

This process can be used as a disambiguation tool: we assume that the user will, in general, not casually refer back to non-existent entities; thus the most likely interpretation is one which requires the smallest number of accommodations. For instance, given the following classic example of prepositional attachment ambiguity

(4.1)
User	a man with a telescope arrives
System	okay
User	a girl appears
System	okay
User	the man sees the girl with the telescope
System	okay

the accommodation module would choose the interpretation where *with the telescope* attaches to *sees* rather than *the girl*. Since the system has not been told about a girl with a telescope, the *girl-with-telescope* relation would have to be accommodated, whereas the other interpretation can be incorporated into the dialogue context with no accommodation. As in Hobbs et al. (1993), we assume that the correct reading is 'the minimal explanation of why the text would be true'—that is, the one requiring the fewest extra assumptions.

4.2.2 Presuppositional weight

Presuppositional resolution can be used for disambiguation even when no accommodation takes place, using **presuppositional weight**. This is defined as the number of presuppositions in the user's utterance which can be matched to the dialogue context. The assumption is that the integration of a sentence into the context is more 'complete' when a larger number of presuppositions match, and that in this case there is a smaller probability that the interpretation matches the context purely by chance. Consider the following example:

(4.2)
User	a man arrives
System	okay
User	a girl with a telescope appears
System	okay
User	The man sees the girl with the telescope

In this case, no presuppositions need be accommodated for either interpretation. But the attachment of *with the telescope* to the noun requires three presuppositional relations to be bound to the context (*the man*, *the girl*, and *girl-with-telescope*). The other reading only contains the first two of these presuppositions. Therefore, the presuppositional weight filter module prefers the former reading.

4.2.3 Saliency

Te Kaitito maintains a **saliency list** of relations introduced during a dialogue. The saliency of a relation is inversely proportional to the number of discourse turns which have elapsed since it was last mentioned. Since an utterance is more likely to refer to a recently mentioned entity than to one which was mentioned a long time ago, this is a valuable source for semantic disambiguation.

The current implementation is based on saliency *windows*. Rather than making use of referents' raw positions in the saliency list, the disambiguation module regards two entities as equally salient if they occur within a certain distance (a window) of each other in the saliency list. The size of the window increases with depth in the saliency stack. For example, in this case

(4.3)

User	A black dog walks.
System	OK
User	A white dog walks.
System	OK
User	The dog chases a cat.

'the dog' in the last utterance would be disambiguated to the white dog. However, if twenty irrelevant utterances (not mentioning either dog) were interposed before the last one, the dogs would be further down the saliency list and thus fall within the same window. In this case the module would not attempt to disambiguate automatically, and a semantic clarification question would have to be asked (see Section 5.5).

4.3 Filter modules at the dialogue act level

Finally, we come to filter modules which operate at the dialogue act level—using inferences drawn from the properties of the utterance as a whole, and its relationship to the preceding dialogue, rather than metrics from a particular aspect of it. Modules at this level might be seen as making use of the co-operative principle described by Grice

(1975): we assume that the user is attempting to co-operate with the system, rather than deliberately making utterances which are hard or impossible to interpret.

Sections 4.3.1 and 4.3.2 deal with distinguishing between different dialogue acts (by making use of information at the dialogue-act level). Distinguishing between questions and statements can usually be done at the syntactic level, but Sections 4.3.1 and 4.3.2 subdivide these classifications further. Sections 4.3.3 and 4.3.4 use more explicitly Gricean reasoning to aid disambiguation at lower levels.

As far as I am aware, the techniques introduced in Sections 4.3.2, 4.3.3, and 4.3.4 have not previously been described or implemented.

4.3.1 Distinguishing between answers and assertions

If Te Kaitito has asked a question, and the user's next utterance can be interpreted as an answer to this question, then it will be interpreted as such. Otherwise, it will be interpreted as a new assertion, apropos of nothing. For example, in this case

(4.4) System Which dog sleeps?
User The black dog sleeps.

the user's utterance is assumed to be an answer, whereas in this case

(4.5) System Which dog sleeps?
User The black cat sleeps.

the user is assumed to have ignored the question, and the utterance is classified as a plain assertion.

4.3.2 Distinguishing between questions and clarification questions

If Te Kaitito makes an utterance and the user responds with a question, how can we distinguish between a question which initiates a clarification subdialogue and an 'ordinary' question? One way is to see whether the question could have been answered by the semantic form from which the original utterance was produced.[3]

Almost all Te Kaitito's utterances (excepting a few 'canned' messages for special cases) are generated from a deep semantic structure in the form of MRS. Thus there is an invisible semantic structure associated with the surface structure which the user

[3]This idea is due to Samson de Jager.

sees. This structure can be used to identify clarification questions: if the system is able to answer the user's question using the semantic material in its own preceding utterance, it is probably a clarification question. For example, in this exchange

(4.6) System The cat sleeps.
User Which cat sleeps?

the utterance 'The cat sleeps.' was generated from an MRS form which must have included a *cat*-relation. This relation can only have come from Te Kaitito's dialogue context database, and must refer to a *specific* cat. Thus the semantic form of the original utterance contains the answer to the user's question. Consider, in contrast, the exchange

(4.7) System The cat sleeps.
User Where does the cat sleep?

The original utterance contains no location, so the MRS which generated it cannot have contained a relation specifying where the cat sleeps. Of course, it is possible that this information is contained in Te Kaitito's dialogue context database, but it cannot be extracted from the question itself. The user's utterance is thus classified as an 'ordinary' question rather than a clarification question.

4.3.3 A preference for answerable questions

A participant in a dialogue usually has some mental model of what their interlocutor knows. Following a co-operative principle, they can reasonably be expected only to ask questions which their other participant is able to answer—or, at least, which they believe the other participant is able to answer. For example, consider this dialogue fragment:

(4.8) A: I missed the play, but made it to the party afterwards.
B: How was it?

B knows that A went to the party, but not the play. Furthermore, A knows that B knows this. A is therefore likely to assume that *it* refers to the party rather than the play.

Te Kaitito makes use of the same principles:[4] if there are multiple parses of a question posed by the user, Te Kaitito will ignore those which it cannot answer. Of course, there may still remain multiple questions which the system *can* answer, but these can be passed down the disambiguation pipeline in the usual way.

[4]This idea is due to Peter Vlugter.

4.3.4 Using ambiguity for disambiguation

One of Grice's Maxims of Manner is 'avoid ambiguity'. We can actually use the level of ambiguity itself as an aid to disambiguation: if one parse of an utterance leads to a great deal of ambiguity at a semantic level, we can disprefer it in favour of a parse with little or no semantic ambiguity. The following example illustrates the principle.

Suppose that a customer walks into a shop which sells swimsuits. Arrayed behind the salesperson is a wide variety of swimsuits. Some are black, but most are an identical shade of shocking pink, in a variety of cuts and styles ranging from the extremely modest to one particular, shockingly skimpy garment. The customer says 'Please hand me the shocking pink swimsuit.' There is clearly ambiguity here: does *shocking* modify *pink* or *swimsuit*? If it modifies *pink* then the customer has made a hugely ambiguous request: *all* the pink swimsuits are shocking pink. But only one of the swimsuits is in itself shocking. Thus it is reasonable to assume that *shocking* modifies the noun. In the converse situation—where all the swimsuits are shocking, and several are pink, but only one is shocking pink—*shocking* would probably be assumed to modify the noun.

Chapter 5

Clarification subdialogues

> The stupidity of people comes from having an answer for everything. The wisdom of the novel comes from having a question for everything.
>
> —Milan Kundera, Interview with Philip Roth (Kundera, 1983)

There is no guarantee that the disambiguation process will result in a unique interpretation; even human listeners occasionally have to ask for explicit clarification of an ambiguous utterance. Before this project, Te Kaitito had some rudimentary facilities for explicit clarification. In this chapter, I describe a new, systematic method for generating clarification questions, handling clarification subdialogues with the user, and integrating clarification with the rest of the interpretation and disambiguation pipeline.

Clarification questions, like disambiguation modules, may be classified according to the stage of the interpretation pipeline to which they correspond. At the level of syntactic interpretation, there are syntactic clarification questions which seek to establish the correct parse of an utterance. Beyond these are semantic questions, which clarify the correct bindings for ambiguous referents in the utterance. Finally, there is dialogue act clarification, encompassing such questions as *Are you asking me or telling me?*. Dialogue act clarification is not currently implemented in Te Kaitito and will not be further discussed in this thesis.

I suggest that clarification questions, in contrast to automatic disambiguation steps, must be posed in the *same* order as the original interpretation was constructed: for example, the syntactic structure of the utterance must be clarified before a meaningful question can be asked about the semantic attachment. Every semantic structure is dependent on a syntactic structure; forming the semantic clarification question makes use of the syntactic structure, so the question cannot be constructed if the syntactic structure is still ambiguous.

It would be possible, of course, to construct a semantic question based on each syntactic interpretation, and present them all to the user; this would correspond to simultaneously disambiguating the syntactic and semantic levels. However, it is hard to see why this would be desirable: why go to the trouble of generating every possible semantic question, and put the user to the trouble of reading them all, when it is possible to ask a syntactic-level question first to eliminate a number of unnecessary semantic questions, and then only ask the semantic questions which remain? With reference to an ambiguous interpretation tree such as that shown in Figure 3.1 on page 29, my proposed clarification scheme involves travelling along a single branch of the ambiguous structure, asking the user for guidance at each branch. Attempting simultaneous semantic and syntactic disambiguation would be more akin to enumerating all the nodes at the semantic level.

In this chapter I will give a fairly full treatment of syntactic clarification using rephrasing, with brief discussions of other syntactic techniques and of semantic and dialogue-act techniques. In Section 5.1 I will briefly review some literature on clarification subdialogues. In Section 5.2 I will explain why clarification is important, both in Te Kaitito and in general. Section 5.3 discusses rephrasing-based syntactic clarification in some detail, creating a theoretical framework within which specific clarification techniques are then described and assessed.

The remaining sections of the chapter are somewhat less extensive. Section 5.4 describes techniques for syntactic clarification which do not fit directly into the rephrasing-based model of Section 5.3, although some of them extend it. Finally, Section 5.5 describes semantic clarification.

5.1 Literature on clarification subdialogues

Clarification subdialogues are a well-known and intensely studied phenomenon. They were first described systematically by ethnomethodologists (see e.g. Sacks, Schegloff, and Jefferson, 1974). Grosz and Sidner (1986), although they do not explicitly discuss clarification, do give a general plan-based model for discourse structure which includes subdialogues and can readily incorporate clarification: in Grosz and Sidner's system, clarification subdialogues would be classified as a 'digression' or type-3 interruption, where the authors define an interruption as 'a discourse segment whose DSP [discourse segment purpose] is not dominated nor satisfaction-preceded by the DSP of any preceding segment'. Litman and Allen (1984) discuss clarification subdialogues specifically,

presenting a hierarchical plan recognition approach to modelling them.

5.2 The importance of clarification

The amount of attention paid to clarification in this chapter may seem a little excessive, given that the number of candidate parses—at least for Te Kaitito's current grammars—is usually fairly low after disambiguation has taken place. But it should be borne in mind that clarification questions are the last resort of disambiguation: if clarification fails, the system has no choice but to choose an interpretation at random (or almost at random: see Section 5.3.2). It is therefore worth investing effort in a fairly sophisticated clarification system, even if it will only rarely be used.

The clarification process can also serve as an aid to the probabilistic grammar itself: as described in Section 4.1.2, a dynamic probabilistic grammar trains itself on fully disambiguated utterances, so when a particular type of ambiguity has been clarified once or a few times the grammar parameters can adapt sufficiently to disambiguate it in future without recourse to clarification. The dynamic grammar is also a strong motivation for clarification to be complete and correct every time: if an incorrect parse is randomly selected once, the grammar's parameters are pushed in the direction of this parse and the probability of a subsequent incorrect disambiguation is increased.

A further reason for careful attention to clarification subdialogues is that many of the techniques I discuss have broader applications: rephrasing-based clarification can be used in any interactive system which has a bidirectional grammar and can generate rephrasings from a candidate semantic structure; the more elaborate clarification techniques could even be used in the absence of any other syntactic disambiguation procedure. And, as described in Section 5.4.2, most of the techniques described for rephrasing-based clarification are also applicable to referent-based clarification.

Clarification techniques developed for users of the system can also be of great utility during system development; Section 5.3.11 details one such case.

5.3 Syntactic clarification by rephrasing

One common method of resolving syntactic ambiguity is simply to rephrase the ambiguous utterance into an utterance with the same semantic content but with a different syntactic structure. For example:

(5.1) A: I saw the girl with the telescope.
B: What do you mean?
A: With the telescope, I saw the girl.

It is also common for the confused party to prompt the speaker with alternatives which resolve the ambiguity under consideration:

(5.2) A: I saw the girl with the telescope.
B: Do you mean that it was with the telescope that you saw the girl, or that it was the girl with the telescope whom you saw?
A: It was with the telescope that I saw the girl.

Note that it is not necessary for the rephrasing to be unambiguous, or even for it to be less ambiguous than the original utterance: it must only be unambiguous at those points where the original utterance was ambiguous.

In this section I will give a fairly detailed and theoretically grounded account of rephrasing-based syntactic clarification. In Section 5.3.1 I will describe a theoretical and notational framework for the subsequent sections; Section 5.3.2 discusses the possibility of clarification failure; Sections 5.3.3 and 5.3.4 derive some general results useful in finding redundant rephrasings; and Sections 5.3.5, 5.3.6, and 5.3.7 discuss general implementation techniques.

Section 5.3.8 proposes metrics for the assessment of different clarification *strategies*—that is, techniques for selecting rephrasings, presenting them to users, and processing their responses in some way. Sections 5.3.9–5.3.13 then describe various strategies and discuss their merits.

The results derived and techniques developed for rephrasing-based clarification are quite general in nature and broadly applicable—to other types of clarification and even to domains very different from natural language processing; Section 5.3.14 briefly discusses these possibilities.

5.3.1 Theoretical framework and notational conventions

Let $\mathcal{R}$ be the set of all possible utterances, and $\mathcal{S}$ be the set of all possible semantic (MRS) structures. (We are using the raw MRS structures before presupposition resolution has taken place; thus there is a one-to-one correspondence between these structures and the syntactic structures which we are attempting to disambiguate.) The relation $\rhd$ (pronounced 'generates') may be defined by

$$\forall s \in \mathcal{S} \forall r \in \mathcal{R} : s \rhd r \iff r \text{ can be generated from } s$$

(or, equivalently, iff r can be interpreted as s). I write $\rhd$'s inverse as $\lhd$ (pronounced 'realizes'). I will denote the image of an element e under a relation E by $E[e]$, so

$$\rhd[s] = \{r | s \rhd r\}$$

and

$$\lhd[r] = \{s | r \lhd s\}$$

We can now define the relation $\lhd\rhd$ ('rephrases') by

$$\forall r_1, r_2 \in \mathcal{R} : r_1 \lhd\rhd r_2 \iff \exists s \in \mathcal{S} : s \rhd r_1 \wedge s \rhd r_2$$

$\lhd\rhd$ is clearly reflexive and commutative, from the properties of $\wedge$; thus, as one would expect, an utterance is a (vacuous) rephrasing of itself, and r_1 rephrases r_2 if and only if r_2 rephrases r_1.

Suppose now that the user has presented the system with an ambiguous utterance u. We then define the set S of **interpretations** of u by

$$S = \lhd[u] = \{s | u \lhd s\} \tag{5.3}$$

and the set R of u's **rephrasings** may be defined in terms of S as

$$R \;=\; \{r | \exists s \in S : s \rhd r\} \tag{5.4}$$

The user wishes the system to interpret their utterance as a specific semantic structure $s^* \in S$ but cannot directly select this structure; all they can do is to tell the system, for various $r \in R$, whether $s^* \rhd r$. That is, the user can specify s^*'s image under $\rhd$.

As an example, consider the sentence

(5.5) The sheep saw the fish in the river.

