Department of Computer Science,
University of Otago

Technical Report OUCS-2004-06

Tauira: A bilingual dialogue-based lexical acquisition system

Author:
Maarten van Schagen
University of Twente

Status: Internship Project Report

Department of Computer Science,
University of Otago, PO Box 56, Dunedin, Otago, New Zealand

http://www.cs.otago.ac.nz/trseries/

Tauira:
A bilingual dialogue-based lexical acquisition system

Maarten H van Schagen

Intern
Artificial Intelligence Group
Department of Computer Science
University of Otago
New Zealand
maarten@cs.otago.ac.nz

Supervisor: Alistair Knott
alik@cs.otago.ac.nz

Student
Parlevink Group
Faculty of Electrical Engineering,
Mathematics and Computer Science
University of Twente
the Netherlands
schagen@cs.utwente.nl

Supervisor: Anton Nijholt
anijholt@cs.utwente.nl

Abstract

A lexical authoring tool (Tauira) placed within a bilingual bidirectional dialogue context is described. The tool's dialogue is initiated when one or more unknown words are uttered by the user in the surrounding dialogue system. Based on the context of the sentence the word was uttered in a set of hypotheses are created for possible word types and stems of the unknown word. These hypotheses are reduced using a set of multiple choice questions until only one remains: the new word entry. Hypothesis reduction is done by asking closed questions about the syntactic validity of the unknown word in example sentences. These example sentences are generated based on sentences from the test suite which accompanies the system's grammar. Because the tool operates in a bilingual (English-Māori) context, both the unknown word and its translation are added. The translation for an unknown word is deduced from a translation of the original sentence containing the source word. Not only does this tool provide an interesting addition to the field of lexical authoring but in addition also can side effects of the project can be used to formally evaluate the coverage of a test suite on a grammar concerning word types.

Contents

Chapter 1

Introduction

This document provides a summary of the work I did during my internship at the Department of Computer Science at the University of Otago. I was working in the Computational Linguistics group and contributing to the **Te Kaitito** system - a natural language system able to engage in dialogues with the user in English and Māori. My project was to build a tool which allows the user to add new words to Te Kaitito's lexicon, prefrably without needing any detailed knowledge of linguistics. The tool is titled **Tauira** which is Māori for both 'the student' as 'the example'. This is because Tauira has the ability to learn new words by using examples.

1.1 Overview

Chapter One will introduce the reader into the context of Tauira by elaborating on the Te Kaitito project in general and its architecture in detail and specify the requirements for Tauira. Chapter Two explores related works in a literature review. Chapter Three presents and motivates the design of Tauira which and a discussion of its implementation is given in Chapter Four. In Chapter Five the results of the project are illustrated through example dialogues as well as benchmarks. Chapter Six summerizes the project and explores future work to be done on Tauira. There are two appendices: Appendix A describes the usage of Tauira for a grammar writer and the Appendix B is a template for a user's guide.

1.2 Te Kaitito project

The artificial intellegence group is one of the four main research groups at the Department of Computer Science at the University of Otago, Dunedin New Zealand.

One of the main projects of the Computational Linguistics subdivision is the development of an architecture for Māori and English language processing and generation. This project is known as Te Kaitito which Māori for the 'composer' or 'the improviser'.

Te reo Māori (lit: the Māori language) is the language spoken by the Polynesian inhabitants (the Māori) who immigrated to New Zealand seven hundred years before Abel Tasman's discovery of New Zealand in 1642. When the English started colonizing New Zealand at the beginning of the 19th century this initiated a lot of conflict between the Māori and Pakeha (the Māori name for the European settlers) until signing of the Treaty of Waitangi in 1840. The treaty not only gave Māori possession of their land but established equal rights for Pakeha and Māori subjects effectively making New Zealand (or Aotearoa in Māori) a biculteral and bilingual culture.

By the 1970's however, te reo Māori was on the brink of extinction as a result of socetal practices such as school education which emphasised the importance of English. In the same perioad a revival in Māori culture was called for by many young Māori. This spurred the government to give the Māori language a greater prominence in schools and the media (Knott *et al.*, 2002; Harding *et al.* 2002).

Villa (2002) suggsets that computer technology has the possibility of filling an important niche in minority language maintenance and teaching. So to assist its survival and the assimilation of te reo Māori by Māori and non-Māori New Zealanders this bilingual architecture is developed. Applications of this architecture will be an on-line teaching program, an on-line translator and a dialog system.

Besides being of use for Māori language maintenance and teaching the project also provides a platform for computer science students interested in computational linguistics to advance their skills in this field. So in addition to the two staff members working on the project, one as a writer for the bilingual grammar and one for the development of the dialog systems and web interfaces, multiple computer science students have done larger or smaller projects using this architecture.

The Human-Media Interaction or Parlevink group is part of the department of Electrical Engineering, Mathematics and Computer Science, (EEMCS) at the University of Twente, Enschede the Netherlands. Speech and Language technology is one of the main topics covered in the group's educational programme. A student from the parlevink group would be able to put his or her theoretical knowledge of computational linguistics to practical use on a project such as Te Kaitito, whilst the project as a whole would benefit from any enhancements made.

1.3 Te Kaitito architecture

Te Kaitito is a bidirectional system: the same declarative modules (the **grammar**, **lexicon** and **morphological rules**) are used for the analysis as generation of sentences. Te Kaitito is also bilingual which means for every declarative module there is an English and a Māori instance. The entire architecture is displayed in Figure 1.1.

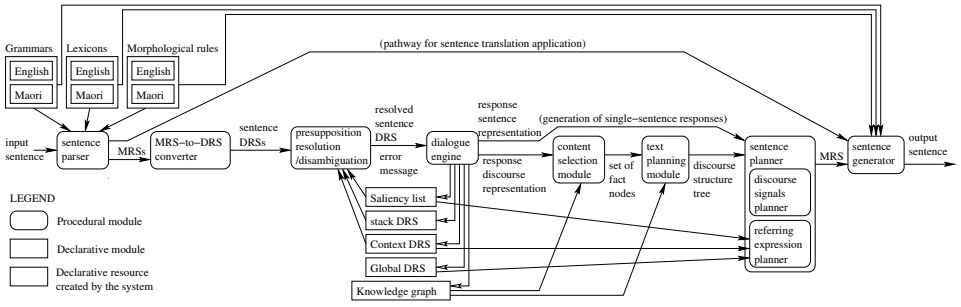

Figure 1.1: Architecture of the Te Kaitito system

The declarative modules are used in the first and final step of generating a responce based on an input sentence. The first step consists of parsing the sentence using the parser from the LKB grammar development system (Copestake, 2002). One of the results of this parsing process is a semantic representation of the sentence in a language called **Minimal Recursion Semantics (MRS)** (Copestake, 2001). MRS is also used to represent the output sentences semantics from which the sentence generator will generate an output.

When the architecture serves as a translator the MRS resulting from the parse is simply passed onto the sentence generator which will then generate the semantics of the original sentence in both Māori and English. e.g. in the following dialog:

```
> The dog barks
ka auau te kurī
the dog barks
```

The architecture can however be configurd to do more complex tasks as explained earlier. For the dialog system the MRS is converted into a **Discourse Representation (DRS)** (Kamp and Reyle, 1993). This DRS is presupposed for the current context. The result presupposistion will create a new response for the user. A detailed discussion of this part of the architecture can be found in Knott *et al.*

(2003). An example interaction with the dialogue system is given below. Note that each sentence generated or parsed in this example could also be done so in Māori.

```
> A blue dog barked
Ok
> A red dog walked
Ok
> Which dog walked?
It was the red dog
> The dog ate the weasel
which dog ate the weasel ?
> The red dog
Ok
```

1.4 Linguistic Knowledge Building system

The **Linguistic Knowledge Building (LKB) system** was created for general language processing and generation tasks using a typed feature structure grammars such as Head-driven Phrase Structure Grammars (HPSG). The LKB system can be seen as a development environment for a very high-level specialized programming language. Typed feature structure grammars are a language based on one data structure (**the typed feature structure**) and one operation (**unification**) (Copestake, 2002).