With a reasonable grammar, this sentence might be expected to have eight possible parses, corresponding to all possible combinations of three independent features: the number of *sheep*; the number of *fish*; and the high or low attachment of *in the river*, corresponding respectively to the seeing occurring in the river or the fish being in the

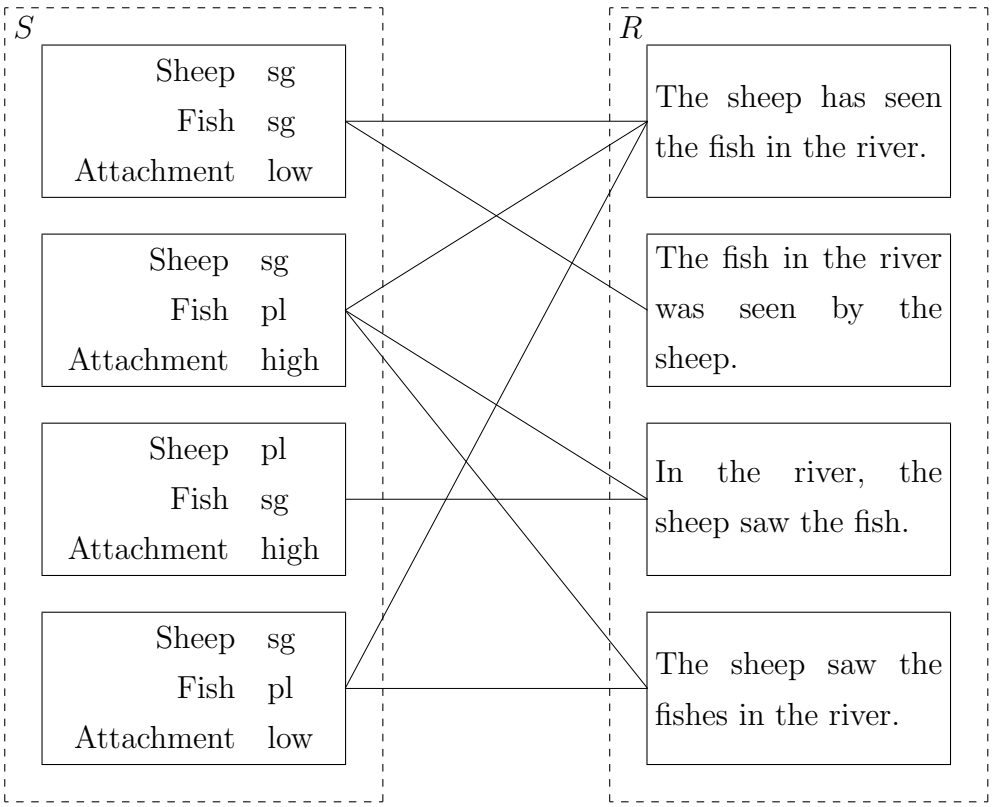

Figure 5.1: A relation between semantic structures and rephrasings. The parses and rephrasings are subsets of those which might be generated from Example 5.3.1.

river. Figure 5.1 illustrates the relation $\triangleright$, relating four of these parses to rephrasings which could be generated from them.

Unfortunately, there is no guarantee that information about rephrasings in R will be sufficient to fully specify s^*—it is possible that a different semantic structure has exactly the same image. So without recourse to more direct information about the set S, even an 'ideal' clarification process (making full use of all the information it can extract from the user) can only produce a set of possible structures

$$S' = \{s | s \in S \wedge (\forall r \in R : s \triangleright r \iff s^* \triangleright r)\}$$

—that is, all the semantic structures which generate exactly the same set of rephrasings as s^*. In practice, a clarification strategy may not make full use of all the information available from the user (either because it does not ask for all the information, or because it processes the supplied information wastefully), producing a larger set of

possible parses.

The process of **rephrasing-based syntactic clarification** consists of asking the user, for various $r \in R$, whether $s^* \rhd r$, or, equivalently, whether $s^* \in \lhd[r]$. Let $R^* \subseteq R$ be the set of utterances we are asking the user about. We can view the user's clarification as supplying us with a binary-valued 'oracle' function telling us whether or not a given rephrasing realizes s^*. I will call this function f; it can be defined by

$$f(r) = \begin{cases} 1 & : \quad s^* \rhd r \\ 0 & : \quad \text{otherwise} \end{cases} \tag{5.6}$$

We can then use f to derive the sets $R^* \cap \rhd[s^*]$ and $R^* - \rhd[s^*]$ of valid and invalid rephrasings respectively, and hence define two sets $\mathbf{S}^+(R^*)$ and $\mathbf{S}^-(R^*)$ consisting of subsets of S:

$$\mathbf{S}^+(R^*) = \{\lhd[r] | r \in (R^* \cap \rhd[s^*])\} \tag{5.7}$$

$$\mathbf{S}^-(R^*) = \{\lhd[r] | r \in (R^* - \rhd[s^*])\} \tag{5.8}$$

$\mathbf{S}^+(R^*)$ contains sets which we know to contain s^*; $\mathbf{S}^-(R^*)$ contains sets which we know not to contain s^*. Thus we can construct a 'minimal' set $S^{\min}(R^*)$ known to contain s^* by

$$S^{\min}(R^*) = \bigcap \mathbf{S}^+(R^*) - \bigcup \mathbf{S}^-(R^*) \tag{5.9}$$

This is is minimal in the sense that, for a given R^*, no smaller set can be constructed which is guaranteed to contain s^*. In practice, different clarification techniques may produce a larger set due to incomplete use of the user-supplied data. I will denote this set (for some given R^* and clarification strategy) by S^{c}. Thus we have:

$$\{s^*\} \subseteq S^{\min}(R) \subseteq S^{\min}(R^*) \subseteq S^{\mathrm{c}} \subseteq S \tag{5.10}$$

For complete clarification, we would like to ensure:

$$\{s^*\} = S^{\min}(R) = S^{\min}(R^*) = S^{\mathrm{c}} \tag{5.11}$$

In Sections 5.3.5–5.3.13, I will discuss techniques for attempting to ensure each of these equalities. First, however, I will describe the consequences of incomplete clarification (Section 5.3.2) and two ways to reduce the number of rephrasings being considered (Sections 5.3.3 and 5.3.4).

5.3.2 Clarification failure

Since, in Te Kaitito, clarification is used as a strategy of last resort, it is extremely undesirable for a clarification strategy to produce an $S^{\min}(R^*)$ containing more than one semantic structure. At this point there is little more that can be done: the system must simply choose a parse at random. The best it can do is to warn the user that they may have been misconstrued by generating a message such as 'I *think* I understand what you mean...'.

In fact, the choice need not always be entirely random: if there is any variation between the scores assigned to the remaining parses by the probabilistic grammar, the system can still choose the most probable—even though the probability was not high enough to pass the syntactic filter module's threshold for automatic disambiguation. Note, however, that if the clarification stage has been reached, this is precisely because the variation in calculated parse probabilities has been deemed too low for reliable disambiguation, so even this method of choice can be regarded as effectively random.

5.3.3 Equivalent rephrasings

This section, and the next, will investigate two ways to reduce the size of the set R without reducing the precision of clarification, regardless of the particular clarification strategy being used.

It is possible that some rephrasing r_1 embodies exactly the same information as another rephrasing r_2; that is, $\lhd[r_1] = \lhd[r_2]$. In this case, the two rephrasings can be termed **equivalent**, which I will write as $r_1 E r_2$. E is clearly an equivalence relation, since $=$ is an equivalence. So we can use E to produce a partition R/E on the set R. This is useful: if we ask the user about the validity of some particular rephrasing, there is no need to ask about the validity of any equivalent rephrasing. Let g be a choice function on R/E—that is, any function such that $g([r]_E) \in [r]_E \ \forall [r]_E \in R/E$. Then we can define a subset R^- of R by

$$R^- = \{g([r]_E) \mid [r]_E \in R/E\}$$

R^- contains only a single rephrasing representing each equivalence class of rephrasings in R—that is, $|[r]_E| = 1 \ \forall r \in R^-$. It can be seen from Equations 5.7 and 5.8 that $\mathbf{S}^+(R^*) = \mathbf{S}^+(R^* \cap R^-)$ and $\mathbf{S}^-(R^*) = \mathbf{S}^-(R^* \cap R^-)$, and thus by equation 5.9,

$$S^{\min}(R^*) = S^{\min}(R^* \cap R^-)$$

That is to say, whatever choice we make for R^*, the amount of information which can be deduced from the user's responses is not reduced at all by removing equivalent rephrasings from R^*.

Thus, provided we can construct a suitable g, we can perform this operation in advance by choosing R^* from the members of R^- rather than R itself. Since it does not affect the amount of information available, this optimization can be performed before any of the clarification strategies described below, simplifying their implementation.

The problem remains of how to define g—that is, how to choose a single rephrasing from each group for presentation to the user. In vague terms, we would like a set of rephrasings which are as identical as possible in parts corresponding to identical parse structures, and as different as possible in the parts corresponding to differences in parse structures. However, I can see no straightforward way of accomplishing this: the correspondence between a part of a derivation tree and a part of an input string is complicated. A more tractable measure might be similarity to the original utterance: by choosing a rephrasing as much as possible like the input sentence, we might hope to make the features which do differ stand out more. This similarity metric could be calculated at the level of syntactic structures, by comparing total counts for different rules used in the derivation tree, or it could be done at the level of character strings, using the Minimum Edit Distance of Wagner and Fischer (1974) or something similar.

Another desirable criterion might be the 'naturalness' of the rephrasing in the user's estimation: out of the competing rephrasings, we should choose the one which seems the most natural. Fortunately, the probabilistic grammar discussed in Section 4.1 is ideal for this purpose: we can use it to score the syntactic structures corresponding to the candidate rephrases, and pick the highest score. The effectiveness of this technique might be limited: in rephrasing-based clarification, many of the rephrasings generated have unusual forms which would seldom be seen outside a clarification subdialogue. However, probabilistic parse ranking might at least be expected to weed out the most horribly convoluted rephrasings from the equivalence classes. Section 7.5.4 will give some results from the use of this technique.

5.3.4 Converse rephrasings

As well as classes of equivalent rephrasings, R may contain pairs of **converse rephrasings**. Two rephrasings r_1 and r_2 are converse iff $\lhd[r_1] \cap \lhd[r_2] = \emptyset$ and $\lhd[r_1] \cup \lhd[r_2] = S$. That is, r_2 can be interpreted as any syntactic structure in S which is *not* a possible interpretation of r_1. As in the case of equivalent rephrasings, r_1 and r_2 embody exactly

the same information: if the user tells us that $s^* \in \lhd[r_1]$ we can infer that $s^* \notin \lhd[r_2]$, and vice versa.

We can use this property, like the property of equivalence considered in the previous section, to reduce the size of R^*. Suppose that r_1 and r_2 in R^* are converse rephrasings. We can reduce R^* to a set $R' = R^* - \{r_2\}$. To show that this set still provides the same information, suppose first that $s^* \in \lhd[r_1]$. Then

$$
\begin{aligned}
S^{\min}(R') &= \bigcap \mathbf{S}^+(R^*) - \bigcup \mathbf{S}^-(R') \qquad (\text{since } \lhd[r_2] \notin \mathbf{S}^+) && (5.12)\\
&= \bigcap \mathbf{S}^+(R^*) \cap \lhd[r_1] - \bigcup \mathbf{S}^-(R') \qquad (\text{since } \lhd[r_1] \in \mathbf{S}^+) && (5.13)\\
&= \bigcap \mathbf{S}^+(R^*) \cap (S - \lhd[r_2]) - \bigcup \mathbf{S}^-(R') && (5.14)\\
&= (\bigcap \mathbf{S}^+(R^*) \cap S) - (\lhd[r_2] \cup \bigcup \mathbf{S}^-(R')) && (5.15)\\
&= \bigcap \mathbf{S}^+(R^*) - \bigcup \mathbf{S}^-(R^*) && (5.16)\\
&= S^{\min}(R^*) && (5.17)
\end{aligned}
$$

If $s^* \notin \lhd[r_1]$ then $s^* \in \lhd[r_2]$. In that case,

$$
\begin{aligned}
S^{\min}(R') &= \bigcap \mathbf{S}^+(R') - \bigcup \mathbf{S}^-(R^*) \qquad (\text{since } \lhd[r_2] \notin \mathbf{S}^-) && (5.18)\\
&= \bigcap \mathbf{S}^+(R') - (\lhd[r_1] \cup \bigcup \mathbf{S}^-(R^*)) \qquad (\text{since } \lhd[r_1] \in \mathbf{S}^-) && (5.19)\\
&= \bigcap \mathbf{S}^+(R') - ((S - \lhd[r_2]) \cup \bigcup \mathbf{S}^-(R^*)) && (5.20)\\
&= (\bigcap \mathbf{S}^+(R') - (S - \lhd[r_2])) - \bigcup \mathbf{S}^-(R^*) && (5.21)\\
&= (\bigcap \mathbf{S}^+(R') - (\bigcap \mathbf{S}^+(R') - \lhd[r_2])) - \bigcup \mathbf{S}^-(R*) && (5.22)\\
&\qquad\qquad (\text{since } \bigcap \mathbf{S}^+(R') \subseteq S)\\
&= (\bigcap \mathbf{S}^+(R') \cap \lhd[r_2] - \bigcup \mathbf{S}^-(R*) && (5.23)\\
&= \bigcap \mathbf{S}^+(R^*) - \bigcup \mathbf{S}^-(R^*) && (5.24)\\
&= S^{\min}(R^*) && (5.25)
\end{aligned}
$$

Thus, removing one of a pair of converse rephrasings does not affect the maximum clarification accuracy; and by repeatedly doing this we can form a set with no converse rephrasing-pairs without reducing maximum accuracy. Once more, the probabilistic grammar could be used to select the more natural of two converse rephrasings (or, if R has not been pruned for equivalent rephrasings, to select a single rephrasing from $[r_1]_E \cup [r_2]_E$).

It is not, however, clear that removing converse rephrasings is always a desirable optimization. If the clarification strategy involves asking the user yes/no questions

about single sentences, it can be more natural to present both alternatives explicitly—for example,

(5.26)
User	The fruit flies like a banana
System	What do you mean?
	1. The fruit does fly like a banana
	2. The fruit flies do like a banana

rather than

(5.27)
User	The fruit flies like a banana
System	Do you mean 'the fruit does fly like a banana'?

It is probably preferable always to be able to select a rephrasing which realizes one's intended parse, than to have to implicitly select it as the converse of another rephrasing. This arrangement produces more confidence that the system really has understood what the user meant—the implicit choice carries the same information from the system's point of view, but the exact form of the non-displayed converse rephrasing may not be clear to the user. It may, of course, be necessary to fall back on a yes/no question in any case if no converse exists in the original R.

Removal of converse rephrasings is not always useless, however: for example, it can be used to reduce the search space of an algorithm running on the set of rephrasings, and the converses reinstated afterwards for convenience. Section 5.3.12 describes one such case.

5.3.5 Producing a set of all possible rephrasings

Most of the rest of this chapter will be concerned with strategies for implementing rephrasing-based clarification in ways which will bring the result as close as possible to the optimal clarification process represented by Equation 5.11. In this section I will briefly describe the easiest parts of the process: creating the complete rephrasing-set R, and producing S^{c} (the fully clarified set of syntactic structures) from the set R^{*} and the user's assessments as to which of its members constitute valid rephrasings. Implementing these parts of the clarification process mostly consists of turning the definitions of Section 5.3.1 into algorithms.

Definitions 5.3 and 5.4 are easy to implement: Definition 5.3 is already implicitly implemented during the process of parsing and initial semantic interpretation to MRS. The implementation of Definition 5.4 is eased by the fact that LKB system and both

Te Kaitito's grammars are bidirectional: we simply take each semantic representation s in S and use LKB's generation module to generate every possible rephrasing for it. We can now produce a representation of R by combining the images of S's members and removing duplicates.

5.3.6 Inferring structures from clarified rephrasings

I will now jump to the other end of the clarification process—that of inferring a minimal set S^{c} of syntactic structures from the results of the consultation with the user. So let us assume for the moment that R^* has been chosen in some way and the user has supplied their responses. We can then implement Equations 5.7 and 5.8: we know $\lhd[r]$ for every rephrasing r, since these are the semantic structures from which they were generated in the first place. We also have a definition for the 'oracle' function f from the user's responses, which we can use to generate $R^* \cap \rhd[s^*]$ and $R^* - \rhd[s^*]$ (corresponding to those rephrasings in R^* which the user has deemed valid and invalid respectively). Using our values for $\mathbf{S}^+(R^*)$ and $\mathbf{S}^-(R^*)$ we can then implement Definition 5.9 using straightforward set operations.

I will now turn to the more difficult issues involved in attempting to ensure the other equalities of Equation 5.11.

5.3.7 Increasing the likelihood of unambiguous rephrasing-sets

This section deals with attempting to ensure that $\{s^*\} = S^{\min}(R)$. As previously mentioned in Section 5.3.2, $|S^{\min}(R)| > 1$ is a circumstance which we wish to avoid if at all possible. For a grammar of realistic size, it is probably impractical to attempt a formal proof that this will never occur; however, we can take steps to reduce its probability.

Firstly, when using the Māori-English grammar, the system can rephrase into a different language. Constructions which are ambiguous in one language are often unambiguous in another, as will be attested by anyone who has ever attempted to translate a pun. This is especially true in language pairs as different as English and Māori. Thus we can engage in clarifications such as this:

(5.28)

User	Kia ora, e hoa mā.
System	What do you mean?
	1. Hello friends
	2. Hello, O white friend

Secondly, we can augment the grammar and lexicon with unambiguous items intended only for use during clarification; for example, we could add an item corresponding to the second person singular pronoun, with orthography *you (by yourself)*. This does introduce the danger that these items might be used when generating sentences in other contexts; we do not really want the system to use the form *you (by yourself)* except during clarification. However, the probabilistic grammar described in Section 4.1 provides a straightforward solution to this: we can use the grammar's parse ranking to rank candidate response sentences generated by the system in exactly the same way as it ranks parses of the user's input. Provided that the 'special' lexical items occur rarely or not at all in the treebank used for training, generated sentences using these forms will always be assigned lower probabilities. Of course, any general treebank will naturally have very low incidences of such unusual forms; the only caveat is that we must be careful if training the grammar on transcripts of previous dialogues held with Te Kaitito.

There is another way to increase the number of different rephrasings produced: syntactic rephrasings can be augmented with semantic information. I will describe this technique in Section 5.4.1.

5.3.8 Assessing clarification strategies

We can now discuss various **clarification strategies**. I use the term 'strategy' to refer to the process of selecting R^*, presenting the rephrasings to the user in some way, gathering the user's responses, and turning them into the binary-valued function f on R^* defined by Equation 5.6. f and R^* can then be used to generate $S^{\min}$ as described in Section 5.3.6.

I will now define two evaluation metrics for use in assessing the usefulness of a clarification strategy.

Precision can be defined as $|S^{\min}(R)|/|S^{\mathrm{c}}|$, the minimum possible number of candidate parses remaining after *any* clarification strategy, divided by the actual number remaining. Its minimum value for a given S and R is thus $|S^{\min}(R)|/|S| \leqslant 1$ (corresponding to not a single parse being eliminated) and its maximum is

$|S^{\min}(R)|/|S^{\min}(R)| = 1$ (all parses which could possibly be determined invalid using this R are eliminated). (This definition is specific to a rephrasing-based clarification strategy, where $S^{\min}(R)$ is a lower bound on the number of remaining parses; a more generally useful definition might be $1/|S^{c}|$.)

Convenience is maximized by minimizing the amount of effort which the user has to expend; it might be defined in terms of (1) the number of rephrasings which the user must read and (2) the number of bits of information which the user must supply to the system. Under the scheme being considered here, the first quantity is an upper bound on the second—the greatest amount of information which can be requested is one bit ('Is this rephrasing valid?') per displayed rephrasing—but it is possible for the second quantity to be far lower, as will be seen below.

There is also a clear lower bound on the information required from the user if full clarification is to be achieved: since, whatever interaction the system has with the user, it must eventually select a single member of S, at least $\log_2 |S|$ bits will be required to specify this member uniquely. If less information than this is collected from the user, the choice must, in part, be arbitrary.

It is difficult to formalize the exact way in which quantities (1) and (2) should be combined to achieve a well-defined numerical measure of convenience, and I will leave this part of the definition vague for the present.

Precision is the more important quantity: making life more convenient for the user is no good if it results in an incorrect interpretation being made—this, after all, is likely to make life less convenient for the user in the long run.

Note that precision and convenience have been defined in terms not only of a particular S and R, but of a particular s^*. It is thus inaccurate to speak of the precision and convenience of a given clarification process where s^* is not known, let alone of a strategy in general. Nevertheless, I will extend the terms in various ways in the following sections. In some cases (such as exhaustive clarification, below) the strategy is sufficiently fixed that the terms may be rigorously extended to mean 'the precision and convenience of any clarification process under this strategy, regardless of R, S and s^*.' In others, such as single-rephrase clarification (Section 5.3.11) and interactive binary clarification (Section 5.3.13), I will make non-rigorous judgements based on an assumption that the probability of any interpretation is about equal.

5.3.9 Exhaustive clarification

Probably the most straightforward clarification strategy might be termed **exhaustive clarification**. We simply choose $R^* = R$ and determine the image of s^* by explicitly asking the user, for every r in R, whether r realizes s^*. The user receives a string of questions of the form 'Did you mean...? (yes / no)' (or in a GUI they might be presented with a check-box to mark for each rephrasing). This is clearly guaranteed to maximize precision: the user is interrogated for every available nugget of information, so the lower bound $S^{\min}(R)$ is achieved. For exactly the same reason, it is guaranteed to minimize convenience: the user is forced to read every member of R, and must supply the theoretical maximum of $|R|$ bits of information.

Exhaustive clarification does have the advantage of being very straightforward in implementation: the choice of R^* is trivial since R^* always equals R; the presentation of choices and collection of information always happen in the same manner; and the definition of f is trivial since the user has been forced, in effect, to define it explicitly. And if equivalent and converse rephrasings are first removed as described in Sections 5.3.3 and 5.3.4, the user can be spared from some entirely redundant questions.

Even without equivalent or converse rephrasings, exhaustive clarification is likely to query the user for redundant information. This creates a danger that the user, accidentally or maliciously, will supply conflicting information; the clarification process could find that $S^{\mathrm{c}} = \emptyset$. Various techniques could be used to deal with this: the user could be put through the ordeal again, or an interpretation chosen at random. Since I am introducing exhaustive clarification chiefly as a theoretical limit rather than a practical proposal, I will not pursue these possibilities any further.

5.3.10 Vacuous clarification

It seems appropriate, after a discussion of exhaustive clarification, briefly to mention the opposite extreme. **Vacuous clarification** maximizes convenience while minimizing precision: it is, in fact, no clarification at all, and is thus not specific to rephrasing-based clarification. Vacuous clarification consists of choosing $R^* = \emptyset$ so that $S^{\min}(R^*) = S$, after which a parse must be chosen at random, as happens when any clarification method fails to produce a single remaining candidate parse.

5.3.11 Clarification with a single rephrasing

I will now describe a strategy which gives high convenience in many cases, at the cost of precision in some cases. **Single-rephrase clarification** tries to find, for each s in S, a rephrasing r such that $\lhd[r] = \{s\}$—that is, r does not realize any other semantic structure in S. Such a rephrasing is thus *unambiguous* in itself: if it is known that one such rephrasing is valid, s^* can be deduced from this fact alone. If such an r can be determined for each member of S, a fairly convenient strategy becomes viable: the user is presented with a list of unique rephrasings, one for each member of S, and simply chooses the one corresponding to s^*.

When this strategy works, it achieves the lower bound for information demanded of the user: $\log_2 |S|$ bits. In effect, the mutually exclusive nature of the rephrasings on the menu allows the system to infer a great deal more information than the user supplies directly: the validity of one rephrasing implies the invalidity of all the others. Exhaustive clarification on the same set of rephrasings would explicitly request another $|R^*| - 1$ bits of information.

If there is a large number of parses (more than around ten, say) it may be inconvenient for the user to read them all; however, when clarification is only used as a last-resort disambiguation technique (as in Te Kaitito) this should seldom occur: the disambiguation process should have removed most of the possibilities.

This strategy's performance with respect to precision is more worrying: there is no guarantee that a unique rephrasing will exist for every member of S. In this case, all the structures which do not have an unambiguous rephrasing must be grouped together under one menu item, producing a menu along the lines of

(5.29)

User	The dog chased the cat under the table.
System	What do you mean?
	1. Under the table, the dog chased the cat.
	2. The dog chased the cat which was under the table.
	3. Something else.

If there is only one parse without an unambiguous rephrasing, the problem is not too great: the user must read through all the rephrasings, realize that none of them express the intended sense, and select the 'something else' item in the hope that the system's inexpressible parse corresponds to their intent.

If there is more than one parse without an unambiguous rephrasing, the situation is worse: the user selects 'something else' and the system must choose at random between

the semantic structures which this item represents.

Single-rephrasing clarification as a system development tool

The user is not the only beneficiary of a rephrasing which maps onto a unique semantic structure: it is also useful during development of disambiguation and clarification code to be able to map a semantic structure unambiguously onto a rephrasing. A rephrasing is far easier for a developer to read than a symbolic representation of a structure, and can be used to keep track of which structures are kept and removed during different stages of the disambiguation and clarification process.

5.3.12 Brute force optimal clarification

A clarification strategy might be considered optimal if it attained maximal precision using as few rephrasings as possible. There is still scope for variation in the way in which the rephrasings are presented and the responses processed, but the core problem is one of optimization over a search space consisting of $\mathcal{P}(R)$, the set of all possible subsets of R.

The simplest algorithm is, as ever, brute force: consider every possible subset of R and choose the smallest one which gives precision equal to choosing R itself (as with the elimination of equivalent rephrasings, the probabilistic grammar could be used to choose between optimally precise rephrasing-sets of equal size). Such an algorithm is hardly innovative or elegant, but for small sets of rephrasings it might well be practical. The size of $\mathcal{P}(R)$ is $2^{|R|}$. Thus, for example, only 256 subsets need be considered when $|R| = 8$. There are easy optimizations to be made, too. Equivalent and converse rephrasings can be removed in advance, and the converses replaced after an optimal subset has been found. And if some particular subset of R gives suboptimal precision, all its subsets can be excluded from the search.

Unfortunately, brute force optimal clarification is rendered useless by a simple fact: for any R^*, calculating $|S^{\min}(R^*)|$ requires knowledge of f—that is, we must know, for each $r \in R^*$, whether the user considers r a valid parse. In particular, we must have defined f on R in order to calculate $|S^{\min}(R)|$. So, to guarantee a maximally convenient clarification process, we must already have put the user through a minimally convenient clarification—at which point we may as well use exhaustive clarification (Section 5.3.9) since we have all the data to hand.

However, all is not lost. We can make some assumptions about the distribution of

correct rephrasings and, crucially, make the calculation of R^* incremental and interactive. The next section describes such a strategy.

5.3.13 Clarification by interactive binary choice

Introduction

I will now discuss a strategy which guarantees to produce an R^* such that $S^{\min}(R^*) = S^{\min}(R)$, although it is not clear that the size of R^* will always be minimal.

Attempting to construct a clarification strategy based on the derivations in Section 5.3.1 runs up against the problem described in the previous section: although the definition of $S^{\min}$ in Equation 5.9 is easy to implement, it requires knowledge of f; that is, for any *potential* choice of R^* which we wish to assess, we must ask the user about each rephrasing in R^*. This makes it difficult to find a minimal 'complete' R^*—that is, R^* such that $S^{\min}(R^*) = S^{\min}(R)$ and $|R^*| < |R'| \; \forall R' : R' \subseteq R \wedge S^{\min}(R') = S^{\min}(R)$. The problem is one of the 'chicken and egg' type: we wish to search through the space of all R^* in order to minimize the number of questions posed to the user, but in order to do so we must first ask every possible question in order to define the search space.

This strategy, which I will refer to as **interactive binary** clarification, attempts to get around the problem by defining R^* iteratively: at each stage, the user is asked for another data point on f, the size of R^* is reduced, and all the information supplied up to this point is used to determine the next question to pose to the user. At each stage, the strategy attempts to reduce the size of R^* as much as possible.

Clarification as binary space partitioning

This strategy, like the previously described exhaustive rephrasing, presents a series of yes-no questions to the user; in this case, however, the system throws away some parses after each answer. Provided we can find suitable clarification questions, this can give a dramatic improvement over single-rephrase clarification in the first measure of convenience (the amount of information presented to the user): rather than forcing the user to read through n rephrasings to find the one which exactly reflects their intended meaning, we can (in the best case) get away with showing them only $\log_2 n$ rephrasings.

The question then becomes how to maximize convenience within this framework of binary choice: we certainly don't want to ask about every possible rephrasing, and ideally we want to minimize the number of questions posed to the user. In effect, the system needs to display the qualities of a good questioner in a game of 'twenty

questions'.

It may be useful to regard the parses as points in a many-dimensional rephrasing-space. Representing each rephrasing as a vector starting at the origin, we can locate a parse uniquely by defining its position along each of these vectors to be 1 (signifying that the rephrasing can be formed from that parse) or 0 (signifying that it can't). Note that the rephrasing-vectors need not be linearly independent: it is possible that some rephrasing conveys information which can be deduced from a combination of other rephrasings. If the rephrasings *are* linearly independent, the dimensionality of the space will be R; otherwise, it will be smaller.

It might also be possible in some cases to find larger sets of rephrasings which completely partition the set of interpretations, allowing binary questions to be replaced with ternary or higher-order ones. In this case the three (or more) rephrasings would be grouped into a single axis with three (or more) points along it, corresponding to the disjoint subsets of S they are able to distinguish. The logical conclusion of this is the single-rephrase technique described in Section 5.3.11, which corresponds to forming—if possible—a single axis with $|S|$ distinct points on it; choosing any one of these points then disambiguates the utterance fully. For the present, I will ignore these extensions and confine myself to the case of binary questions and binary-valued co-ordinates.

If two parses occupy the same point in this space, there is nothing we can do: none of the rephrasings can distinguish between them (though we can still hope that the user meant neither of them). But distinguishing parses through a series of yes-no questions can now be seen as a problem in efficient binary space partitioning. We wish to construct a sequence of planes in this vector space—each one normal to one of the rephrasing-vectors and cutting it between the points 0 and 1 along it—until we have partitioned off a single point corresponding to the correct parse. And we wish to minimize the number of planes used to do this; this is equivalent to minimizing the number of questions we ask the user.

Similar problems occur in the construction of rendering engines for three-dimensional scenes in the field of computer graphics, where efficient rendering algorithms can be implemented by using a sequence of planes to successively split the space under consideration into smaller and smaller divisions; Fuchs, Kedem, and Naylor (1979) developed the Binary Space Partitioning algorithm for this purpose. Unfortunately, the problem is sufficiently different that computational geometry algorithms cannot be directly applied here: amongst other difficulties, they tend to be designed for three dimensions only, and are not restricted to planes normal to the axis.

		Parse							
	sheep number	sg	sg	sg	sg	pl	pl	pl	pl
	fish number	sg	sg	pl	pl	sg	sg	pl	pl
	attachment	high	low	high	low	high	low	high	low
Rephrase	In the river, the sheep saw the fish.	■		■		■		■	
	The sheep saw the fishes in the river.			■	■			■	■
	The sheep has seen the fish in the river.	■	■	■	■				

Table 5.1: A full parse-rephrasing table for 'The sheep saw the fish in the river'.

I propose the following algorithm: choose a plane which produces the most even split of points (parses) within the space. Pose the question corresponding to this plane, then discard all the points on the non-desired side of it. Continue, choosing planes in the same manner, until you have only a single parse remaining, or you run out of planes.

Precision and convenience

It is easy to see that this clarification strategy is maximally precise: the termination condition for the iteration is that the dimensionality has been reduced to zero—that is, there are no more rephrasings which can differentiate the semantic structures within rephrasing-space. If there is more than one structure left, there is no way to pick the right one using members of R.

It is somewhat harder to prove that the strategy is maximally convenient—and indeed, I am not entirely sure that it is. I will not attempt a proof here, but a promising approach would seem to be from information and coding theory: a sequence of rephrasing judgements clarifying an interpretation could be regarded as a binary code. Assuming an even distribution of interpretations, a proof could be attempted that the code constructed by this scheme has minimal length.

Example

Consider Example 5.3.1 ('The sheep saw the fish in the river.'), and suppose that it has resulted in the eight parses described on page 51. Suppose also that the following three rephrasings have been generated:

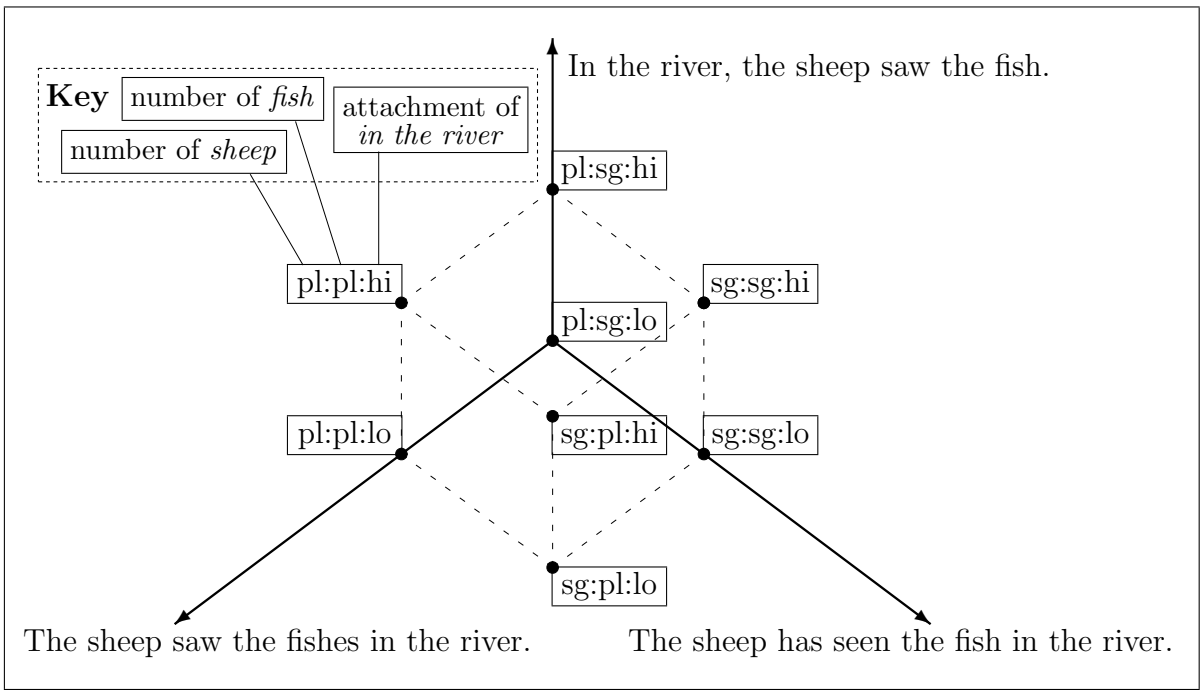

Figure 5.2: Parses as points in rephrasing-space.