A feature structure is basically a set of features possilby given a name. Everything (grammar rules, start symbols, lexical entries) to define the grammar is defined in a feature structure. An example of the definition of a lexical entry for a grammar is:

```
man := noun &
[ ORTH.LIST.FIRST "man",
  HEAD.AGR.GENDER masc
  SEM.HOOK.KEYPRED < [ PRED man_rel ] >
].
```

Man is the identifier of this feature structure and the **word type** *noun* is a feature structure whose features are added to this feature structure. Other features define the orthography, gender and semantics of this lexical entry. The largest grammar that so far has been developed on the LKB system is the **English Resource Grammar (ERG)** by the LinGo group (Copestake and Flickinger, 2001). For the larger grammars large **test suites** have been developed. A test suite is a set

of systematically constructed individual sentences designed to test the covarege of the grammar. Characteristics of test suites are (Oepen *et al.*, 1997):

- The sentence of test data are controlled, unlike those of e.g. corpora.
- Every specific gramatical phenomenon is covered systematically.
- Non-redudant representation (every phenomenon appears only once).
- Ill-formed sentences are included (and annotated as such).
- The annotation is coherent.

Large test suites have been created as part of the **Test Suites for Natural Language Processing (TSNLP)** project (Oepen *et al.*, 1997). These test suite are stored in a database called **itsdb** (Oepen, 2001). The LKB is case insensitive and ignores symbols such as *'-'*, *'"'*, *'?'* and *'.'*.

1.5 Tauira project goals

During any form of human-computer interaction through a dialogue system unknown words are bound to show up. In order to correctly handle unknown words, an extension of the dialogue system is required for Te Kaitito. This system also serves a secondary goal: expanding the lexicon of Te Kaitito, which is currently quite small. The extension (Tauira) should react to words that cannot be found in the lexicon during interaction with te Kaitito.

There are many ways in which this extension could be made. The most simple method would be to open up a text editing tool, to allow the user to add a new entry to Te Kaitito's lexicon file. But this requires to know details about how the system is built up. The original idea of the current project was to build a tool that allows the user to author new words *in the context of a human-computer dialogue*. So, In stead of just giving a "word-not-found" error a subdialogue is instantiated. This dialogue should consist of various questions to which the answers of the user can be used so all the required aspects of the word can be filled in. Aspects that need to be filled a word are:

- Its word type.
- Its Māori or English translation.
- Its translation's word type.

General Requirements of the system include:

- The user should not be required to have any knowledge of linguistics.
- The extension should be adaptable to the ever changing Māori and English grammar of te Kaitito.
- These adaptations can be done by the grammar writer.
- The system should be integratable with the current dialogue system.

The system has the following presuppositions:

- The underlying grammar of the current dialogue system is sound and complete in respect to its lexicon.
- For each possible grammatical word type there exists at least one lexical entry
- Users of the system are familiar with both English and Māori.

Chapter 2

Related works

In this chapter relevant literature for the project is discussed. There are two basic paradigms for word authoring which need to be adressed. Section 2.1 will discuss the 'learning by instruction' paradigm and Section 2.2 will discuss syntactic extraction from sentence context. Finally in Section 2.3 I will provide some background in lexical semantics in sentence translation which is neccessary for authoring new words in a bilingual context.

2.1 Learning by instruction

The suggestion that lexical items can be authored in a human-computer dialogue system is not new. The idea is first proposed by Knight (1996) as the **Learning by Instruction** paradigm. Knight proposes two dialogue based methods whereby a naive user could add new words to a system's lexicon:

The first approach is to ask a set of multiple choiche questions in which the new word is placed in different grammatical contrasts and the user is asked about their syntactic corectness or where the user is asked various conceptual questions about the nature of the new word, as is illustrated in the following dialogue:

```
H: "John is hungry."
C: I don't know the meaning of "hungry".
   Is "very hungry" a reasonable phrase?
H: Yes
C: Is "hungry" a visually detectable property ?
H: No.
C: Does "hungry" refer to a capability (or lack of one) ?
H: No
C: Does "hungry" refer to a desire?
```

```
H: Yes.
C: Is it a desire to (a) do something, (b) own something,
   (c) be something, (d) other?
H: (a).
C: A desire to do what ?
H: To eat.
```

The second approach is to let the user specify a phrase (the **paraphrase**) semanticly equivalent to the original phrase but using concepts the system is already familiar with as in the following example:

```
H: "John is hungry."
C: I don't know the meaning of "hungry"
H: I mean: "John wants to eat".
C: I understand.
```

Knight observes that although the multiple choice method might work well for syntax, for semantics there are many features and cases and the lingistical diagnosic is rather vague it would be impractical.

A major problem in the multiple choice method however is actually coming up with these sentences automatically, since what Knight calls limited features and cases make up to five hundred word types for a large-coverage grammar such as the ERG. This is a problem my system will address.

The paraphrasing method is pursued further by Knight. Semantics of words are represented in graphs, semantics of sentences can be created using graph unification. With this representation the semantics of an unknown word can be extracted using the semantics of a paraphrase. This is compared to solving an equation such as $hungry * is + John = John + want/eat$, so $hungry = want * is/eat$. Finding the solution of semantic graphs is of course somewhat more complicated, but is proven to be possible using Knight's algorithm.

2.2 Syntactic processing of unknown words

2.2.1 Extracting information from sentence context

Erbach (1990) and Barg and Walther (1998) devised a method of processing unknown words in unification grammars. Besides processing sentences with unknown words an entry for the unknown word is created based on characteristics of the sentence context. A human reader ignorant of the word *'child'* can deduce from the sentence *'the child chased his mother'* that *'child'* is a noun, that *'child'* is singular (ortherwise it would be *'the child chase their mother'*) and *this particular instance* of *'child'* is male.

Erbach (1990) presents an algorithm for processing of unknown words. For an unknown word a new entry will be created of a type wich is considered open class (these types will be refered to as **open word types**). The algorithm consists of three steps:

1. Parse the sentence. Use the set of all open word types as the feature structure for an unknown word.

2. Enrich the unknown word entry with information from the sentence context.

3. Refine the information from the previous step to create an appropriate entry.

Before Step 1 the unknown word is represented as a disjunction of all the open word types. During the parsing parts of the disjunction are removed because they are unable to unify with the rest of the sentence context. In Step 2 whatever remains of the unknown word entry is enriched with information from the sentence context. In Step 3 overspecified information is eliminated.

In the previous example *'child'* initially is the disjunct of nouns, intransitive verbs, transitive verbs and adjectives (given that these are the only open word types). After Step 1 only the feature structure for the noun remains. After Step 2 *'child'* is enriched with the information that its gender must be *male* and its number is *singular*. In step 3 the information on gender is removed and a new lexical entry is added.

For step 3 filters need to be defined for each open word type. The feature structure of step 2 will be filtered and unified with the feature structure of a particular word type. A filter in the example for nouns that only maintains the number and removes the gender feature might look something like:

```
<lexentry val arg num> = <info val arg num>
```

Barg and Walther (1993) extend this algorithm so it can also handle:

- Selectional restrictions
- Semantic types
- Argument structure

Also in Barg and Walther's vision of unknownness words are not necessarily known or unknown but are revised constantly. Instead of using filters for specific words, the grammar writer must now fully declare which individual pieces of information are open to revision. This is done by identifying generalizable or specializable clauses like for gender in noun:

specializable(1) :=

$$\left[\text{synsem}|\text{loc}\left[\begin{array}{l}\text{cat}|\text{head}\ noun\\ \text{cont}|\text{ind}|\text{num}\ \boxed{1}\end{array}\right]\right]$$

Specializable information is information which like the plurality of a noun cannot be overspecified in the context of a sentence. Generalizable information is information like the gender of a noun which can be overspecified within the context of a sentence.

Fouvry (2003) adapted the technique described by Erbach and Barg and Walther in an algorithm for the LKB system. In his paper he observes that:

- Longer sentences with unknown words can become unparsable .
- Sentences with multiple unknown words are unparsable.
- There are grammars in which lexical entries are defined without features (like the ERG). In these grammars generating an unknown word entry is a matter of selecting the right word type.