- In the river, the sheep saw the fish.
- The sheep saw the fishes in the river.
- The sheep has seen the fish in the river.

This is an unrealistic set of rephrasings: it is very small, and the rephrasings are orthogonal—that is, they each clarify exactly one ambiguous feature of the original sentence. However, they make a useful example.

The relation between parses and rephrasings is shown in Table 5.1. Here, rows represent rephrasings, columns represent parses, and a cell is marked when the parse corresponding to its column can be realized by the rephrasing corresponding to its row. It is clear from the diagram that the parse can be fully clarified using only these three rephrasings: each clarifies a different ambiguous feature, so in combination they uniquely specify a parse.

The relationship is somewhat clearer when the parses are envisaged as points in rephrasing-space. Figure 5.2 shows this arrangement. Each axis has two possible co-ordinates, corresponding to the user's assessment of the associated rephrasing as invalid (the origin) or valid.

5.3.14 Generalization of rephrasing-based clarification

The above sections are couched in terms of 'semantic structures' and 'rephrasings', but in fact they deal with quite a general case: we have a set S containing a member s^* which we wish to determine, a binary relation $\triangleright$ between S and another set R, and a function f on R defined by

$$f(r) = \begin{cases} 1 & : \quad s^* \triangleright r \\ 0 & : \quad \text{otherwise} \end{cases}$$

which in our case is implemented by the user's responses. In any such situation, the same bounds and relations hold, and the same clarification can be applied. (The development of the clarification techniques does, however, assume that there is a cost associated with calling f and attempts to minimize this cost.)

This kind of deduction—attempting to uniquely determine a member of a set by probing its image under some relation—is often found in the field of expert systems. Rephrasing-based disambiguation is a far more definite and constrained problem than those addressed by real expert systems, but there are nevertheless strong parallels—for example, in the case of the medical diagnosis system INTERNIST (Miller, Pople, and Myers, 1982). Alty and Coombs (1984), in their account of INTERNIST, give a diagram of a relation between diseases and manifestations which bears a strong similarity to Figure 5.1; and Jackson (1990) gives the following description of an INTERNIST consultation:

> At the start of a session, the user enters a list of manifestations. Each of these evokes one or more nodes in the disease tree. The program creates a *disease model* for each such node, consisting of four lists:
>
> (1) observed manifestations not associated with the disease;
>
> (2) observed manifestations consistent with the disease;
>
> (3) manifestations not yet observed but which are always associated with the disease;
>
> (4) manifestations not yet observed but which are consistent with the disease.

Replacing the words 'observed', 'manifestations', and 'disease' with 'queried', 'rephrasings', and 'parse' respectively, the first three items in the list become, for some $s \in S$,

descriptions of the sets $R^* - \triangleright[s]$, $R^* \cap \triangleright[s]$, and $\triangleright[s] - R^*$ respectively. (There is no analogue to the fourth item, since we have no concept of rephrasings 'consistent' but not 'associated' with a parse: they are either definitely associated or entirely inconsistent.)

5.4 Other syntactic clarification techniques

5.4.1 Augmented rephrasings

Adding semantic information to syntactic rephrasings

One possible extension of the rephrasing scheme would be to incorporate 'parenthetical' information from the dialogue context into the rephrasings, helping to make them clearer. Rather than simply generating strings from the semantic representation of the original input sentence, the system could use the resolved semantic interpretations and attempt to add extra information from the context. The 'lookahead' approach to disambiguation means that the initial interpretation pipeline forms complete representations for all possible interpretations, so we can make use of any semantic-level information about the binding of the referents that has already been generated. This would result in questions which, although still attempting disambiguation at a purely syntactic level, would include semantic information. So instead of pure equivalent rephrasings of the input, such as this:

(5.30)
User The fruit flies like a banana
System What do you mean?
1. The fruit does fly like a banana
2. The fruit flies do like a banana

users might be presented with something more akin to the following example.

(5.31)
User I put some fresh fruit in the bowl
System okay
User Andy showed me some mutant fruit flies in the genetics lab
System okay
User The fruit flies like a banana
System What do you mean?
1. The (fresh) fruit (in the bowl) does fly like a banana
2. The (mutant) fruit flies (in the genetics lab) do like a banana

In fact, even adding the parenthetical information to the original, ambiguous utterance can result in an unambiguous utterance:

(5.32)
...
User The fruit flies like a banana
System What do you mean?
1. The (fresh) fruit (in the bowl) flies like a banana
2. The (mutant) fruit flies (in the genetics lab) like a banana

Of course, this technique can only be of use when the ambiguous utterance refers to entities already in the dialogue context; where it introduces new referents, no parenthetical information can be added. It is also unusable where there is also unresolved ambiguity at the semantic level: if *the fruit* has more than one possible binding, the system won't know whether to augment the rephrasing as, say, 'the (fresh) fruit' or 'the (mouldy) fruit'. Of course, the system could present all the possible augmentations and attempt to clarify the syntactic and semantic levels simultaneously, but, for the reasons given at the start of this chapter, I have chosen not to make use of this kind of multi-level technique.

Augmented rephrasings of this kind have two benefits: firstly, they make it easier for the user to see the indicated interpretation by differentiating it further from the other alternatives, and by tying it explicitly to entities in the dialogue context. Secondly, they increase the probability of finding enough distinct rephrasings to clarify an utterance fully (as discussed in Section 5.3.7): as Example 5.32 shows, identical rephrasings can be made thoroughly distinguishable by interpolating semantic-level information.

Making accommodation explicit

As described in Section 2.1.2, Te Kaitito *accommodates* definite NPs if they cannot be resolved: if the user says 'the dog walked' when no dog has previously been mentioned, Te Kaitito will assume its existence. During rephrasing-based clarification, the rephrasings could be augmented to make this kind of accommodation explicit. For instance, suppose in the following example that fruit has previously been mentioned, but fruit flies haven't.

(5.33)
User The fruit flies like a banana
System What do you mean?
1. The (fresh) fruit (in the bowl) flies like a banana
2. There are some fruit flies, and these fruit flies like a banana

This is particularly useful in cases where the system finds an interpretation which the user would probably never have imagined—for example, the NP-verb-NP reading of the sentence 'The individual costs matter.' Making any accommodation explicit ('There is an individual, and this individual costs matter.') helps the user—in part simply by isolating the supposed noun phrase—to see the indicated reading.

5.4.2 Referent-based syntactic clarification

Rephrasing is by no means the only way to resolve syntactic ambiguity. Consider this example:

(5.34)
A: The fruit flies like a banana.
B: Are you talking about fruit or about fruit flies?
A: Fruit flies.

This is in some ways more elegant and natural than clarification by a full rephrasing: only one feature of the sentence is clarified, and the rest of the structure is then inferred by B.

In some cases, this could be quite easy to implement in a computer dialogue system: in the case above, the system could go through the two semantic structures seeking a nominal unique to each structure, then slot them into a canned 'Are you talking about <n1> or <n2>' question, and choose the parse containing whichever nominal the user selected.

This technique is more versatile than it might at first appear. At its root it consists of calculating the difference between the sets of relations corresponding to each of two candidate parses. Once these discriminant relations have been found, they must be slotted into a clarification question. Although the process seems most natural with nouns, it can also be applied to any part of speech or linguistic phenomenon which can be nominalized; for example verbs ('Are you talking about flying or liking?') and even attachment ambiguity ('Are you talking about seeing with a telescope or a girl with a telescope?').

Minimal Recursion Semantics, the formalism used for semantic representation in Te Kaitito, is ideally suited to this application: as the name implies, the semantic representation of a sentence is embodied as a flat, non-recursive structure—a bag of relations. Thus, performing set-theoretic operations on the relations contained in different interpretations should pose few problems.

Where there are more than two candidate interpretations, all the techniques discussed for rephrasing-based clarification (discussed in Section 5.3) can be brought to bear. The situation is entirely isomorphic—the only difference being that the members of the set R are not rephrasings but relations. However, the method for presenting the members of R to the user is less well-defined, and it may be that even if there is some subset of R which can uniquely identify an interpretation, its members cannot be worked into a comprehensible question that can be put to the user.

This type of clarification will not be further explored in this thesis, but it is certainly worthy of consideration.

5.4.3 Hybrid clarification

There is no reason (other than ease of implementation) why the system should be restricted to using only one of the syntactic clarification techniques described above. A technique can be selected as being most appropriate to a particular clarification instance—for example, referent-based clarification where the ambiguity can be resolved through disambiguation of a single nominal. Techniques could also be combined dynamically: single-rephrase clarification could be used in the first instance, falling back to a more reliable technique if the user selects the ambiguous 'other' option. Or dynamic binary tree clarification could be used to pare large sets of parses down to a manageable number, at which point single-rephrase could be employed (when sufficient unambiguous rephrasings exist).

5.5 Semantic clarification

Once a unique syntactic representation has been selected, it is still possible that there will be more than one interpretation at the semantic level. At this point, semantic clarification must be used. This thesis does not involve any work on semantic-level clarification, but I will briefly describe the current implementation.

In Te Kaitito, the only semantic semantic ambiguity arises in the binding of definite NPs—for example, 'it' or 'the dog' could refer to different entities in the dialogue's common ground. In this case, the system extracts the ambiguous NP from the user's utterance, and rearranges the remainder into a *wh*-question:[1]

[1]For the purposes of this example, I ignore the fact that the referents could in this case be disambiguated by the saliency filter module (Section 4.2.3).

(5.35)

User	A big dog chased a small cat
System	okay
User	A small dog chased a big cat
System	okay
User	The cat bit the dog
System	Which cat bit the dog?
User	The small cat
System	Which dog did the cat bite?
user	The big dog

Note that the system iterates over the ambiguous referring expressions, asking a question for each in order to fully clarify the utterance.

These semantic clarification questions have some unusual properties: the *wh*-pronoun does not have to be fronted ('You chased which dog?') and the intonation is unusual ('You chased *which* dog?'). Ginzburg and Sag (2000) discuss the features of such questions in more detail.

Once the user has supplied the answer, the system must integrate the original ambiguous utterance, the echo question, and the user's reply to form a coherent semantic representation.

Since clarification questions are asked in order of increasing level, it is of course possible that the user will be asked one or more syntactic clarification questions followed by one or more semantic clarification questions:

(5.36)

User	The fruit flies like a banana.
System	Do you mean
	1. The fruit does fly like a banana, or
	2. The fruit flies do like a banana?
User	1
System	Which fruit flies like a banana?
User	The fresh fruit.

The error accommodation facilities described in the next chapter will add another potential level to this concatenation of clarification.

Chapter 6

Dealing with errors

> "The lawn is full of fletchers," Della told me one morning... I got dressed and went downstairs and looked up the word in the indispensable Century. A fletcher, I found, is a man who makes arrows. I decided, but without a great deal of conviction, that there couldn't be any arrow-makers on my lawn at that hour in the morning and at this particular period in history.
>
> —James Thurber, 'What do you mean it was brillig?' (Thurber, 1945)

One important aspect of utterance interpretation, especially in a CALL application such as Te Kaitito, is the treatment of errors in user input. Errors are not a major focus of this thesis, but I will describe a straightforward scheme for dealing with minor errors which can easily be incorporated into the current disambiguation framework and makes considerable use of it.

I have described several techniques for selecting between multiple possible interpretations of a single input. But how is the system to deal with the opposite extreme—input without a single valid interpretation? It can, of course, simply give up and produce a message such as 'I don't understand'. This, however, is unlikely to be very helpful to a language learner, forcing them to resort to a blind trial-and-error approach to correcting their errors. Instead, the system could attempt to divine what the user *meant* to say. There are several stages of the pipeline at which this kind of divination could occur. One approach is to augment the grammar with *mal-rules* describing known error patterns (Weischedel, Voge, and James, 1978; Sondheimer and Weischedel, 1980).[1] At a semantic level, the technique of accommodation described in Section 2.1.2 could be

[1] Mal-rules have usually been formulated as meta-rules which operate *on* the grammar rather than as rules *of* the grammar. However, there seems to be no compelling reason why mal-rules (albeit of a more specific type) could not be implemented using suitably flagged rules within the grammar itself.

viewed as error tolerance of this type, although accommodation does not necessarily indicate an error.

In this chapter, I will consider another approach designed to deal with a restricted class of errors, namely those which result in a string very close (at a character level) to the correct input string (that is, the string corresponding to the user's intended meaning). (I will shortly attempt a more specific definition of 'very close'.) The method of correction is modification of the input at a character level, but this does not mean that it is entirely restricted to correcting errors in spelling or typing: errors made at the semantic or (especially) the syntactic level can often manifest themselves as minor differences at the character level. For example, in

(6.1) *He never did cared for the river, did Montmorency.

an incorrect inflection of the verb *care* has manifested itself as the insertion of a single letter.

The proposed principle is simple: we make minor changes to the input in the hope that one of them will transform the user's erroneous input into the input they originally intended to provide.

6.1 Relation to disambiguation

The work of this thesis deals only with minor typing and spelling errors, but gives a principled account of how their detection and correction may be regarded as part of the disambiguation process and implemented as such. Hobbs et al. (1993) and Menzel and Schröder (1999) integrate error handling with disambiguation in a similar fashion, and the schemes described in Section 2.2 can cope with errors in input provided that there is enough other information available to allow an intended meaning to be inferred. Errors are accommodated by making extra assumptions (in the system of Hobbs et al.) or retracting extra constraints (in the system of Menzel and Schröder). Menzel and Schröder give particular attention not only to error detection but also to appropriate responses, since their work is concerned with a language learning system.

6.2 Perturbing whole utterances

Given an utterance string u and some metric d which defines a numerical distance between any two strings, we wish to search the space $S_\epsilon(u) = \{v | d(u, v) < \epsilon\}$ for some

ϵ representing (intuitively) the size of a fairly 'minor' change to a string. This must take place in a module at the very start of the pipeline, before syntactic parsing. I will refer to this technique as **perturbation**.

Clearly much depends on the choice of d. Perhaps the most obvious candidate is the *minimum edit distance* defined by Wagner and Fischer (1974). For now I will assume that this is the metric being used, until we come to discuss the processing of strings at the word level.

At first glance, the search space might seem discouragingly large: even if the permissible changes are restricted to extremely minor ones, the range of possibilities is vast: for a twenty-character string written in an alphabet of twenty-six characters, there are $25 \times 20 = 500$ distinct strings which differ from it only in a single character, and $26 \times 21 = 546$ distinct strings which can be created by inserting a single character. Clearly, it is computationally unfeasible to create all perturbations within a certain edit distance and attempt to run them all through the interpretation pipeline.

Fortunately, the search space can be greatly constrained by a simple observation: every valid sentence consists of a string of valid words. Thus our search space becomes restricted to $S_\epsilon(u) \cap W$, where W is the set of all strings consisting of concatenations of valid words.

One way of implementing this constraint is to use a simple spell-checker as a filter for the perturbed strings before giving them to the interpretation pipeline. Only a small proportion will pass, and spell-checking is far quicker than attempting a syntactic parse.

6.3 Perturbing single words

Another method, yet more efficient, would be to split the input into words and then perform perturbations on the individual words. The size of the search space thus becomes geometrically proportional to the number of words rather than the number of letters. Care must be taken not to exclude perturbations which would split or join words of the original input—in the first case, the perturbation algorithm run on the words can simply be allowed to split them; in the second, words must be considered pairwise as well as individually to allow for the possibility of joining them.

I will now discuss techniques for correcting a single word, deferring for the time being methods for locating the erroneous word in the first place.

There has been a large amount of research into algorithms for checking and correcting spelling at the word level (Kukich (1992) provides a good survey) and implemen-

tations of such algorithms are now commonplace in text editing and word processing applications. In particular, there are effective techniques for finding known words (or sequences of words) of which a given non-word text string could be a plausible misspelling. It would be straightforward to run such a spell-checking algorithm on the input string, and attempt to parse it with plausible substitutions for misspelled words.

Usually, the best that spell-checking algorithms can do is to suggest a list of alternatives for the user to choose between: their lack of sensitivity to syntactic and semantic context means that they are unable to pick a replacement with any degree of reliability. By integrating a spelling checker into Te Kaitito, we can do far better, since we can use the interpretation pipeline to reject impossible substitutions and the disambiguation pipeline to weed out implausible ones.