2.2.2 Extracting information from statistics

Barg and Walther (1998) constantly revise their lexical entries based on new information. When e.g. a system is given the following sentences from different users:

```
The mother chases the cat
The mother held her child
The mother held her gun
The officer chases the cat
The officer held her child
The officer held his gun
```

The system can conclude that *'mother'* is feminine and *'officer'* is both feminine and masculine. The drawback of this method is that for it to create a robust lexicon a large number of input sentences need to be given.

Fouvry (2003) creates a solution using the final observation made in the previous section: the fact that there are grammars where lexical entries are defined without features. In his setup selecting of an unknown word becomes word-type selection opposed to feature structure creation. He further observes that the ERG approach to features creates the disadvantage that the number of types is fairly high (in his case 463). To help the selection process of the appropriate word-types

Fouvry (2003) integrates knowledge from a statistical Part of Speech (PoS) tagger to calculate the probability of each type in this sentence. This method drastically decreases the number of types.

2.3 Translation

A final piece of literature to introduce relates to formalisms for representing word semantics in two different languages. Copestake and Sanfilippo (1993) describe bilingual cross-links called **tlinks**, to be used to define relations between monolingual lexical items. Through these tlinks different forms of translation equivalence can be represented. The tlinks are defined bidirectionally. *Simple*-tlinks are applicable when two lexical entries denote single place predicates and are straightforwardly equivalent, without any transformation being necessary. E.g. *'chocolade'* is the Dutch translation for the English *'chocolate'*.

A more complex tlink could be defined where it states that for the word *'human'* the *sex* feature for source feature structure and target feature structure is equal. Also restictions on the use where a word exists only in plural in one language (e.g. the English *'furniture'*) while it does exits in both singular and plural in the other language (e.g. the Spanish *'mueble'* and *'muebles'*).

Translation within the current Te Kaitito architecture is realised by predicate equivalence. E.g. *'dog'* and *'kurī'* both have the same predicate (*dog_rel*). A bit more complex translations are allowed for entries such as *'kitten'* which has the semantics *cat_rel* and *young_rel* so this could be translated to *'punua'* with *young_rel* and *'poti'* with *cat_rel*. All these translations could be seen as simple-tlinks.

Chapter 3

Design

This chapter describes and motivaties the design for Tauira. First a top-level overview is given of the design (Section 3.1) then this design is motivated (Section 3.2).

3.1 Top-level

3.1.1 Introduction

Tauira operates in the context of a human-computer dialogue system. The system will initiate once the user enters a sentence containing an unknown word. From the sentence context a set of so called hypotheses for the unknown word is derived. A **hypothesis** is a pair of a word type (e.g. *n_intr_le*) and morphological **stem** (e.g. *'walk'*). The set *hypotheses* are all the type-stem pairs which could be valid for the unknown word in the sentence. This method is an improvement to that of Erbach (1990), Barg and Walther (1998) and Fouvry (2003) in the respect that:

- No filters are required to specifiy lexical entries.
- Multiple unknown words making up one lexical entry can be resolved.
- Morphologically complex unknown words can be resolved.

The system will then try to enter a dialogue similar to Knight's (1996) multiple choice dialogues (Section 2.1). The goal of this dialogue is to iteratively reduce the set of hypotheses until only one type-stem pair remains. A key innovations is that the system generates the multiple choice alternatives automatically. It takes sentences from the grammar's test suite and tries to apply a hypothesis to this sentence in order to create a phrase for the user to accept or decline.

Finally the system will require the user to provide a translation of the original sentence the unknown word was uttered in. This is similar to the paraphrasing method Knight (1996) proposes. The semantics for the translation will then be devised on a simple-tlink level as described by Copestake and Sanfilippo (1993).

In summary the Tauira system combines and extends the dialogue strategies proposed by Kight (1996) with the syntactic extraction by Erbach (1990), Barg and Walther (1998) and Fouvry (2003).

3.1.2 Modifications to Te Kaitito

Figure 3.1 shows the modified architecture of the Te Kaititito system for Tauria. The system is extended with two extra procedural modules: Tauira and the preprocesser and two extra declarative resource: the user lexicon and the preprocessed test suite. The **user lexicon** consists of the newly created (i.e. previously unknown) words for the current user. Also an additional declarative resource is used: test suites.

If no unknown words are found in a parse the MRS is fed through the rest of the system as usual. If an unknown word is found in the sentence, the sentence is passed to Tauira. Tauira will create an appropriate response for the user (questions concerning the unknown word) and will intercept input sentences (which are the answers to the previous questions) from the user. Once all unknown words are resolved and added to the user lexicon, the original sentence is passed to the sentence parser and the normal dialogue continues.

Tauira makes use of the preprocessed test suite to generate its output sentences. The preprocessed test suite is filled with test items by the **preprocessor**. The preprocessor does this on the basis of test suites.

3.1.3 Preprocessing: trawling a test suite to look for example sentences

In order to decrease the number of hypotheses the user will be queried using example sentences. Ideally, for each open word type *t* in the grammar, we would like to find a sentence in which a word of type *t* appears, and no other open word type is possible in the same sentence context. If we can find such a sentence we can query the user if the unknown word *w* would be grammatically appropriate in this type. If the user accepts this sentence as a correct sentence the word *w* is of type *t*; if the user declines this sentence every hypothesis containing *t* will be removed from *hypotheses*. Potentially very useful sets of sentences are test suites, the reasons why are explained in Section 3.2.3. For efficiency purpose sentences are preprocessed so they can be used directly by the system; this process is described in Section 4.2.

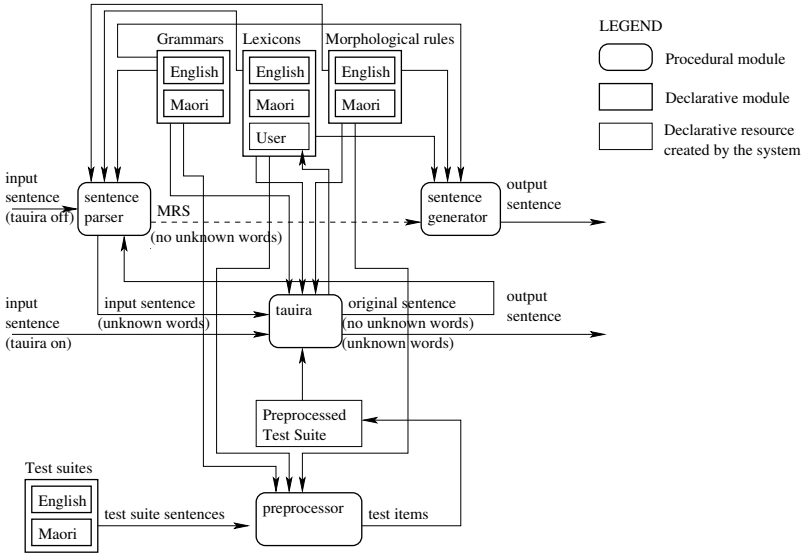

Figure 3.1: Tauira within the architecture of the Te Kaitito system

Before this can be done it has to be decided which word types are considered open (class), this is discussed in Section 4.1.

3.1.4 Algorithm overview

The algorithm of Tauira performs the following basic steps to add a new word. These are motivated in Section 3.2 and elaborated in Chapter 4.

1. Locate unknown words in the sentence.
2. Create hypotheses for the unknown word based on the context of the sentence.
3. Decrease hypotheses until only one remains.
4. Ask for a translation and create semantic relations.
5. Repeat steps 1 to 3 for the translation if the translation is also a new word.
6. Add the new items to the lexicon.
7. Process the original sentence.

3.2 Motivation

3.2.1 Locating unknown words

The first step of the algorithm is to identify which are the unknown lexical entries in the sentence and to make sure one at a time is processed. If there is only one unknown word in the sentence, this step is trivial. If there are multiple unknown words there are two possibilities:

1. Multiple unknown words make up one lexical entry.

2. Multiple unknown words make up separate lexical entries.

Possibility 1 will occur a lot since proper names such as (*'Maarten van Schagen'*) and Māori nouns with the word part *'te'* such as *'te taha ki'* will always be a large class of unknown lexical entries. In case there are multiple *consecutive* unknown words the user is consulted whether these make up one lexical entry.