Effectively, we can use a spell-checker both to implement the dissimilarity metric d, and to constrain the search space to strings consisting of valid words. If we have an input utterance consisting of words $w_1, \ldots, w_n$, and a spell-check algorithm s which produces a list of substitutions $s_1(w), \ldots, s_{m_w}(w)$ for a word (ranked in decreasing order of plausibility), then we can define d by

$$d((w_1, \ldots, w_n), (w_1, \ldots, s_j(w_i), \ldots, w_n)) = j$$

—that is, if we take an input sentence, and substitute a single word with the jth-best suggested substitution from the spell-checker, the new sentence is at a distance of j from the original. (The distance is defined to be ∞ when one string cannot be transformed into another by a single word-substitution from the spell-checker's suggestion list.)

The definition could perhaps be extended to multiple word-substitutions by summing the ranks of the substitutions, but since I have left vague the spell-checker's notion of a 'plausible substitution', it is impossible to formally compare, say, the relative distances of a single rank-2 substitution and two rank-1 substitutions.

6.3.1 Word similarity metrics

In the preceding discussion I have regarded the spell-checker as a black box, producing oracular judgements on the similarity of words by their ranking in its substitution list. In fact, this is sufficient for a simple implementation: there are freely available spelling checkers (e.g. Kuenning, 2001; Atkinson, 2005) which could be interfaced with Te Kaitito to produce such suggestion lists. Provided that substitution was only attempted on one word at a time, and that there was some way to coerce the spelling

checker into producing substitution lists even for words in its lexicon, there would be no need for a well-defined distance metric. However, it is clearly desirable to be able to talk in more concrete terms about the distance between a word and its substitution. Most spelling checkers only *rank* suggested substitutions rather than assigning them distances from the input word; even if an off-the-shelf component is used to produce the substitution list, there is nothing to stop Te Kaitito implementing an algorithm to assign a more concrete score to the suggestions.

Again, the most obvious choice is the minimum edit distance, this time applied to a word-pair rather than a string-pair. But this is very unlikely to be the best measure of how close an erroneous word is to the intended one. To construct a serious model for the distance metric, one would require a corpus of incorrect sentences paired with their intended forms. Te Kaitito could be instrumented to amass such a corpus during normal operation, but it is beyond the scope of this thesis.

A less data-driven approach would be to hypothesize the causes of errors. The two obvious ones appear to be misspelling and mistyping, which give rise to two different distance models. It would seem sensible to calculate distances separately in spelling-space and typing-space, then combine them—probably by taking the minimum, since either one alone could explain a plausible substitution. In spelling-space, distance would be proportional to how similar two words sounded; in typing-space, it would be proportional to the similarity of the actions required to type them. In spelling-space, *dough* and *doe* would be near to one another; in typing-space, they would be widely separated, but *dough* and *rough* would be very close.

These ideas are only a brief sketch of how a realistic word-distance metric might be defined; Kukich (1992) gives a more thorough treatment.

6.4 Fitting perturbation into the disambiguation framework

Conceptually, the perturbation procedure fits quite neatly into the framework described in Chapter 3. The process can be split into two parts: the actual perturbation, producing a potentially large number of candidate sentences close to the original utterance; and the application of the metric D to rank them in terms of similarity to the original input. Of course, since the generated perturbations are constrained by a maximum value of D, the similarity ranking is done at the same time as the perturbations are generated. But, in the framework of this thesis, the former belongs to the disambiguation

process and the latter to the initial interpretation process.

On the 'domain' axis, perturbation as it is described here can be classified as a global data source—although it is possible to envisage a more sophisticated scheme capable of learning that, say, a particular speaker consistently misspelled certain words.

With these classifications, it is easy to locate perturbation—at least theoretically—within the overall interpretation process: operating as it does at the very shallow level of characters and single words, perturbation generation should occur right at the start of the interpretation pipeline, before syntactic parsing; by the same token, the rankings of perturbations should be used at the very end of the disambiguation pipeline; and clarification between different perturbations, if it is required, should happen at the start of the clarification pipeline. Figure 6.1 shows the augmented interpretation pipeline.

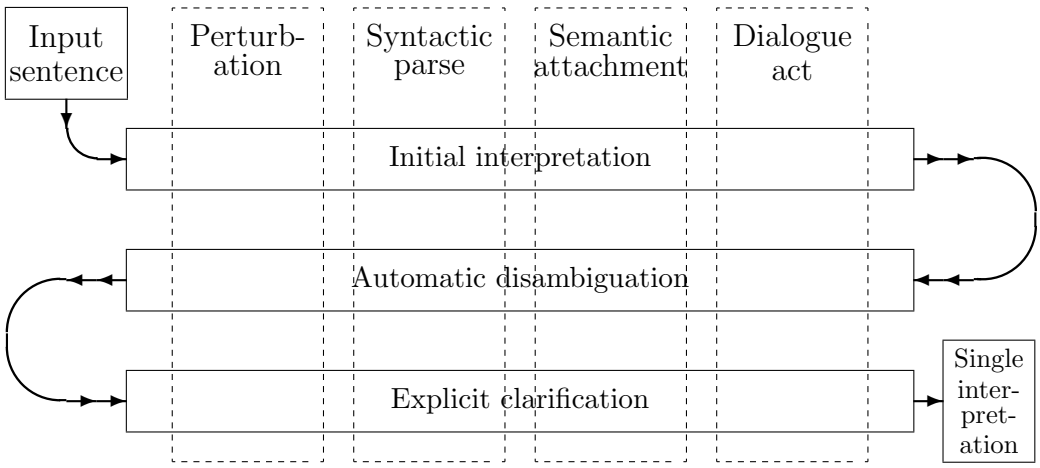

Figure 6.1: The position of perturbation within the interpretation pipeline.

Ideally, then, the process would be as follows: for any input sentence, all the perturbations within a certain (small) distance are produced, and they are all fed onward into the interpretation pipeline. Most of them will probably fail to parse, but a few may be successfully interpreted. Now every surviving perturbation, and every syntactic parse, semantic attachment and dialogue act resolution thereof, is passed into the disambiguation pipeline. Many more perturbations will probably be weeded out here: there might be references to things not in the dialogue context, removed by the accommodation module; and there might be parses which, while syntactically valid, are so unlikely that they are pruned by the probabilistic grammar. Finally, if more than one interpretation remains after syntactic disambiguation, the metric d can be used to weed out any perturbations too far from the original utterance.

Finally, there is the clarification pipeline. As befits its level, the perturbation stage comes first here: before the user can be asked a question about the syntactic structure of the sentence, the intended input string must be definitely determined. Fortunately, this clarification step is far easier than those further along the pipeline: since strings of raw text are the objects under consideration, they can simply be presented to the user as they are. A clarification process encompassing both the string and syntax level might look something like the following:

(6.2)

User	The fruit flees like a banana.
System	Did you mean—
	(1) The fruit flees like a banana.
	(2) The fruit flies like a banana.
	(3) The fruit fleas like a banana.
User	2
System	Did you mean—
	(1) The fruit flies do like a banana.
	(2) The fruit does fly like a banana.
User	1

As with the other levels of clarification, character-level clarification is intended only as a last resort: it is expected that there will in most cases only be one candidate string remaining after the automatic disambiguation process.

6.5 Improving the efficiency of perturbation

In spite of the optimizations possible by perturbing at the word level rather than the character level, perturbing every input sentence is still a computationally expensive proposition. The obvious optimization to make is only to perturb utterances in which the system is able to detect an error. We can classify character-level errors with respect to Te Kaitito's pipeline structure, according to the level at which the error is detectable:

1. The error results in a non-existent word; for example, 'Come live with me and *bea my love.'

2. The error transforms a word into a different word, but the utterance is not syntactically valid after this transformation. For example, 'Come live with me and *bee my love.'

3. The error transforms a word into a different word, and the resulting sentence parses syntactically, but it seems impossible or grossly implausible at a semantic level. For example, 'Come live with me and be my *glove.'

4. The error transforms a word into a different word, creating a syntactically valid, semantically plausible sentence with an incorrect meaning. The error might become apparent at a later stage in the dialogue, but there is nothing in the utterance itself or the preceding context from which it might be deduced. For example, 'Come *lie with me and be my love.'

The first case is the easiest to deal with: not only is the error immediately apparent, it is immediately localized. We can produce some likely perturbations of the non-existent word, and ignore the rest.

The last case seems insoluble: at a later stage in the dialogue, it may (in a future version of Te Kaitito) be possible to correct the error by, for example, making a contradictory assertion; but by that point re-disambiguating the original utterance is neither practically feasible nor particularly relevant.

The two intermediate cases, however, are more tractable. The second case, in particular, might be solved by means similar to those described for the first case. Again, we have an unparseable sentence; the additional difficulty is that we do not know which of the words is in error, since they are all in the lexicon. If we postulate only a single error, it is feasible to attack the problem by brute force: for each word in turn, we make a plausible substitution, and attempt to parse the sentence. For an n-word sentence, attempting m substitutions for each word, this requires mn parsing attempts; n is rarely likely to exceed 15 or so, and m can be made as small as one or two, so this is not an excessive number.

The third case—a syntactically correct but semantically implausible sentence—is more difficult. Te Kaitito's semantic knowledge is currently limited to the current discourse context; thus, it might be possible to do something about mistakes such as

(6.3)
User	I saw some fruit flies.
System	okay
User	The fruit fries like a banana.

but even this would not be easy: as described in Section 2.1.2, Te Kaitito accommodates new definite NPs. At present only the number of accommodations is counted for disambiguation purposes, and it is not clear how to define a more sophisticated metric

for the plausibility of an accommodation. It seems that such a metric would require a significant amount of world-knowledge and 'common-sense' reasoning: were a friend to write to me that he had been 'attacked by a doge on the way home', I would probably assume—drawing on a number of beliefs and inferences—that he meant a dog rather than the chief magistrate of Venice. Such deductions are, unfortunately, beyond the current scope of Te Kaitito.

Bever (1970) gives another, more compelling example of a perfectly valid sentence which human listeners nevertheless prefer to perturb:

(6.4) The horse raced past the barn fell.

Although this can be interpreted without modification as 'the horse [that was] raced past the barn fell', most people prefer to interpret it as an incorrect form of 'The horse raced past the barn and fell'. In this case, however, the perturbation occurs at the syntactic level rather than the character level, and is thus beyond the scope of the current work.

6.6 An example

Here is an example of an utterance with two type-1 errors (that is, errors resulting in non-words), and a discussion of how it might be corrected.

(6.5)

User	I went to a bar
System	okay
User	The bae was pretty bae

The spell-checker aspell (Atkinson, 2005) produces 41 candidate replacements for the misspelling *bae*, of which the following subset are in the ERG's lexicon: *bar*, *be*, *base*, *bare*, *bad*. This makes for a total of 25 possible perturbations, consisting of every possible combination of each of the five replacements for each misspelling. Of these twenty-five, only four result in valid parses:

1. The bar was pretty bare
2. The bar was pretty bad
3. The base was pretty bare
4. The base was pretty bad

Of these four, the accommodation module should be able to weed out 3 and 4, since there is no base in the current discourse context. The other two interpretations are left to the lower-level disambiguation modules: perhaps the probabilistic grammar will be sufficiently confident to dispose of one of them, or perhaps a clarification question will be required.

6.7 Perturbing for specific error classes

Many simple errors are common to large numbers of language learners. We can program the perturbation module to look for these specific errors. For example, it has been found that, among first-year university students of Māori, 15% of errors involve missing, spurious or incorrectly placed macrons (Earnshaw, Fleming, Weatherall, and Knott, 2004). In this case, we can process every word in an utterance independently, and perform a dictionary lookup for words which are identical save in the placement of macrons. In fact, the results of Earnshaw et al. (2004) seem to show a distribution similar to that found by Zipf (1949) for word frequency: when error classes are ranked in descending order of the frequency of their occurrence, and when the frequencies are plotted against this rank, the resulting slope has the shape of a reciprocal function $y = c/x$ for some constant c. In short, a few very common error classes account for a large proportion of the total collection of errors. This implies that by making specific perturbations for only a few types of error it would be possible to accommodate a large proportion of the total errors made. Of course, some of these errors might require perturbation at the syntactic rather than the character level, but character-level perturbation for specific errors would nevertheless offer a large amount of error tolerance for a relatively small cost in implementation and processing.

Chapter 7

Implementation and Results

> The result was not altogether the success that Harris had anticipated. There seemed so little to show for the business. Six eggs had gone into the frying-pan, and all that came out was a teaspoonful of burnt and unappetizing looking mess.
>
> —Jerome K. Jerome, *Three Men in a Boat* (Jerome, 1889)

In this chapter, I will describe the details involved in implementing many of the ideas I have described in the preceding chapters, and reproduce dialogue transcripts showing the results of this implementation.

It is difficult to effectively demonstrate disambiguation by simply showing dialogue transcripts: after all, a fully functional disambiguation system should be entirely invisible, automatically and silently selecting the correct interpretation every time. Therefore I have adopted various techniques in order to produce visible demonstrations of the disambiguation modules working. I have instrumented the dialogue system with diagnostic routines able to print out the verdicts of the modules; and in some cases I have disabled other disambiguation modules to more clearly show a single module in operation. The highly modular integration architecture makes this easy to do: the disambiguation pipeline is simply a global variable holding a list of functions, and can be altered at any point—even during a dialogue—to add, remove, or reorder modules. When demonstrating clarification questions, I simply disable *all* the disambiguation modules. This chapter, then, is not intended to present a thoroughgoing evaluation of the disambiguation system *in toto*; this kind of evaluation needs to be conducted through structured user trials—which, though planned for Te Kaitito, have not commenced at the time of writing. The focus in presenting results is to show—within artificially constructed dialogues—the operation of the individual modules within the context of the disambiguation framework and of the Te Kaitito system as a whole.

Unfortunately, due to time constraints, it was not possible to implement all the features proposed in the preceding chapters; in particular, error tolerance by perturbation (Chapter 6) has not been implemented at all.

This chapter first briefly describes some general implementation details (Section 7.1) and gives an account of the construction of the disambiguation architecture (Section 7.2). In Section 7.3, I describe the probabilistic parsing module, its implementation, and its performance both with and without contextual augmentation. Section 7.4 describes other disambiguation modules, including the implementation and performance of the accommodational-weight and presuppositional-weight modules. Finally, Section 7.5 gives implementation details and results for clarification subdialogues.

7.1 General implementation details

The Te Kaitito system, like the LKB system it uses as its grammar engine, is written in Common Lisp (Steele, 1990). For the purposes of this thesis the system was run on CMU Common Lisp (MacLachlan, 1992) version 18e, though it can also be run on Allegro Common Lisp. With a few minor exceptions, the work described in this thesis was also implemented in Common Lisp and integrated with the existing Te Kaitito codebase.

7.2 Integrating information sources

The integration algorithm, simple in conception, was also fairly straightforward in implementation. The disambiguation data was stored within the candidate interpretations, which are represented as tree-like structures to avoid replication of data. The data structures are similar to the 'ambiguity tree' shown in Figure 3.1 on page 29: higher-level representations branch off the lower-level ones.

Each disambiguation module was implemented as a function which takes a list of interpretations and returns another list of interpretations (which is, of course, expected to be a subset of the input list). The threshold for removing or keeping a parse is set within the disambiguation module itself, and modules are implemented so as never to remove all the interpretations.

Thus the integration code actually does very little: it simply starts with the full set of interpretations, passes them sequentially through the disambiguation modules in order of descending representational level, and stops when no modules remain or when

only one parse remains.

7.3 Statistical parsing

As described in Section 2.3, a statistical grammar has a set of attributes on LKB syntactic and morphological rules, and assigns each a probability. The main requirement for the building of a statistical grammar on top of the existing grammar is a source of data from which to infer these probabilities. As mentioned in Section 2.3.3, there is a treebank available for the ERG: the Redwoods treebank (Oepen et al., 2002) is a corpus of sentences assembled for the Verbmobil project (Wahlster, 2000), with ERG parses and human-generated annotations showing the correct parses.

There is another, longer-term advantage to basing the probabilistic grammar on the Redwoods treebank: Redwoods is developed using [incr tsdb()][1] (Oepen, 1999), an open-source grammar development environment, and Redwoods data is published in a format produced by [incr tsdb()]. [incr tsdb()] is useful to the Te Kaitito project for several reasons:

- It provides an integrated environment for the development and maintenance of a treebank. Since it is planned to develop a treebank for Te Kaitito's custom-written Māori-English Grammar (MEG), these facilities would be extremely useful. [incr tsdb()]'s open-source licence not only makes these facilities available at no monetary cost, but gives the freedom to modify its source code should extra capabilities prove necessary. (For example, as mentioned in Section 8.2.5, we will probably wish to add annotations at the dialogue-act level.)

- [incr tsdb()] provides a user-friendly parse annotation environment, allowing human annotators to select preferences for parses in the treebank with relatively little knowledge of the technicalities of grammar writing. This speeds up the process of producing a correctly parsed treebank by increasing the number of people able to work on annotation.