Fouvry (2003) observes that in his current setup for feature extraction multiple unknown words in a sentence can almost never be treated due to the compounded ambiguity. So sentences of possibility 2 cannot be treated. Tauira can however take advantage of the fact that it can consult the user directly. The system will therefore ask the user to devise a sentence using just one of the unknown lexical entries. For each lexical entry the steps of the algortithm can be traversed iteratively given this new sentence.

3.2.2 Creating hypotheses

The next step is to create hypotheses for the unknown lexical entry based on its sentence context. As described earlier in Section 2.2 Erbach (1990) and later Barg and Walther (1998) have devised a technique to extract a feature structure for an unknown word based on the context of the sentence the word is contained. Choosing to adapt this algorithm is a very logical choice: Erbach (1990) came up with a method for processing sentences which contain unknown words, Barg and Walther (1998) refined this method and related it to a Head-driven Phrase Structure Grammar (HSPG) and Fouvry (2003) created an implementation of their method using the LKB system. Tauira will now use the method for creating a provisional feature structure to be refined by a dialogue. The method first proposed by Erbach (1990) always possesed two major drawbacks:

- Filters have to be defined to ensure there is no overspecification.

- Lexical items still have to be refined even after filtering (underspecification).

As for the first drawback in the English Resource Grammar (ERG) as observed by Fouvry (2003), there has been a move from defining a word type and its features to defining a sub-word type using this feature. E.g. a lexical entry for a possessive intransitive noun is not defined as an intransitive noun (*n_intr_le*) with a possessive feature set to + but as a possessive intransitive noun (*n_intr_poss_le*) which is its own type and has the possessive feature set.

This methodology for defining word types effectively defines filters. A filter is after all nothing else then a description of features that need to be set. When no features need to be set for any type, no additional definition of filters is required. The system will therefore just need to add a disjunction of all open word classes as the unknown word entry and find out which unknown word type parses. The set of hypotheses generated by the system will therefor consist of all possible entries.

Erbach, Barg and Walther and Fouvry avoid using morphology. The Tauira system however will deal with some forms of morphology. The morphology the system will be able to handle is the regular morphology based on the rules defined by the grammar writer. These morphological rules define the inflection of words based on affixes and suffixes. E.g. for the ERG there is a rule for plurality inflecting *'dog'* to *'dogs'*, *'albatross'* to *'albatrosses'* but not *'analysis'* to *'analyses'*. Irregular words such as this latter example must be defined individually. These morphological rules are based on the original stem of a word. Possible stems for a word are found by applying these morphological rules backwards. This is done in the same manner as the sentence parser except for the fact that the sentence parser checks whether or not morphological stems exist in the lexicon and Tauira does not.

3.2.3 Dialogue based approaches for hypothesis reduction

The next step is reducing the number of hypotheses to one to create an unambiguous lexical entry. Barg and Walther (1998) create complete lexical entries by constantly updating incomplete lexical entries. Fourvry (2003) uses a part of speech tagger to calculate the probability of a certain unknown word. Statistical approaches are not neccessary for our system since we have the opportunity to consult the user about all uncertainities concerning the unknown word. Since we would like to be able to not only parse but also generate the sentence containing the unknown word (since Te Kaitito is a bidirectional system) we require Tauira to come up with a complete lexical entry.

I will now consider a number of different dialogue-techniques so that their advantages and disadvantages can be considered. In the final paragraph of this section I will motivate the chosen approach for the dialogue system.

Dialogue generated corpora

Analogous to the statistical method of Barg and Walther (1998) in Section 2.2.2 one could let the user provide various examples of the usage of the word until all required features are set. E.g. by the following dialogue:

```
H: The albatross chases the cat.
C: I don't know the word "albatross".
C: Could you give me another use of the word "albatross".
H: The cat chases the albatross
```

Although this it is fairly easy for the user to come up with new examples, it is difficult for the user to come up with all possible applications of the word within the grammar without having a linguistic background. Also the system would have to know if certain sentences are not allowed for a certain word. The fact that the word *'water'* is not personifiable can only be known by finding out a sentence like *'The water who is afraid'* is incorrect.

Linguistic multiple choice questions

Inspired by Knight's (2003) example a multiple choice test can decide the type for the word:

```
H: The albatross chases the cat.
C: I don't know the word "albatross".
```
The system knows it is a noun, based on the sentence context but does not know whether or not it is a personified noun
```
C: Can albatross be personified ?
H: Yes.
```

This method however directly violates the requirement that users require no prior knowledge of the system or linguistics. For this method to comply to this requirement the questions need to explain a lot more about the meaning of a syntactical property. It would however be impossible to generate these elaborate questions automatically. These would have to be described by a grammar writer with the ability to explain his grammar in layman's terms.

Syntactical synonyms multiple choice questions

Analogous to Knight's use of paraphrasing the user could specify a word with the same syntactical abilities as the unknown word.

```
H: The albatross chases the cat.
C: I don't know the word "albatross".
C: Is "albatross" more like "food" or "tree" ?
H: Tree.
```

These syntactical synonyms could be generated automatically without that much difficulty since they are already part of the lexicon. The drawback of this approach would be that users might have difficulty relating their word to the syntactic synonym without the knowledge of the grammar of the system. Letting the user choose syntactical synonyms freely (without multiple choice options) might not be practical since the words the user would come up with might not be in the lexicon.

Example phrases multiple choice questions

A way to comply with the requirement that the user need not be a linguist nor familiar with the system would be for the system to come up with example phrases for the user to accept or refuse.

```
H: The albatross chases the cat.
C: I don't know the word "albatross".
C: Is "The albatross who barks." a grammatically correct
sentence?
H: Yes.
```

The challenge however is generating these sentences automatically. Generating all possible sentences based on a grammar will provide a very large set of sentences to take examples from of which some will be unusable due to overgeneration, so hand made sentences would be preferable.

A potential source of example sentences is test suites. A test suite is applicable for use in example sentences because (a) it contains simple sentences and (b) it should provide an example for every phenomena described by the grammar (see Section 1.4). Naturally we have to make some assumptions about the coverage of the test suite: even using a very extended test suite there is still a change no example sentence can be generated for a specific hypothesis. In fact, the ability of the test suite to provide sentences which disambiguate between word types can be seen as a very usefull evaluation criteria (see Section 4.2.2).

Conclusion

Tauira will at first generate an example sentence for a word-type, because this appears to be the most user friendly of the given approaches. Preferably these sentences will be asked in a multiple choice fashion. If no example phrase can be generated the syntactical synonyms multiple questions approach is chosen since this is failsafe given a complete lexicon (in respect to the grammar).

3.2.4 Translation feature extraction

The next step is to find out the semantics for the unknown-lexical entry through its translation. Since translation was not the emphasis of the project we will adopt Te Kaitito's basic translation scheme and not consider more complicated translations such as in Copestake and Sanfillipo (2003).

Single word translation

A possibility for translation is to let the user specify the translation of the unknown word in a dialog such as:

```
H: How does "albatross" translate to Māori?
C: Toroa
```

There are however a few complications about this method. First of all it would be difficult to find out what part of speech the translation is. Māori adjectives can for example be translated as either verbs or adjectives in English. Of course a mapping of one source word type to possible translated word types could be derived from the lexicon to create a set of hypotheses for the unknown word.

Secondly this method does not allow space for more complex translations such as translating the English *'take this hammer'* to the Dutch *'pak deze hamer aan'* for *'take'*/*'aanpakken'*.

Sentence translation

Another method of getting the user to provide a translation is by letting the user translate the entire sentence it uttered the original word in. This bears close resemblance to the paraphrasing method of Knight (2003), as in the following example:

```
H: How does "The albatross chases the cat" translate to
Māori?
```

```
C: Ka whai te toroa i te ngeru.
```

First of all based on this sentence it is easy to create a set of hypotheses since this is one unknown word within a sentence the system should be able to parse. I.e. hypotheses for the translated word can be generated analogues to as they were generated for the source word.

Secondly this method does allow space for more complex translations since the translation is user-defined on a sentence level.

Conclusion

For the implementation the sentence translation method is chosen, since this is easy to integrate with the rest of the components and it gives way to more complex translations.

Chapter 4

Implementation

In this chapter the preprocessing and algorithm steps identified in Sections 3.1.3 and 3.1.4 are worked out in detail. The system was implemented in (Allegro) Commen Lisp as the original LKB system was implemented in this as well.

4.1 Setting open word types

Before any creation of test items can be done it needs to be decided which word types are considered as open. This can be done either by hand or by some heuristic. When writing a grammar for each new type added it can be decided whether or not this word type should be considered open. Proper names could be considered as one of the openest types while the determiner type should be considered closed. For a non-grammar writer however in a grammar such as the ERG consisting of over five hundred word types it can be quite difficult to decide which types are open. For this purpose a simple heuristic was defined: the number of occurrences in the lexicon. Table 4.1 illustrates the number of word types remaining relative to the minimal number of words required for a word type for the ERG (the **cut-off**).

For Tauira we have decided to take 10 as the cut-off effectively reducing the number open word types to 86. A global variable **unknown-word-types** is created, consisting of a list of all these types.

4.2 Creating test sentences

4.2.1 Preprocessing algorithm

Test sentences to use to query the user are created based on sentences from the test suite's set of sentences. These can be extracted from the tsdb database for test

Cut-off	Types
1	560
2	294
3	215
4	162
5	139
7	103
10	86
20	60
50	29
100	19
750	4
1000	2

Table 4.1: Cutt-off and remaining word types for the ERG

suites (Oepen, 2001). For each sentence in the test suite:

1. The sentence is parsed.
2. The word types and morphological rules for each word are extracted.
3. For each word type which is in *unknown-word-types* a new test sentence is created with a 'blank' position.
4. We invent a special nonexistent stem, spelled *'UNIWORDE'*. It has this spelling because every existing morphological rule can be applied to it. This stem is added to a temporary lexicon as each possible open word type. We then replace 'blank' with an appropriately inflected version of *'UNIWORDE'* and parse this sentence.
5. Word types of *'UNIWORDE'* that are used to create a parse of the sentence from the previous step are included in a set of ambiguous types called *all-types*.
6. Each test sentence (*sentence*) with its type (*type*), ambiguous types (*all-types*) and morphological inflectional rule (*morph-rule*) are written to a file as a **test-item**.

I will provide an example for a sentence from the MRS test suite of English using the ERG as a grammar. The sentence is *'The dog arrived barking'*. Step 1 will result in one possible parse.

Step 2: The determiner *'the'* is of type *det_the_le*, *'dog'* is an intransitive noun and *'arrived'* and *'barking'* are both unergative intransitive verbs. *'The'* and *'dog'* have not undergone morphological inflection (i.e. they have *nil*-morphology) while *'arrived'* and *'barking'* are inflected by the rules for verbs to be put in past and present participle form respectively.

Word	*'The'*	*'dog'*	*'arrived'*	*'barking'*
Type	*det_the_le*	*n_intr_le*	*v_unerg_le*	*v_unerg_le*
Morph-rule	*nil*	*nil*	*past_verb_infl_rule*	*prp_verb_infl_rule*

Step 3: All words except *'the'* are of a type in **unknown-word-types**. The following test sentences are created: *'The* blank *arrived barking.'*, *'The dog* blank *barking.'*, *'The dog arrived* blank*.'*.

Step 4: Each sentence is reparsed using the word *'UNIWORDE'*: *'The UNIWORDE arrived barking.'*, *'The dog UNIWORDED barking.'*, *'The dog arrived UNIWORDING.'*.

Step 5 en 6: After parsing the sentences from step 5 the following test-items are created.

Sentence	*'The* blank *arrived barking.'*
Type	*n_intr_le*
Morph-rule	*nil*
All-types	{*n_intr_le, n_holiday_le, n_mass_le, n_mass_count_le, n_mass_count_ppof_le, n_ppof_le, n_day_of_month_le, n_mass_ppcomp_le, n_month_le, n_plur_xmod_le, n_ppcomp_le, n_plur_le, n_year_le, n_plur_ppcomp_le, n_intr_temp_le, n_season_pp_le, n_day_of_week_le, n_proper_le*}
Sentence	*'The dog* blank *barking.'*
Type	*v_unerg_le*
Morph-rule	*past_verb_infl_rule*
All-types	{*v_unerg_le, v_np_trans_le, v_np*_trans_le, v_unacc_le, v_to*_trans_le*}
Sentence	*'The dog arrived* blank*.'*
Type	*v_unerg_le*
Morph-rule	*prp_verb_infl_rule*
All-types	{*v_unerg_le, v_np*_trans_le, v_unacc_le*}

'The blank *arrived barking'* is a very ambiguous sentence since the 'blank' can be replaced by 17 other types ranging from holidays to proper names. Naturally, since all grammars overgenerate, it will not be possible for a human to place every lexical item of every type in *all-types* into this test sentece and still end up with a

cohert sentence. In many cases the resulting sentence will be semanticly abnormal e.g. *'The easter weekend arrived barking'* In other cases it might even appear syntactically abnormal. e.g.(*'The William arrived barking'*). We will return to this in Section 4.5 when these test sentences are actually used.

'The dog blank *barking'* and even more so *'The dog arrived* blank*'* are far less ambiguous.

4.2.2 Evaluation

After a test suite has been preprocessed to create a set of test items of the kind just described the effectiveness of every test sentence as a sentence to use when querying the user can be evaluated. For this a number of benchmarks are defined:

1. The number of test-items for a certain word type. (If a word type is not used at least once, then it will be impossible to positively verify a hypotheses, if the unknown word is of this type.)

2. The (number of) **positively indistinguishable** word types for a certain word type. Calculated by taking the intersection of all *all-types* sets of all test items for a certain word type.

   E.g. If the only test-items for *v_unerg_le* are those created in the example of Section 4.2.1 then asking these questions would still leave the types {*v_unerg_le, v_np*_trans_le, v_unacc_le*} indistinguishable from each other and make up the set of positively indistinguishable types.

3. The (number of) **negatively indistinguishable** word types for a certain word type. This is calculated for word-type *wt1* by removing those word-types *wt2* from the positively indistinguishable word types for *wt1* that do not have *wt1* in the set of positively indistinguishable word types of *wt2*.

   E.g. If besides the two test items mentioned in the previous example, the test suite would have the following test-item preprocessed based on the sentence *'He interviewed them'*.

Sentence	*'He* blank *them.'*
Type	*v_np*_trans_le*
Morph-rule	*past_verb_infl_rule*
All-types	{*v_np*_trans_le, v_np_trans_le, v_to*_trans_le*}

   Then quering the user with this test item would result in the elimination of word type *v_np*_trans_le* if the word type is *v_unerg_le*. Negatively undistingishable types for *v_unerg_le* for a test suite consisting of the previously mentioned three tests are therefore {*v_unerg_le, v_unacc_le*}.

We have preprocessed the MRS test suite consisting of 107 English test sentences provided with the English Resource Grammar. A cut-off of 10 was chosen to select the open word types. The resulting benchmarks are displayed in Table 4.2.

Since these results left a lot of indistinguishable types the CSLI test suite also provided with the ERG was preprocessed. Since it consisted of 961 wellformed sentences it was cut into 20 files of 50 sentences which were preprocessed individually. Of these 20 files only 6 were preprocessed successfully. In the other files there were sentences too complex for the preprocessor to handle so it ran out of memory. (Fouvry (2003) also observes this problem in his implementation of an unknown word processing system for the ERG on the LKB system.) The resulting test-items were added to those preprocessed based on the MRS and the resulting benchmarks are shown in Table 4.3.

These benchmarks provide a very nice formal illustration of the shortcomings of a test suite. In the MRS test suite for example there is no test sentence distinguishing transitive from intransitive verbs or distinguishing unergative intransitive from unaccusative intransitive verbs. In fact these benchmarks are very useful to help a grammar writer choose new sentences to add to the test suite. This is a merrit for future works (see Section 6.2.2).

This concludes the preprocessing of the test suite. The remainder of this chapter describes in detail how Tauira deals with unknown words when they arise in a dialogue.