- [incr tsdb()] stores the annotators' parse preferences as first-class data. This makes it easier to update the treebank to keep pace with development of the grammar (which is important since the MEG is currently under heavy development). When the grammar is updated, the corpus is parsed using the new

[1] Pronounced, according to the manual, 'tee ess dee bee plus plus'.

grammar; the annotation process can then be 'replayed' using the stored preference data. Of course, this cannot cope with sentences for which the set of available parses has changed significantly; but it can greatly reduce re-annotation effort by automating the process when the new parses are isomorphic to the old.

- There are sophisticated facilities for profiling a grammar's performance. The treebank can thus do double duty as a source of statistical information for training a probabilistic grammar, and as a test suite to assess and improve the quality of the MEG's implementation.

For the initial, ERG-based implementation described in this thesis, [incr tsdb()] is not necessary; however, since published Redwoods data is in the form of [incr tsdb()] output, a system developed to use Redwoods data will work with any treebank produced using [incr tsdb()].

The initial implementation of the probabilistic grammar was as a straightforward PCFG with parameters inferred directly from counts of productions and their heads. There were several reasons for choosing such a simple implementation; some, relating mainly to the goals of the project, are explained in Section 4.1.1. From the point of view of implementation, the main consideration was the planned integration of dynamically updated probabilities: it seemed most reasonable—in terms of maintaining a coherent probability model—to integrate the static and dynamic parts of the grammar at the level of raw production counts. Thus a simple parameter estimation procedure was desirable, both from the point of view of speed (since parameter estimation would have to be done on-line) and ease of comprehension and modification.

7.3.1 The global grammar

I use the term 'global' to refer to the main, static portion of the grammar, with parameters inferred from a treebank, in contrast to the more context-sensitive components with parameters inferred from the unfolding dialogue.

Training a grammar from the Redwoods treebank

In theory, inferring PCFG parameters from the parsed, annotated sentences of the Redwoods treebank should be straightforward; in practice, several difficulties were encountered.

Each release of Redwoods data (or 'growth', as they are termed) consists of sentences parsed with the ERG version current at the time of the release. Due to the

swiftly evolving nature of the ERG, the parses themselves are of limited utility for any other ERG version.

As previously mentioned, the Redwoods treebank stores the human annotator's parse preferences as first-class data, allowing the same annotation process to be re-run on a different version of the grammar. One motivation for this is to alleviate the problem of matching the treebank to the grammar. Actually performing this procedure can, however, be difficult: changes in the grammar can be large enough that the stored parses are no longer relevant, so that manual intervention is once more required. More seriously for my purposes, the parsing process can require large amounts of memory. It proved impossible to re-parse any of the Redwoods corpora, since no computer available to me at the University of Otago had sufficient memory to do so.

The path of least resistance thus appeared to be that of modifying Te Kaitito itself to work with an ERG version for which a pre-parsed Redwoods corpus was available. The Redwoods third growth (Toutanova et al., 2003) and the version of the ERG bundled with it (dated October 2002) was chosen as offering the best compromise between amount of data and closeness to Te Kaitito's previously ERG version, a release dated 21 December 2003. Nevertheless, the two ERG versions differed significantly and the modifications proved fairly labour-intensive.

Another, unexpected, problem was encountered while processing the treebank: even the version of the ERG distributed with the Redwoods third growth did not correspond exactly to the one evidently used to parse the corpus. Around a dozen lexical items and several type definitions used in the treebank, accounting for several hundred rule applications, were missing from the grammar. Most of these were added by hand with guidance from their definition in later versions of the ERG.

With these preliminaries dealt with, it was relatively straightforward to make use of the Redwoods parse data. Each Redwoods growth is divided into a few sub-corpora, each consisting of a set of data files produced by [incr tsdb()]. Of these, three were of interest here: `result`, containing a derivation tree for every possible parse of every sentence in the sub-corpus (together with various other information); `preference`, containing annotator-supplied indications as to which is the preferred parse; and `parse`, containing additional information about parses—most pertinently, an indication of whether a particular sentence caused an error during parsing, allowing us to exclude such sentences from consideration. Figure 7.1 shows a sample derivation tree from the Redwoods treebank. I wrote a short script in Perl (Wall, Christiansen, and Orwant, 2000) to read these three files and produce a single file containing only valid, preferred

parses. The parses in `result` were stored as bracketed S-expressions and were thus easy to read from Lisp, but the extra information, and the other files, were in a character-delimited format better suited to Perl's string manipulation capabilities—hence the preprocessing step.

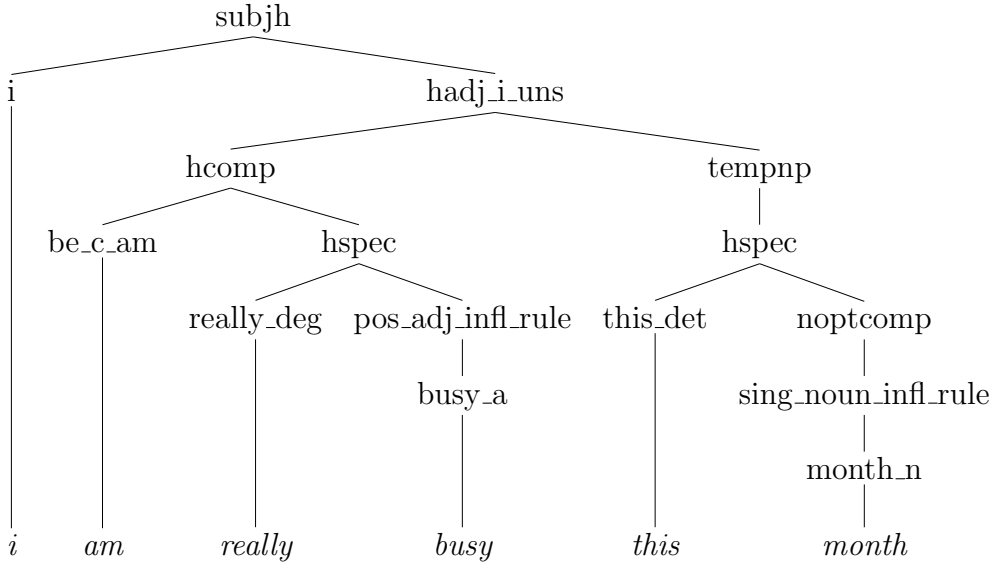

Figure 7.1: A sample derivation tree from the Redwoods treebank (third growth, vm6 sub-corpus, item 697, parse 2).

For the grammar itself a top-down probability model was used. Three hash tables of counts were built up from the treebank: `roots`, counting the number of times each rule was used as the head of a sentence; `heads`, counting the number of times each rule was used; and `productions`, counting the number of instances of each parent-child-child triplet in the derivation tree (all the ERG's rules are binary). The probability of any new sentence can then be calculated from these three sets of data. Dividing the count of the root rule by the total of the counts in `roots` gives the probability of that rule occurring at the root of a parse tree. Then, for every production in the tree, the triplet's entry in `productions` is divided by the head's entry in `heads` to give the probability of that production occurring from that head. Multiplying all these quantities (or, in practice, adding their logarithms) produces a probability for the sentence.

I implemented few of the refinements possible in a statistical grammar, though most of these could be added without too much disruption of the current architecture; they are discussed in Section 8.2.2. In fact, the production probabilities were not even computed and logarithmized in advance; this was for easier integration of the dynamic grammar, as will be described in the next section. Smoothing was of the simple add-one

type (see e.g. Chen and Goodman, 1996).

The global grammar in action

In order to show the operation of the grammar, I instrumented the single-rephrase clarification module to show the scores assigned by the grammar, and configured the disambiguation pipeline not to reject any interpretations—that is, to ask for full clarification in every case. As mentioned in Section 5.3.11, single-rephrase clarification is very useful for showing parse structures as user-friendly unambiguous rephrasings, in the cases where entirely unambiguous rephrasings can be found. After the rephrasing, the clarification module shows the score assigned by the probabilistic grammar as a logarithm of the sentence's probability. Hence a higher number indicates a preference for a particular interpretation.

```
> i ate lunch at the office

What do you mean?
1: at the office i ate lunch
   pcfg -22.90
2: lunch at the office i ate
   pcfg -23.44
```

The probabilistic grammar prefers the high attachment of the PP, which fits with the intuitively correct reading that it was the eating that occurred at the office. The next example is the utterance 'John is really angry', in which 'really' should be interpreted as an adverb of degree, modifying 'angry', rather than an adverb attached to the verb and indicating that John is genuinely angry.

```
> John is really angry

What do you mean?
1: john is being really angry
   pcfg -21.30
2: john is really being angry
   pcfg -24.05
```

Once more, the grammar selects the correct interpretation.

Here is a sentence with a more fundamental ambiguity: 'The patient authors lunch'. The more intuitive reading is that some long-suffering writers are taking their midday

meal; however, the ERG includes the word *author* as a verb, and thus can also interpret this sentence as being about an ill person 'authoring' a meal, whatever that might mean.

```
> the patient authors lunch

What do you mean?
1: lunch the patient authors
   pcfg -17.30
2: Something else.
   pcfg -16.36
```

Once more, the intuitively correct reading is selected. In this case, however, there was no unambiguous rephrasing available for the correct reading, so it is presented as a 'something else' choice.

7.3.2 The contextually augmented grammar

Implementation

Implementing the contextual augmentation described in Section 4.1.2 required thoroughgoing but fairly straightforward modifications to the grammar's data structures and algorithms. First, hash tables were set up parallelling exactly those constructed for the global grammar. Then the sentence scoring routines were modified to make use of these new data sources: wherever they previously read a count from the single global hash table, they now read both a global and contextual count. These are combined by linearly scaled addition, assigning a higher weight to the contextual count to reflect the fact that it is likely to be more relevant to the current context, and to compensate for the fact that it is derived from a far smaller sample. Weighted linear combination does not destroy the probability model.[2] In effect, it is equivalent to the following procedure:

For each set of weights to be combined
    For each sentence used to build the table of counts
        Duplicate the sentence a number of times equal to the scaling factor
        Add all the duplicates to a new treebank
Infer probabilities from the new treebank

Although this algorithm is hardly a sensible basis for a practical implementation, it can be seen that it maintains a consistent probability model, and produces the same results as weighted linear combination.

[2]Of course, as described in Section 2.3.3, the probability model itself is somewhat inappropriate.

Next, a new **grammar update** module was added to the end of Te Kaitito's interpretation pipeline to maintain the values for the local grammar. This module first multiplies all the counts in the local grammar's tables by a damping factor; again, since the same linear scale factor is applied to both head and production counts, the probability model is not disrupted. The damping factor, as described in Section 4.1.2, is applied to allow the grammar parameters to adapt as the focus of the dialogue changes.

To see how the probability model is affected by context-augmentation, suppose that we have a global grammar induced from a collection of parses G, and that during combination, the local grammar is given a weight of 32 and the global grammar a weight of 1. Suppose also that the damping factor is one-half, and that we have just incorporated p_n, the disambiguated parse of the dialogue's nth utterance, into the local grammar. Then the current combined grammar is equal to equivalent to a normal, static PCFG induced from the collection

$$G + 32 \times p_n + 16 \times p_{n-1} + 8 \times p_{n-2} + \cdots + 1 \times p_{n-5} + \cdots + 2^{5-n} \times p_0$$

where the notation '$+\ m \times p$' corresponds to adding m instances of the parse p to the collection. In practice it is of course impossible to use fractions of a sentence in a treebank; however, identical probabilities would be produced if every sentence in the combined treebank were duplicated 2^{n-5} times to give integral values, so it is clear that fractional sentence counts do not disrupt the probability model.

This module then takes the single interpretation of the current utterance (which by this stage of the pipeline has been fully disambiguated and clarified), and counts its root, heads, and productions in the same way as the global grammar counts those of a parse from the treebank. Finally, it passes the dialogue state on, unchanged, to the dialogue engine module (see Figure 2.1), which will construct an appropriate response.

Results

I now repeat the 'patient authors lunch' example, but precede it with two sentences designed to cue the interpretation which would normally be considered less likely.

```
> the doctor authors a report
okay

> the secretary authors a book
okay
```

```
> the patient authors lunch

What do you mean?
1: lunch the patient authors
   pcfg -15.99
2: Something else.
   pcfg -16.60
```

The preceding utterances with isomorphic parses have resulted in a slight preference for the *authors-as-verb* interpretation. Next, I will demonstrate the same mechanism operating on a slightly less contrived example.

To distinguish between people with identical names, it is fairly common to append a PP—for example,

(7.1) A: I was talking to John yesterday.
B: Which John?
A: John at the office.

If both the speakers are used to disambiguating instances of 'John' with such PPs, one might expect a preference for low attachment in a phrase such as 'I saw John at the office', even though attachment of 'at the office' to the verb would more usually be the preferred interpretation. The following example demonstrates this effect.

As a control, I first give Te Kaitito the test sentence without any context:

```
> i saw john at the office yesterday

What do you mean?
1: i saw john yesterday at the office
  pcfg -31.28
2: john at the office i saw yesterday
  pcfg -34.19
```

As expected, the high attachment is preferred. Next, I try introducing two PP-disambiguated Johns before uttering the test sentence:

```
> john at the office works
  pcfg -25.49
```

```
okay

> john at the bank drinks
  pcfg -16.85

okay

> i saw john at the office yesterday

What do you mean?
1: i saw john yesterday at the office
   pcfg -30.13
2: john at the office i saw yesterday
   pcfg -28.06
```

The low PP attachment is successfully cued.

7.3.3 Idiolect-sensitive parsing

The idiolect-sensitive grammar was implemented in a very similar way to the contextually augmented grammar: parallel hash tables were added to store counts of heads and rules.[3] The grammar update model was modified to add its counts to the idiolect tables as well as the local tables—the difference in this case being that a user's idiolect table is *only* updated using data from utterances made by that user, rather than (as for the contextual tables) every utterance made during a dialogue.

The PCFG parse ranker was similarly modified to make use of the idiolect tables as well as the global and local tables; all the counts are combined by a linearly weighted sum, as described above. In combining counts, the idiolect component is weighted less than the local component, in keeping with the second prioritization rule of Section 3.1.2. Also, since idiolects change more slowly than dialogue contexts, a smaller amount of damping is applied.

Unfortunately, idiolect-sensitive parsing is hard to demonstrate with short examples: to demonstrate the effect it would be necessary to enlist users for long-term trials with Te Kaitito and compare disambiguation performance on the same utterances using their idiolect-sensitive grammars. In my case, there was neither the time nor the

[3]Since the version of Te Kaitito used in this implementation only supports a single human user at a time, it is not yet necessary to use multiple idiolect tables simultaneously.

user base to do this: Te Kaitito user trials have not yet commenced at the time of writing. However, since the basic mechanism of idiolect-sensitive parsing—skewing of grammar parameters according to utterances seen within the dialogue itself—is identical to that of contextually-augmented parsing, the results of the previous section are very promising for idiolect-sensitive parsing. With the development of Te Kaitito to support multi-agent dialogue (Knott, Bayard, and Vlugter, 2004), idiolect-sensitive parsing should become more useful and more demonstrable.

7.4 Other modules

The modules for distinguishing between questions and clarification questions (Section 4.3.2) and disambiguating using ambiguity level (Section 4.3.4) have not yet been implemented. In the following sections I describe those modules which have been implemented.

7.4.1 Accommodation and presuppositional weight

Implementation

These modules were fairly straightforward to implement. The DRS structures produced by the semantic interpretation module separate the assertion of the sentence from its presupposition. To count them, I wrote a recursive depth-first tree-walking algorithm which went through an utterance's presupposition list, descending into any nested presuppositions and counting the number of presuppositions made at each level. Summing these values throughout the tree gives the total presuppositional weight for an utterance.

Accommodation weight was calculated in an isomorphic fashion: the semantic module annotates each presupposition-set with the number of presuppositions which had to be accommodated, so in this case the algorithm simply had to recurse through the presupposition tree calculating the total of these values.

Results: accommodation

To demonstrate accommodation, I instrumented Te Kaitito to show accommodation scores and unique rephrasings when disambiguating.

```
> i saw the girl with the telescope

 1 with the telescope i saw the girl
   accommodations 2
 2 the girl with the telescope i saw
   accommodations 3

 choosing 1

okay
```

Here, the *girl-with-telescope* parse requires three accommodations (girl, telescope, and girl-with-telescope), versus two accommodations (girl and telescope) for the *saw-with-telescope* parse (since there are no girls or telescopes in the context), so the latter is selected.

Results: presuppositional weight

```
> a red girl with a telescope walks
okay

> a blue girl sleeps
okay

> i saw the girl with the telescope
okay

> which girl did i see

you saw the red girl
```

Here, the red-girl interpretation is preferred, because—in addition to matching the *girl* relation—it involves matching the *girl-with-telescope* relation to the context, resulting in a higher presuppositional weight.

7.4.2 Saliency-based referent disambiguation

I was not involved in the implementation of this module, so I merely give an example of it in operation here:

```
> a black cat sleeps
okay

> a white cat eats
okay

> a dog chases the cat
okay

> which cat does the dog chase
the dog chases the white cat
```

When the ambiguous utterance is made, the white cat has been more recently mentioned, and is hence more salient. The ambiguous term 'the cat' is thus bound to the white cat.

7.5 Clarification subdialogues

Clarification, like disambiguation, is hard to demonstrate concisely within its proper context of a full interpretation-disambiguation-clarification pipeline: if the disambiguation modules are working as they should, clarification will seldom be required. Thus, in order to generate the examples for this section, I have disabled the disambiguation modules.

7.5.1 Framework

Te Kaitito's architecture includes a versatile facility for incorporating subdialogues at any stage of the processing pipeline, independently of whether the console or web interface is being used. The structure of the interpretation pipeline is one of functional composition: a large data structure representing the entire dialogue state is passed through the pipeline from one module to the next, each module transforming it to the next required form. However, as well as the dialogue state, each pipeline module returns a **continuation command**, specifying what action should be taken next. The continuation command can take one of three values:

`pass` is the most commonly used continuation, and corresponds to the normal operation of the pipeline: it simply stipulates that the module's output should be passed straight to the input of the following module.

`clear` is used when processing fails in some way, for example when the syntactic interpretation module fails to find any valid parses. This command breaks out of the pipeline, aborts the interpretation process, and presents a supplied message to the the user (for example, 'I don't know the word "boustrophedon".').

`subdialogue` initiates a subdialogue feeding back into the current module: a supplied message is presented to the user through whichever front-end is in use. The user's response is collected and returned—as part of the monolithic dialogue data structure—to the module which initiated the subdialogue. This cycle can be repeated an arbitrary number of times.

The subdialogue facility has already been used in van Schagen (2004) to allow interactive expansion of the lexicon during a dialogue,[4] and it is also planned to use it in error-correcting teaching subdialogues (Slabbers, in preparation).

Figure 7.2 shows a part of the pipeline, with these commands, as a finite state machine.

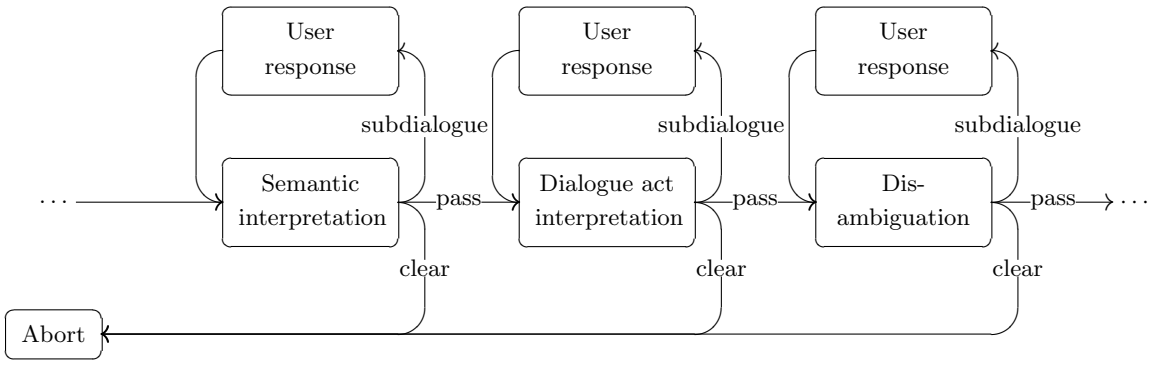

Figure 7.2: A fragment of the interpretation pipeline as a finite state machine, showing the effects of the three continuation commands.

To conduct a clarification subdialogue, then, it is simply necessary for the disambiguation module to return `subdialogue` for every response which forms part of the clarification, relying on the system to call it back with the user's response. Once the clarification subdialogue has finished, the disambiguation module replaces the ambiguous, underspecified interpretation structure in the monolithic dialogue structure with a single, fully disambiguated interpretation, and returns `pass` to pass this interpretation on to the dialogue engine.

[4]This facility has not yet been integrated with the work described in this thesis.

7.5.2 Implementation

The formal presentation of the techniques developed in Chapter 5 means that they translate relatively directly into code; most of the implementation involved constructing and manipulating representations of the various sets involved in Chapter 5's derivations. Section 5.3.5 and 5.3.6 describe most of the strategy-independent aspects of implementing rephrasing-based clarification; further implementation details are presented along with the results below.

7.5.3 Single-rephrase clarification

I will give an example using the Māori-English Grammar, which allows ambiguity in one language to be resolved by rephrasing into a different language. In this case the ambiguous utterance is the following Māori sentence

(7.2) Kei te kai te tangata.

with the following two interpretations.[5]

(7.3) Kei te kai te tangata.
[present progressive] eat the person
The person is eating.

(7.4) Kei te kai te tangata.
[present locative] the food the person
The person is at the food.

No special provision is required to activate bilingual rephrasing: if the MEG is loaded, a request to the ERG's generation module to generate sentences from a MRS form will naturally produce every possible sentence in both languages, since the *language* feature has not been constrained. In this case, the clarification module has been configured to show every possible rephrasing for a candidate syntactic structure, rather than picking one using the probabilistic grammar (which does not yet work with the MEG due to a lack of training data).

```
> kei te kai te tangata
```

[5] In presenting English glosses to make the interpretations clear, I am of course performing bilingual rephrasing-based clarification for the reader.

```
What do you mean?
1: the person is at the food
   kai te kai te tangata
2: the person is eating
```

The clarification module produces two unambiguous English rephrasings, and one unambiguous rephrasing in Māori (effected by changing the tense and aspect marker *kei* to *kai*).

7.5.4 Interactive binary clarification

One of the advantages of interactive binary clarification, compared to single-rephrase clarification, is its ability to rapidly pare down a large set of readings. In the best case, where a perfectly even split is found at every stage, it is possible to select one parse out of n using only $\lceil \log_2 n \rceil$ binary choices. In an attempt to test this strategy on a suitably large set of rephrasings, I concocted a sentence with a high degree of attachment ambiguity: 'I ate lunch at the office on the floor under my desk'. I instrumented the clarification routines to produce output detailing their operations; in this example I have elided some of it for brevity.

```
> i ate lunch at the office on the floor under my desk

**** we now have 14 interpretations ****

Removing (my desk i ate lunch under on the floor at the office)
Removing (under my desk i ate lunch on the floor at the office)
Removing (under my desk i ate lunch at the office on the floor)
Removing (lunch i ate on the floor at the office under my desk)
Removing (lunch i ate under my desk on the floor at the office)
Removing (lunch i ate under my desk at the office on the floor)
[ 22 more rephrasings removed ]
```

The clarification module reports that it has fourteen interpretations to choose between, generates a complete set of rephrasings, and begins to remove rephrasings which are equivalent (in the sense of Section 5.3.3) to other rephrasings, eventually pruning each equivalence class down to a single rephrasing. The probabilistic grammar is used to select a most 'natural' rephrasing from each equivalence class, which is kept when all the others are removed.

Next, the clarification module must choose an initial question to ask. I have configured it, in this case, to print out a list of the alternatives under consideration:

```
(my desk i ate lunch on the floor under at the office) 1 6
(my desk i ate lunch at the office under on the floor) 2 5
(my desk i ate lunch under at the office on the floor) 2 5
(my desk i ate lunch on the floor at the office under) 2 5
(my desk i ate lunch at the office on the floor under) 5 2
(lunch i ate on the floor under my desk at the office) 2 5
(lunch i ate at the office on the floor under my desk) 5 2
[ 21 more rephrasings considered ]

Did you mean 1 (i ate lunch at the office under my desk on the floor)
 or 2 (something else)?

> 1
```

The number printed beside the considered rephrasings require some explanation: the first is the number of candidate syntactic structures corresponding to the rephrasing. The second is a measure of *asymmetry*, calculated as the difference between the number of structures associated with the rephrasing and half the total number of structures. In this case the total number of structures is 14. So, for example, the first rephrasing, realizing only a single structure, has a high asymmetry value of 6. The clarification module picks one rephrasing by the following criteria:

1. As described in Section 5.3.13, it first looks for as even a split as possible—that is, it seeks a rephrasing with the lowest asymmetry.

2. If there are multiple such rephrasings, it picks (if possible) one which has a converse (as defined in Section 5.3.4) so as to be able to ask a 'this or that' question rather than a 'yes or no' question.

3. If there are still multiple rephrasings satisfying these criteria, it picks the one with the highest score from the probabilistic grammar (along with the highest-scoring converse, if a converse exists).

The use of the probabilistic grammar as a last-resort selection technique increases the likelihood of a fairly natural-sounding rephrasing being presented to the user (as in this case). Note that a maximally symmetrical rephrasing with a converse could not be found, forcing the system to resort to the 'something else' question.

```
**** we now have 5 interpretations ****

(my desk i ate lunch on the floor under at the office) 1 3/2
(my desk i ate lunch at the office under on the floor) 2 1/2
(lunch i ate at the office on the floor under my desk) 2 1/2
(i ate lunch on the floor at the office under my desk) 2 1/2
(i ate lunch at the office on the floor under my desk) 5 5/2
(at the office on the floor under my desk i ate lunch) 1 3/2
(lunch at the office on the floor under my desk i ate) 2 1/2
(lunch at the office i ate on the floor under my desk) 1 3/2
(lunch on the floor at the office under my desk i ate) 1 3/2

Did you mean 1 (i ate lunch on the floor at the office under my desk)
 or 2 (something else)?

> 1
```

After a single clarification act by the user, just five candidate interpretations remain (this low figure is partly due to luck: the other choice would of course have led to nine interpretations). Once more, the available rephrasings are scored according to how symmetrically they split the space of interpretations. Note that, among the four interpretations with the minimal asymmetry score of one-half, the system has chosen the one which sounds most natural ('i ate lunch on the floor...' rather than, say, 'my desk i ate lunch...') using the probabilistic grammar. Again, no converse can be found, so the 'something else' option must be used.

```
**** we now have 2 interpretations ****

(my desk i ate lunch on the floor under at the office) 1 0
 converse: (lunch on the floor at the office under my desk i ate)
(i ate lunch on the floor under my desk at the office) 2 1
(lunch on the floor at the office under my desk i ate) 1 0
 converse: (my desk i ate lunch on the floor under at the office)

Did you mean 1 (my desk i ate lunch on the floor under at the office)
or 2 (lunch on the floor at the office under my desk i ate)?

> 1
```

```
**** we now have 1 interpretation ****

okay
```

Now only two interpretations remain, with three possible rephrasings of them. Here, the system is finally able to locate a converse; where a converse to a rephrasing is found, it is shown after the rephrasing itself. Thus each rephrasing in the pair is shown twice: once during its own evaluation, and once during the evaluation of its converse. The system uses the converse rephrasings to present the user with an 'either-or' question explicitly showing *both* alternatives.

Once the user selects a rephrasing, the utterance is at last fully disambiguated, and the system delivers its standard 'okay' response indicating that it has, at last, successfully interpreted the user's original utterance.

7.5.5 Semantic clarification

Since I was not involved in the implementation of saliency-based semantic clarification, I simply give an example here of its operation.

```
> i have a red telescope
okay

> i have a green telescope
okay

> i saw a girl with the telescope

What do you mean?
1: with the telescope i saw a girl
2: a girl with the telescope i saw

> 1

which telescope did you see a girl with
```

This is an example of disambiguating an utterance exhibiting both syntactic and semantic ambiguity; in accordance with the principle laid down at the star of Chapter 5, the utterance is clarified first syntactically, then semantically.

Chapter 8

Conclusions and further work

> And it is understood that there is added to the general consideration of the whole matter, the consideration what is greater than and what is less than, and what is like the affair which is under discussion, and what is equally important with it, and what is contrary to it, and what is negatively opposed to it, and the whole classification of the affair, and the divisions of it, and the ultimate result.
>
> —Marcus Tullius Cicero, Treatise on Rhetorical Invention. i. 18 (Cicero, 1903)

8.1 Conclusions

8.1.1 Integrating disambiguation data

The main goal of this project has been the construction of a comprehensive framework for classifying and integrating disambiguation data, and the implementation of a disambiguation process based on this framework. The architecture devised in this thesis and described in Chapter 3 encompasses both a two-dimensional classification of disambiguation data sources, and, derived from this classification, heuristics for use when combining data sources.

On a theoretical level, the classification provides a useful framework within which to discuss and assess sources of disambiguation data: both the classification metrics—the *representational level* corresponding to a point in the processing pipeline, and the *domain*, corresponding to a particular area or body of knowledge—are grounded in readily comprehensible and measurable properties, so virtually any source of disambiguation data can be located fairly clearly within the framework. My data prioritization heuristics are couched in terms of this classification framework, so while I have ultimately

had to appeal to intuitive notions of 'correct' interpretations—which is probably inevitable when proposing disambiguation techniques—I have at least made my reasoning explicit.

For implementation purposes, it is useful that data sources are classified with respect to practically grounded measures: it removes some of the trial-and-error from the process of integrating a new data source into the framework. Because it is generally clear how a particular data source should be classified, it is also clear (from the prioritization heuristics) how it should be incorporated into a disambiguation algorithm.

This principled and procedural approach also brings benefits when tuning the performance of the system. The way in which different data sources are combined is significantly constrained by their classification within the theoretical framework, so—while there are some tunable parameters—there is little chance of an attempted implementation becoming bogged down in a huge search space of interdependent parameter weights. As described below, there is some leeway within the heuristics, and there is no well-defined method for resolving conflict *between* the two heuristics, but it should nevertheless be a fairly quick job to classify and integrate a new module into the disambiguation pipeline.

Tuning the system is also made easier by another feature of the architecture: because the constraints of the different disambiguation modules are applied sequentially rather than simultaneously, the modules are rather loosely coupled: any module can operate independently of any of the others, and the parameters for any module can be set independently of the others. What is more, modules can be added to, removed from or rearranged within the disambiguation pipeline without even interrupting the current dialogue, which is a boon for development of the disambiguation procedure.

8.1.2 Clarification subdialogues

There are two main aspects to the treatment of clarification subdialogues in this thesis. The first, forming a relatively distinct subproject, was the theoretical framework built up for the systematic treatment of syntactic clarification by rephrasing. The second and (at least for Te Kaitito) more important aspect is the theoretical and practical integration of clarification subdialogues into the framework developed for disambiguation.

The theoretical groundwork of Section 5.3.1 provides a solid basis for discussion of syntactic rephrasing-based clarification in the rest of Section 5.3. This basis will be of use in any further development of syntactic clarification: it formalizes such desirable

qualities as precision and convenience and allows the discussion of aspects of rephrasing-based clarification in concrete terms. The theoretical framework is also general enough to have wider applicability to virtually any situation where an entity must be specified indirectly via some arbitrary relation.

8.1.3 Error handling

Chapter 6 successfully showed how the detection and correction of user errors can be incorporated into the disambiguation framework. The interesting result here was the usefulness of the disambiguation pipeline in error handling: the proposed techniques involved fairly basic operations of perturbation and ranking at the character level, but attaching the module to the rest of the interpretation pipeline allowed it to take advantage, indirectly, of information right up to the dialogue act level.

8.1.4 Integrated natural language processing

The reciprocal benefits of integrating a new module into a large, integrated system like Te Kaitito were apparent in implementing error handling, but were also visible throughout the project: for example, the stochastic grammar, once implemented, could be used to rank generated sentences as well as interpreted ones—which then, in turn, permits the grammar writer to include rare forms specifically for use in syntactic clarification (see Section 5.3.7) without fear of them being used in normal dialogue. In general, adding a new module to a system like Te Kaitito has potential benefits proportional to the number of modules already in the system—it might be able to interact usefully with any of the pre-existing modules. Thus, for the same amount of work, implementing a technique within an existing system provides far more benefit than implementing it in isolation.

8.2 Further work

8.2.1 Integration

The main area of uncertainty in my architecture for the classification and combination of data sources is how the two classification metrics should be combined with respect to each other. In this thesis, it was not necessary to address this question thoroughly, since Te Kaitito currently uses only a subset of the data sources covered by the taxonomy,

resulting in few potential points of conflict between the heuristics.

The work on accommodation of user errors was a relatively late addition to the project, and although it was successfully incorporated into the disambiguation framework, it would be pleasing—at least on a theoretical level—to try to develop a more unified framework which places errors and ambiguity on the same footing, in a similar fashion to Menzel and Schröder (1999). The successful theoretical integration of character-level error tolerance into the pipeline suggests that the theoretical framework can be expanded to accommodate errors in a more general way. This would probably cause few changes in the current implementation, but might form a useful guide when attempting to integrate higher levels of error tolerance into the interpretation process.