4.3 Locating unknown words

The LKB system, when appropriately patched, will return which words are unknown for the current lexicon when a sentence is parsed by the LKB's parser. If there are unknown words Tauira will initiate a subdialoge. The first step in this dialogue is to make sure there is one unknown lexical entry per sentence to be resolved in the rest of the algortihm. If consecutive words are unknown the user is consulted whether or not these words make up one lexical item.

If there is more than one unknown lexical item the user will be requested to make a sentence using just this unknown word for each unknown word except the last unknown word (since by that time all other unknown words are known).

```
H: The yellow-eyed penguin chases the albatross.
C: I do not understand yellow-eyed, penguin and albatross.
Is yellow-eyed penguin one word ?
H: Yes.
```

Name (Type)	Occ	Tests	Poss-indistinguishable Types	Neg-indistinguishable Types
Intransitive vp post adverb (*adv_int_vp_post_le*)	142	0		
np* transitive verb (*v_np*_trans_le*)	196	22	{*v_np*_trans_le, v_np_trans_le*}	{*v_np*_trans_le, v_np_trans_le*}
Energusative Intransitive verb (*v_uner_le*)	215	71	{*v_unerg_le, v_unacc_le*}	{*v_unerg_le, v_unacc_le*}
Particle np verb (*v_particle_np_le*)	217	2	{*v_particle_np_le*}	{*v_particle_np_le*}
Intransitive vp adverb (*adv_int_vp_le*)	363	0		
np transitive verb (*v_np_trans_le*)	676	6	{*v_np_trans_le, v_np*_trans_le, v_to*_trans_le*}	{*v_np_trans_le, v_np*_trans_le, v_to*_trans_le*}
proper name (*n_proper_le*)	837	66	{*n_proper_le, n_intr_le, n_mass_count_le, n_mass_le, n_mass_count_ppof_le, n_proper_abb_le, n_mass_ppcomp_le*}	{*n_proper_le, n_mass_count_le, n_mass_le, n_mass_count_ppof_le, n_proper_abb_le*}
Intransitive noun (*n_intr_le*)	983	69	{*n_intr_le, n_ppcomp_le, n_mass_count_le, n_mass_count_ppof_le, n_ppof_le*}	{*n_intr_le, n_ppcomp_le, n_mass_count_le, n_mass_count_ppof_le*}
(Intransitive) pp of noun (*n_ppof_le*)	1057	8	{*n_ppof_le, n_mass_count_ppoff_le, n_holiday_le, n_ppcomp_le, n_day_of_week_le*}	{*n_ppof_le, n_mass_count_ppoff_le, n_holiday_le, n_ppcomp_le, n_day_of_week_le*}
Intransitive adjective (*adj_intrans_le*)	1223	8	{*adj_intrans_le, adj_trans_le*}	{*adj_intrans_le, adj_trans_le*}

Table 4.2: Benchmarks for MRS test suite

Name (Type)	Occ	Tests	Poss-indistinguishable Types	Neg-indistinguishable Types
Intransitive vp post adverb (*adv_int_vp_post_le*)	142	3	{*adv_int_vp_post_le, adv_le, adv_vp_le, adv_int_vp_le*}	{*adv_int_vp_post_le, adv_le, adv_vp_le*}
np* transitive verb (*v_np*_trans_le*)	196	85	{*v_np*_trans_le*}	{*v_np*_trans_le*}
Energusative Intransitive verb (*v_uner_le*)	215	156	{*v_unerg_le, v_unacc_le*}	{*v_unerg_le, v_unacc_le*}
Particle np verb (*v_particle_np_le*)	217	4	{*v_particle_np_le*}	{*v_particle_np_le*}
Intransitive vp adverb (*adv_int_vp_le*)	363	14	{*adv_int_vp_le, n_ppof_le, n_mass_count_le, n_mass_le, n_mass_ppcomp_le, n_plur_xmod_le, n_ppcomp_le, adv_vp_le, adv_le, n_day_of_week_le, n_generic_pro_adv_le, n_intr_temp_le*}	{*adv_int_vp_le, n_mass_count_le, n_mass_ppcomp_le, n_plur_xmod_le, adv_vp_le, n_generic_pro_adv_le, adv_le, n_intr_temp_le*}
np transitive verb (*v_np_trans_le*)	676	136	{*v_np_trans_le, v_np*_trans_le, v_to*_trans_le*}	{*v_np_trans_le, v_to*_trans_le*}
proper name (*n_proper_le*)	837	366	{*n_proper_le, n_intr_le*}	{*n_proper_le*}
Intransitive noun (*n_intr_le*)	983	134	{*n_intr_le, n_ppcomp_le, n_mass_count_le, n_mass_count_ppof_le, n_ppof_le*}	{*n_intr_le, n_ppcomp_le, n_mass_count_le, n_mass_count_ppof_le*}
(Intransitive) pp of noun (*n_ppof_le*)	1057	100	{*n_ppof_le, n_ppcomp_le*}	{*n_ppof_le, n_ppcomp_le*}
Intransitive adjective (*adj_intrans_le*)	1223	19	{*adj_intrans_le, adj_trans_le*}	{*adj_intrans_le, adj_trans_le*}

Table 4.3: benchmarks for MRS test stuite extended with 300 CSLI test sentences

```
C: Could you make a sentence with 'yellow-eyed penguin' and
words I understand.
H: The yellow-eyed penguin barks.
```

The system can now resolve the unknown word *'yellow-eyed penguin'* in the sentence *'the yellow-eyed penguin barks'* and *'albatross'* in the *'The yellow-eyed penguin chases the albatross'*.

4.4 Creating hypotheses

Once it is certain there is one unknown lexical entry in each sentence a set of hypotheses for this unknown lexical entry can be created based on the sentence context. As explained in Section 3.1.1 a hypothesis is a pair of a word type and stem. Hypotheses are creating using a generate and test algorithm. The generation space consists of each open word type and stem. Possible stems are created using the morphological rules defined in the grammar. E.g. for the ERG *'jogged'* would have *'jog'*, *'jogg'*, *'jogge'* or *'jogged'* as possible stems, while *'penguin'* would just have *'penguin'* as a possible stem. Testing is done by adding the different hypotheses to the lexicon and selecting those that result in successful parses.

Suppose the only open word types are unergative intransitive verbs (*v_unerg_le*) and proper names *n_proper_le*) and the user has entered the sentence *'He is hawking the cat'* with unknown word *'hawking'*. The generation space is now {*(v_unerg_le, 'hawk'), (v_unerg_le, 'hawke'), (v_unerg_le, 'hawking'), (n_proper_le, 'hawk'), (n_proper_le, 'hawke'), (n_proper_le, 'hawking')*}. Suppose the grammar would both allow sentences such as *'He is walking the cat'* and *'He is Garfield the cat'*. After testing the generation space would then be reduced to: {*(v_unerg_le, 'hawk'), (v_unerg_le, 'hawke'), (n_proper_le, 'hawking')*}

4.5 Eliminating hypotheses

Choosing a word type is done by consulting the user. If available an example phrase based on one from the test suite is created, otherwise a list of syntactical synonyms is provided. To accomplish this the following strategies are attempted in chronological order.

4.5.1 Multiple choice example phrases strategy

This strategy is tried first but requires there is only one stem for each word type.

1. For the *hypothesis* with the highest *occurrence* in the lexicon the *test* is selected where the intersection between *hypotheses* and the ambiguous types (*test.all-types*) is the smallest.

2. For *hypotheses* that do not appear *test.all-types* of used *test*'s step 1 is repeated until no more adequate tests can be found.

3. For each *test* the morphological rule (*test.morph-rule*) is applied to the hypothesised stem and the result is inserted into the test sentence (*test.sentence*).

The different tests will be asked as a multiple choice question of the form:

```
Which sentence is correct ?
1.  applied sentence for test 1
2.  applied sentence for test 2
3.  None.
```

If only one test is found a question of the following form is asked:

```
Is 'applied sentence for test 1' a correct sentence ?
```

If the user answers by picking one of the example phrases *hypotheses* is set to those with types of the ambiguous types of that test item. If the user responds that none of the sentences are correct all the original types of the test-sentences will be removed. The ambiguous types are not removed as hypotheses since as explained in Section 4.2 these may not make sense for the user although they will be parsed by the grammar. (Recall the example *'the William arrived barking'*.)