8.2.2 Probabilistic parsing

Compared to the state of the art, the current probabilistic grammar is extremely primitive; as mentioned in Section 7.3, the goal here was not *per se* to produce a high-performance grammar. The two main considerations were to integrate a probabilistic grammar with the existing structure of Te Kaitito and with the new disambiguation architecture, and to explore techniques for improving the grammar's context-sensitivity in ways particularly suited to the environment of a dialogue system.

With the basic structure now in place, it would clearly be desirable to improve the performance of the grammar itself. Fortunately, there has been much recent theoretical and practical work on parameter estimation for stochastic attribute-value grammars. We might begin with some of the techniques used by Toutanova et al. (2002). The obvious improvement to make is to change the overall model from a simple PCFG to a more theoretically sound log-linear model—although it would not be straightforward to transfer my contextual augmentation techniques to a log-linear framework. Toutanova et al.'s results show that adding ancestor annotation[1] to a PCFG results in a performance increase roughly equal to that produces by switching to a log-linear model. And although Toutanova et al. (2002) alludes to a fairly complex method for selecting the degree of ancestor annotation, it seems from the follow-up publication Toutanova et al. (2003) that equal performance is achieved by fixed annotation with the parent and grandparent node. Fixed ancestor annotation, then, would seem to be a very simple way of strongly improving the grammar's performance.

[1] Ancestor annotation consists of conditioning a rule's probability on one or more of its ancestors in the derivation tree.

8.2.3 Clarification subdialogues

Single-rephrase and binary choice clarification perform adequately, but there is scope for the implementation of more sophisticated and natural techniques. Referent-based clarification has great promise, especially since so much of my work on rephrasing-based clarification can be carried over to it. As briefly mentioned in Section 5.3.13, binary-rephrase clarification also has scope for extension by considering ternary and higher-degree partitions of the interpretation set.

8.2.4 Errors

Apart from the actual implementation of the techniques I propose, the main consideration for error handling is integrating it with Te Kaitito's recently added module for acquiring unknown words from the user, described by van Schagen (2004). Clearly decisions will have to be made regarding when to consider an unknown word as a misspelling of a known one, and when to initiate a word-acquisition subdialogue.

8.2.5 Data gathering

These are exciting times for Te Kaitito. In addition to the work in this thesis, other new modules are being developed (see e.g. Slabbers, in preparation; van Schagen, 2004), and user trials are scheduled to begin soon. From the point of view of system development, this brings two opportunities: firstly, that of evaluating the performance of the system in the environment for which it was originally designed; and secondly, that of making use of Te Kaitito's new interlocutors as a source of data.

It will be easy to add facilities to Te Kaitito to produce records of its dialogues at every level from text string to dialogue act. Essentially we can enlist our users as annotators in the semi-automated building of a treebank. Instead of explicitly indicated preferred parses, they use the clarification mechanism—although in most cases the disambiguation pipeline should be able to do the job automatically.

There are vast possibilities for the purposes to which this data might be put. In relation to the work of this thesis, the most obvious use is the construction of a syntactic-level treebank to set parameters for a stochastic Māori-English Grammar, with all the attendant benefits for various parts of the dialogue system. But we can also record data at the semantic and dialogue act levels. There has been work on tuning pronoun resolution algorithms using corpora marked up for anaphoric dependencies using the Penn Discourse Treebank (Miltsakaki, Prasad, Joshi, and Webber, 2004); similar

techniques could be applied to Te Kaitito's semantic-level data. The dialogue-act information could be used to set parameters for a discourse parser similar to the one proposed by Baldridge and Lascarides (2005). Data can be collated by learner level to produce concrete representations of the linguistic knowledge expected at each stage of a language course.

There are currently treebanks marked up by hand at a sentence level with human-generated parses—for example, the Penn treebank. There are also sentence-level treebanks created by semi-automatic processes where the annotator merely has to select a machine-generated parse, such as the Redwoods treebank used in this project. And there are hand-generated treebanks marked up at the discourse level, such as the annotations on the SWITCHBOARD corpus described by Stolcke et al. (2000). Te Kaitito's users can help us to create a treebank with this kind of discourse-level mark-up, but without the time and effort required to explicitly mark up every utterance by hand. Thus, for no extra effort on the part of our users, we will gain a valuable resource with applications far beyond sentence disambiguation.

References

Steven Abney. Statistical methods and linguistics. In Judith Klavans and Philip Resnik, editors, *Balancing Act: Combining Symbolic and Statistical Approaches to Language*. The MIT Press, 1996.

Steven P. Abney. Stochastic attribute-value grammars. *Computational Linguistics*, 23: 597–618, 1997.

Jim L. Alty and Mike J. Coombs. *Expert Systems—Concepts and Examples*. NCC Publications, The National Computing Centre Ltd., Manchester, England, 1984.

Kevin Atkinson. aspell. Web page, 2005. URL `http://directory.fsf.org/aspell.html`. Viewed on 26 January 2005.

J. Baldridge and A. Lascarides. Annotating discourse structure for robust semantic interpretation. In *Proceedings of the Sixth International Workshop on Computational Semantics*, Tilberg, The Netherlands, 2005.

Ian Bayard, Alistair Knott, and Samson de Jager. A unification-based grammar for a fragment of Māori and English. In *Proceedings of the 2nd Australasian Natural Language Processing Workshop (ANLP 2002)*, 2002.

Thomas Gordon Bever. *The Cognitive Basis for Linguistic Structures*, pages 279–352. Wiley, New York, 1970.

Taylor L. Booth. Probabilistic representation of formal languages. In *Tenth Annual IEEE Symposium on Switching and Automata Theory*, pages 74–81, 1969.

Chris Brew. Stochastic HPSG. In *Proceedings of the seventh conference of the European chapter of the Association for Computational Linguistics*, pages 83–89. Morgan Kaufmann Publishers Inc., 1995.

Eugene Charniak. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the Fourteenth National Conference on Artifical Intelligence (AAAI '97)*, pages 598–603, 1997.

Stanley F. Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th annual meeting of the Association for Computational Linguistics*, pages 310–318. Association for Computational Linguistics, 1996.

Marcus Tullius Cicero. *The Orations of Marcus Tullius Cicero*, volume 4. George Bell & Sons, 1903. Translated by C. D. Yonge.

Michael Collins. Three generative, lexicalized models for statistical parsing. In Philip R. Cohen and Wolfgang Wahlster, editors, *Proceedings of the Thirty-Fifth Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, pages 16–23, Somerset, New Jersey, 1997. Association for Computational Linguistics.

Ann Copestake, Dan Flickinger, and Ivan A. Sag. Minimal recursion semantics: An introduction (draft), 1999. URL `http://www.cl.cam.ac.uk/\%7Eaac10/papers/newmrs.ps`.

Anne Copestake and Dan Flickinger. An open-source grammar development environment and broad-coverage English grammar using HPSG. In *Proceedings of LREC 2000*, Athens, Greece, 2000.

Samson de Jager, Alistair Knott, and Ian Bayard. A DRT-based framework for presuppositions in dialogue management. In *Proceedings of the 6th Workshop on the Semantics and Pragmatics of Dialogue (EDILOG 2002)*, September 2002.

Stephen Della Pietra, Vincent Della Pietra, and John Lafferty. Inducing features of random fields. Technical Report CMU-CS-95-144, CMU, 1995.

Lorene Earnshaw, Stewart Fleming, Victoria Weatherall, and Alistair Knott. Analysis of errors in the writing of first year students of Maori. *Journal of Maori and Pacific Development*, 5(2):31–47, September 2004.

Dan Flickinger. On building a more efficient grammar by exploiting types. *Natural Language Engineering*, 6(1):15–28, 2000. Special Issue on Efficient Processing with HPSG.

Henry Fuchs, Zvi M. Kedem, and Bruce Naylor. Predetermining visibility priority in 3-D scenes (preliminary report). In *SIGGRAPH '79: Proceedings of the 6th annual conference on Computer graphics and interactive techniques*, pages 175–181. ACM Press, 1979.

J. Ginzburg and I. Sag, editors. *Interrogative investigations*. CSLI Publications, Stanford, 2000.

Herbert Paul Grice. Logic and conversation. In Peter Cole and Jerry L. Morgan, editors, *Syntax and Semantics*, volume 3: Speech Acts, pages 41–58. Academic Press, New York, 1975.

Barbara J. Grosz and Candace L. Sidner. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(3):175–203, July-September 1986.

J Hobbs, M Stickel, D Appelt, and P Martin. Interpretation as abduction. *Artificial Intelligence*, 63, 1993.

Homer. *The Odysseys of Homer*. J. R. Smith, London, 1857. Translated by George Chapman.

Peter Jackson. *Introduction to Expert Systems*. International Computer Science Series. Addison-Wesley, Wokingham, England, second edition, 1990.

Jerome K. Jerome. *Three Men in a Boat*. J. W. Arrowsmith, 1889.

Mark Johnson, Stuart Geman, Stephen Canon, Zhiyi Chi, and Stefan Riezler. Estimators for stochastic "unification-based" grammars. In *Proceedings of the 37th Meeting of the Association for Computational Linguistics*, pages 535–541, 1999.

Daniel Jurafsky and James H. Martin. *Speech and Language Processing*. Prentice Hall Series in Artificial Intelligence. Prentice Hall, 2000.

Hans Kamp, J. van Genabith, and Uwe Reyle. Discourse representation theory. In *Handbook of Philosophical Logic*. Springer-Verlag, in preparation.

Frank Keller and Mirella Lapata. Using the web to obtain frequencies for unseen bigrams. *Computational Linguistics*, 29(3):459–484, 2003.

S. King, A. Knott, and B. McCane. Language-driven nonverbal communication in a bilingual conversational agent. In *Proceedings of the 16th International Conference on Computer Animation and Social Agents (CASA)*, May 2003.

Alistair Knott and Peter Vlugter. Syntactic disambiguation using presupposition resolution. In *Proceedings of the 2003 Australasian Language Technology Workshop (ALTW2003)*, pages 98–104, 2003.

Alistair Knott, Ian Bayard, Samson de Jager, and N. Wright. An architecture for bilingual and bidirectional NLP. In *Proceedings of the 2nd Australasian Natural Language Processing Workshop*, December 2002.

Alistair Knott, Ian Bayard, and Peter Vlugter. Multi-agent human-machine dialogue: Issues in dialogue management and referring expression semantics. In C Zhang, H Guesgen, and W Yeap, editors, *PRICAI 2004: Trends in Artificial Intelligence. Proceedings of the 8th Pacific Rim Conference on Artificial Intelligence*, Lecture Notes in AI, pages 872–881. Springer Verlag, 2004.

Geoffrey Kuenning. ispell. Web page, 2001. URL `http://directory.fsf.org/ispell.html`. Viewed on 26 January 2005.

Karen Kukich. Techniques for automatically correcting words in text. *ACM Computing Surveys*, 24(4):377–439, December 1992.

Milan Kundera. *The Book of Laughter and Forgetting*, page 237. Penguin, 1983.

Corrin Lakeland. *Lexical Approaches to Backoff in Statistical Parsing*. PhD thesis, University of Otago, Dunedin, New Zealand, in preparation.

Alex Lascarides and Nicholas Asher. Discourse relations and defeasible knowledge. In *Proceedings of the 29th Conference of the Association for Computational Linguistics*, pages 55–63, Berkeley, CA, 1991.

D. Lewis. Scorekeeping in a language game. *Journal of Philosophical Logic*, 6:339–359, 1979.

Diane J. Litman and James F. Allen. A plan recognition model for clarification subdialogues. In *Proceedings of the 22nd conference of the Association for Computational Linguistics*, pages 302–311. Association for Computational Linguistics, 1984.

Elias Lönnrot, editor. *Kalevala: the land of the heroes*. Number 259 in Everyman's Library. J. M. Dent & Sons Ltd., London, 1907. Translated by W. F. Kirby.

Pontus Lurcock, Peter Vlugter, and Alistair Knott. A framework for utterance disambiguation in dialogue. In Ash Asudeh, Cécile Paris, and Stephen Wan, editors, *Proceedings of the Australasian Language Technology Workshop*, pages 101–108, Sydney, Australia, 2004. Macquarie University, The Australian Speech Science and Technology Association.

Robert A. MacLachlan. CMU Common Lisp user's manual. Technical report, Carnegie Mellon University, 1992.

Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts, 1999.

Wolfgang Menzel and Ingo Schröder. Error diagnosis for language learning systems. *ReCALL*, 1999. URL `citeseer.ist.psu.edu/menzel99error.html`.

A. M. Miller, H. E. Pople, and J. D. Myers. INTERNIST-I, an experimental computer-based diagnostic consultant for general internal medicine. *New England Journal of Medicine*, 307:468–476, 1982.

Eleni Miltsakaki, Rashmi Prasad, Aravind Joshi, and Bonnie Webber. The Penn discourse treebank. In *Proceedings of the Language Resources and Evaluation Conference*, Lisbon, Portugal, 2004.

Saif Mohammad and Ted Pedersen. Complementarity of lexical and simple syntactic features: The SyntaLex approach to Senseval-3. In *Proceedings of the Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text (Senseval-3)*, Barcelona, Spain, 2004.

Stephan Oepen. [incr tsdb()] competence and performance laboratory: User and reference manual, 1999. URL `http://citeseer.ist.psu.edu/457852.html`.

Stephan Oepen, Dan Flickinger, Kristina Toutanova, and Christoper D. Manning. LinGO Redwoods: A rich and dynamic treebank for HPSG. In *Proceedings of the First Workshop on Treebanks and Linguistic Theories (TLT2002)*, 2002.

Carl Pollard and Ivan A. Sag. *Head-Driven Phrase Structure Grammar*. University of Chicago Press, 1994.

Jan Potocki. *The Manuscript Found in Saragossa*. Penguin, London, 1995. Translated by Ian Maclean.

Ronald Rosenfeld. A maximum entropy approach to adaptive statistical language modeling. *Computer, Speech and Language*, 10:187–228, 1996.

H. Sacks, E. Schegloff, and G. Jefferson. A simplest systematics for the organization of turn-taking for conversation. *Language*, 50(4):696–735, 1974.

Nanda Slabbers. A system for generating teaching initiatives in a computer-aided language learning dialogue. Technical report, University of Otago, Dunedin, New Zealand, in preparation.

Norman K. Sondheimer and Ralph M. Weischedel. A rule-based approach to ill-formed input. In *Proceedings of the 8th Conference on Computational Linguistics*, pages 46–53, Morristown, NJ, USA, 1980. Association for Computational Linguistics.

Guy L. Steele, Jr. *Common Lisp: The Language*. Digital Press, second edition, 1990.

Laurence Sterne. *The Life and Opinions of Tristram Shandy*. Number 617 in Everyman's Library. J. M. Dent & Sons Ltd., London, 1912.

Andreas Stolcke, Klaus Ries, Noah Coccaro, Elizabeth Shriberg, Rebecca Bates, Daniel Jurafsky, Paul Taylor, Rachel Martin, Marie Meteer, and Carol van Ess-Dykema. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational Linguistics*, 26(3):339–371, 2000.

D. A. Swinney. Lexical activation during sentence comprehension: (re)consideration of context effects. *Journal of Memory and Language*, 1979.

James Thurber. *The Thurber Carnival*. Hamish Hamilton, London, 1945.

Kristina Toutanova, Christopher D. Manning, Stuart M. Shieber, Dan Flickinger, and Stephan Oepen. Parse disambiguation for a rich HPSG grammar. In *First Workshop on Treebanks and Linguistic Theories (TLT2002)*, pages 253–263, 2002.

Kristina Toutanova, Christopher D. Manning, Stephan Oepen, and Dan Flickinger. Parse selection on the Redwoods corpus: 3rd growth results. Technical Report 64, Stanford University, California, 2003.

David Traum. Computational models of grounding in collaborative systems. In *Working Papers of the AAAI Fall Symposium on Psychological Models of Communication in Collaborative Systems*, 1999.

Maarten van Schagen. Tauira: A bilingual dialogue-based lexical acquisition system. Technical Report OUCS-2004-06, Department of Computer Science, University of Otago, Dunedin, New Zealand, 2004.

R. A. Wagner and M. J. Fischer. The string-to-string correction problem. *Journal of the Association for Computing Machinery*, 21:168–173, 1974.

Wolfgang Wahlster, editor. *Verbmobil: Foundations of Speech-to-Speech Translation*. Artifical Intelligence. Springer, Berlin, July 2000.

Larry Wall, Tom Christiansen, and Jon Orwant. *Programming Perl*. O'Reilly & Associates, Inc., third edition, 2000.

Yuanyong Wang and Achim Hoffman. A new measure for extracting semantically related words. In Ash Asudeh, Cécile Paris, and Stephen Wan, editors, *Proceedings of the Australasian Language Technology Workshop*, Sydney, Australia, 2004. Macquarie University, The Australian Speech Science and Technology Association.

Ralph M. Weischedel, Wilfried M. Voge, and Mark James. An artificial intelligence approach to language instruction. *Artificial Intelligence*, 10(3):225–240, 1978.

G. Zipf. *Human Behaviour and the Principle of Least Effort*. Addison-Wesley, 1949.