Suppose the user enters *'She walks'*, where the word *'walks'* is unknown; the user will be left with the type-hypotheses {*v_unerg_le, v_np*_trans_le, v_unacc_le, va_modal_neg_le*} meaning the word could be an intransitive verb (both unergative and unaccusative), a transitive verb or a negative modal expression (e.g. *'she needn't'*).

The first test to be selected is based on the test suite sentence *'the dog arrived and barked'* since *'barked'* is an unergative intransitive verb (and *v_unerg_le* is most frequent in the lexicon) and a word replacing *'barked'* in this sentence can by either an unergative or an unaccusative verb. If the user accepts this phrase the unknown word could be either one of these types. If the user does not select this sentence only *v_unerg_le* will be removed as a hypothesis since there is no way to be sure whether or not it will make sense for a non accusative verb.

Since *v_unerg_le* and *v_unacc_le* are already used in an alternative the algorithm will attempt to find a test for either *v_np*_trans_le* or *va_modal_neg_le*. The test

based on *'He interviewed them'* has three different transitive verbs ({*v_np*_trans_le, v_np_trans_le, v_to*_trans_le*}) as ambiguous types but only one of these is a hypothesis, so it is a good candidate. No test for *va_modal_neg_le* can be found. Based on these tests the following multiple choice question can be asked to the user. (The corresponding remaining hypotheses types are shown in curly braces for each answer).

```
Which sentence is correct?
1 The dog arrived and walked.  {v_unerg_le, v_unacc_le}
2 He walked them.  {v_np*_trans_le}
3 None {v_unacc_le, va_modal_neg_le}
```

4.5.2 Single choice example phrases strategy

This strategy is tried secondly if the multiple choice example phrases strategy does not succeed. Stems are ordered with the shortest stem first (e.g. *'bark'* before *'barked'*), not only because this is will out rule any ambiguity since morphological rules always extend the stem. For example *'I walked'* can be both accepted for the stem *'walked'* (present tense, e.g *'I bobsled'*) as *'walk'* (past tense). But *'I walk'* can only be accepted for the stem *'walk'*.

1. For the *hypothesis* with the highest *occurrence* in the lexicon the *test* is selected where the intersection between the other hypotheses and the ambiguous types *test.all-types* is the smallest.

2. The user is asked a yes-no question about the grammaticality of the sentence of the test item applied to the shortest stem for its type. The shortest stem is chosen from multiple stems because this will correctly filter out the different morphological analyses. For example, asking wether or not the sentence *'I bobsled'* is correct for *'bobsled'* can be answered affirmatively both when interpreting this as the verb *'bobsle'* in the past tense or the verb *'bobsled'* in the present tense. When applying the shortest stem first the sentence *'I bobsle'* is asked to the user which can only be interpreted as the verb *'bobsle'* in the present tense.

3. If the user accepts the sentence only hypotheses remain that a) have a type in *test.all-types* (like the previous strategy) and b) have a stem which can be inflected to the form as appeared in the sentence queried to the user (e.g. both *'chas'* and *'chase'* can be inflected to *'chased'* for the sentence *'the dog arrived and chased'*) . If the user declines the sentence, hypotheses of

test.type and having a stem which can be inflected to the form as appeared in the sentence queried to the user are removed.

For example let's say that in stead of the phrase *'She walks'* the user has entered *'I walked'*, where *'walked'* is the unknown word. Possible hypotheses now are {*(v_unerg_le, 'walk'), (v_unerg_le, 'walke'), (v_unerg_le, 'walked'), (v_np*_trans_le, 'walk'),(v_np*_trans_le, 'walke'), (v_np*_trans_le, 'walked'), (v_unacc_le, 'walk'), (v_unacc_le, 'walke'), (v_unacc_le, 'walked'), (va_modal_neg_le, 'walked')*} . Every type has three different stems since the phrase can be in past as well as present tense (with *'walked'* being the past tense inflection of either the verb *'walke'* or *'walk'*). This does not apply for *va_modal_neg_le* since there is no inflection on this type in this context (e.g. Only *'I needn't'* is possible for stem *'needn't'*).

The same test (*'the dog arrived and barked'*) is selected and consulted to the user with the shortest stem for *v_unerg_le*.

```
Is 'the dog arrived and walked' a correct sentence ?
```

If the user replies yes, *'walked'* can be removed as possible stem since then the sentence would read *'the dog arrived and walkedded'*. Hypotheses of other types than *v_unerg_le* or *v_unacc_le* are removed. The resulting hypotheses then are: {*(v_unerg_le, 'walk'), (v_unerg_le, 'walke'), (v_unacc_le,'walk'), (v_unacc_le, 'walke')*} If the user replies 'no', only *(v_unerg_le, 'walk'* and *(v_unerg_le, 'walke')* are removed as possible hypotheses, since there is no way of knowing wether the rejection of the sentence is due to the stem or the word type.

4.5.3 Multiple choice example words strategy

If the example phrases strategies fail because no applicable test items can be found the user left to choose between examples from the lexicon for a specific word. The only hypotheses remaining after this example are those corresponding to the example types. For the previous example, if there would be no more example phrases the system would ask examples for unergative and unaccusative intransitive verbs.:

```
Which of these words is most like your word ?
1.  'arise', 'withdraw', 'cook'
2.  'flop', 'intervene', 'linger'
```

4.5.4 Multiple choice stem selection strategy

If after the previous three strategies the stem still remains ambiguous, the user is directly consulted about the stem. After this strategy there will certainly be only one hypothesis remaining. For our example this would be:

```
What is the stem of 'walked' ?
1.  walk
2.  walke
```

4.6 Translation request

After the authoring process for a word is complete a translation for the original input sentence is requested from the user. Since this new sentence possesses only new words due to the original new word the semantics of the translated word are trivial. A the moment the algortihm allows three possiblities for translation:

- The word in translation and source is the same, e.g. for proper names.
- An existing word is in the translation for the source word.
- A new word (or set of consecutive words) is in the translation for the source word.

If the latter is the case then based on this new sentence steps 1 to 3 from the original algorithm (Section 3.1.4) are revisited for the translated word.

4.7 Finalization

After the lexical entries for both words are known they are given the semantic predicate (derived from the orthography of the source word) and are added to the lexicon. The addition of new words to the lexicon dynamicly was quite an undertaking since the LKB system was designed with the thought that all words would be loaded at the start of a session (either from a file or database).

The new lexical entries will also be re-indexed so the system is able to paraphrase (for a bilingual grammar translate) the sentence. The original input sentence can now be parsed by the original system (e.g. the dialogue or translation system) and the original dialogue can now resume.

Chapter 5

Results

In this chapter the results of Tauira will be illustrated be means of a set of typical examples. In Section 5.1 two examples for a unknown word with many hypotheses will be given. In Section 5.2 an example which illustrates the morphological disambiguiation will be given. These examples were generated running Tauira on the ERG using the preprocessed test suite described in Section 4.2.2 and Illustrated in Table 4.3.

Section 5.3 illustrates how Tauira deals with multiple words and translation. This example was generated running Tauira on Te Kaitito's bilingual grammar using the test suite from Table 4.3 extended with test items preprocessed from a set of Māori sentences. Te Kaitito uses the same English word types as the ERG.

For each example the remaining set of hypotheses and an identifier for each question are shown. These are not visible to the user in normal systems operation.

5.1 High ambiguity example

These examples demonstrate how hypotheses are reduced for a very large set of hypotheses. The sentence *'my name is Maarten'* yields a large set of hypotheses for unknown word *'Maarten'* and trivially *'my name is dutch'* yields the same set of hypotheses but for the unknown word *'dutch'*.

5.1.1 Proper name

'Maarten' has no morphological inflection so only pairs with the stem *'maarten'* are generated for the sentence *'my name is maarten'*. The multiple choice example phrases strategy is used for Question *Q1.1*. Alternative 1 tests intransitive adjectives (the most frequent word type), Alternative 2 tests for nouns allowing the *'.. of ..'* coustruction. Because all intransitive noun tests also parse for the nouns of

alternative 2 as seen in Table 4.3, there will be no alternative generated within this question for intransitive nouns. So alternative 3 is a sentence for the next probable type, namely the proper name, which is the alternative user will select. Questions *Q1.2* and *Q2.2* eliminate the hypotheses which also parse for alternative 3 using the single choice example phrases strategy.

```
> my name is maarten
```

hypotheses = {*(adj_intrans_le, 'maarten'), (n_ppof_le, 'maarten'), (n_intr_le, 'maarten'), (n_proper_le, 'maarten'), (adv_int_vp_le, 'maarten'), (adv_int_vp_post_le, 'maarten'), (n_mass_count_le, 'maarten'), (n_mass_le, 'maarten'), (adj_bare_unspecified_card_two_le, 'maarten'), (n_mass_count_ppof_le, 'maarten'), (adv_disc_le, 'maarten'), (adj_trans_le, 'maarten'), (adj_comp_le, 'maarten'), (n_proper_abb_le, 'maarten'), (n_mass_ppcomp_le, 'maarten'), (adj_superl_prd_le, 'maarten'), (n_year_le, 'maarten'), (n_plur_xmod_le, 'maarten'), (n_ppcomp_le, 'maarten'), (n_plur_le, 'maarten'), (adv_vp_post_le, 'maarten'), (n_month_le, 'maarten'), (pp_le, 'maarten'), (n_hour_le, 'maarten'), (adj_bare_unspecified_card_one_le, 'maarten'), (n_adv_le, 'maarten'), (adv_vp_aux_le, 'maarten'), (adv_vp_le, 'maarten'), (n_day_of_week_le, 'maarten'), (n_generic_pro_adv_le, 'maarten'), (adj_pred_intrans_le, 'maarten'), (n_approx_hour_le, 'maarten'), (adv_le, 'maarten'), (n_generic_pro_le, 'maarten'), (n_plur_ppcomp_le, 'maarten'), (n_season_pp_le, 'maarten'), (n_intr_temp_le, 'maarten')*}

Q1.1 I do not understand the word MAARTEN.
Which of these phrases illustrates the correct use of
MAARTEN ?
1. how maarten was abrams
2. the maartens of the projects are trustworthy
3. mr maarten browne is the manager
4. there are five maarten in the room aren t there
5. None
> 3

hypotheses = {*(n_intr_le, 'maarten'), (n_proper_le, 'maarten'), (n_mass_count_le, 'maarten'), (n_mass_le, 'maarten')*}

Q1.1 Is 'five of the seven maartens work for abrams' a
correct sentence ?
> no

hypotheses = {*(n_proper_le, 'maarten'), (n_mass_count_le, 'maarten'), (n_mass_le, 'maarten')*}

Q1.2 Is 'maarten knows that sara will sleep doesn t he' a
correct sentence ?
> yes

hypotheses = {*(n_proper_le, 'maarten')*}

...

5.1.2 Adjective

For the sentence *'my name is dutch'* the set of hypotheses and the first question (*Q1.4*) is the same as Question *Q1.1* except for the word spelling. The user will now however select alternative 1. A lot of adjectives and adverbs however also parse for this alternative, so the system comes up with another multiple choice question (*Q.1.5*). As shown in Table 4.3 with the currently processed part of the test suite no distiction between transitive and intransitive adjectives can be made. Therefore Question *Q.1.6* provides a number of example words to choose from. The helpful *'European'* example assists the user in its choice for alternative 1.

```
> my name is dutch
```

hypotheses = {*(adj_intrans_le, 'dutch'), (n_ppof_le, 'dutch'), (n_intr_le, 'dutch'), (n_proper_le, 'dutch'), (adv_int_vp_le, 'dutch'), (adv_int_vp_post_le, 'dutch'), (n_mass_count_le, 'dutch'), (n_mass_le, 'dutch'), (adj_bare_unspecified_card_two_le, 'dutch'), (n_mass_count_ppof_le, 'dutch'), (adv_disc_le, 'dutch'), (adj_trans_le, 'dutch'), (adj_comp_le, 'dutch'), (n_proper_abb_le, 'dutch'), (n_mass_ppcomp_le, 'dutch'), (adj_superl_prd_le, 'dutch'), (n_year_le, 'dutch'), (n_plur_xmod_le, 'dutch'), (n_ppcomp_le, 'dutch'), (n_plur_le, 'dutch'), (adv_vp_post_le, 'dutch'), (n_month_le, 'dutch'), (pp_le, 'dutch'), (n_hour_le, 'dutch'), (adj_bare_unspecified_card_one_le, 'dutch'), (n_adv_le, 'dutch'), (adv_vp_aux_le, 'dutch'), (adv_vp_le, 'dutch'), (n_day_of_week_le, 'dutch'), (n_generic_pro_adv_le, 'dutch'), (adj_pred_intrans_le, 'dutch'), (n_approx_hour_le, 'dutch'), (adv_le, 'dutch'), (n_generic_pro_le, 'dutch'), (n_plur_ppcomp_le, 'dutch'), (n_season_pp_le, 'dutch'), (n_intr_temp_le, 'dutch')*}

```
I do not understand the word DUTCH.
Q1.4 Which of these phrases illustrates the correct use of
DUTCH ?
1.  how dutch was abrams
2.  the dutches of the projects are trustworthy
3.  mr dutch browne is the manager
4.  there are five dutch in the room aren t there
5.  None
> 1
```

hypotheses = {*(adj_intrans_le, 'dutch'), (adv_int_vp_le, 'dutch'), (adv_int_vp_post_le, 'dutch'), (adj_bare_unspecified_card_two_le, 'dutch'), (adj_trans_le, 'dutch'), (adj_superl_prd_le, 'dutch'), (n_hour_le, 'dutch'), (adj_bare_unspecified_card_one_le, 'dutch'), (adv_vp_le, 'dutch'), (adj_pred_intrans_le, 'dutch'), (adv_le, 'dutch'), (n_approx_hour_le, 'dutch')*}

Q1.5 `Which of these phrases illustrates the correct use of DUTCH ?`

```
1.  abrams works for a dutch manager
2.  how dutch does abrams interview a programmer
3.  dutch hundred twenty dogs bark
4.  None
> 1
```

hypotheses = {*(adj_intrans_le, 'dutch'), (adj_trans_le, 'dutch')*}

Q1.6 `Which of these words is most like your word ?`

```
1.  'european', 'stupid', 'gusty'
2.  'uncertain', 'invisible', 'aware'
> 1
```

hypotheses = {*(adj_intrans_le, 'dutch')*}

...

5.2 Morphology example

The sentence *'I danced'* generates three possible morphological stems: *'danc'*, *'dance'* and *'danced'*. Within the sentence these morphological stems can be all be linked to three verb types (two intransitive and one transitive verb). For the negative modal verb type (*va_modal_neg_le*, with stems such as *'needn't'*) only the orginal spelling is possible as a verb type.

Question *Q2.1* is a test for transitive verbs. Since stems *'danc'* and *'dance'* are both inflected to *'danced'* accepting this sentence is therefore accepting these two stems (and the word-types parsing for this sentence).

Question *Q2.2* is just used for stem checking; in this case the shortest stem *'danc'* is not the correct one, and refusing it will only remove the pair with the type tested (*v_unerg_le*). Since there is no test distinguishing between unergative and unaccusative verbs a list of syntactical synonyms is given in Question *Q2.3*. Unlike Question *Q1.6* this might be a bit harder to answer, though *'dance'* is semantically and syntactically close to *'stand'*.

```
> I danced
```

hypotheses = {*(v_unerg_le, 'danc'), (v_unerg_le, 'dance'), (v_unerg_le, 'danced'), (v_np*_trans_le, 'danc'), (v_np*_trans_le, 'dance'), (v_np*_trans_le, 'danced'), (v_unacc_le, 'danc'), (v_unacc_le, 'dance'), (v_unacc_le, 'danced'), (va_modal_neg_le, 'danced')*}

Q2.1 `I do not understand the word DANCED.`
`Is 'the dog arrived and danced' a correct sentence ?`
`> yes`

hypotheses = {*(v_unerg_le, 'danc'), (v_unerg_le, 'dance'), (v_unacc_le, 'danc'), (v_unacc_le, 'dance')*}

Q2.2 `Is 'abrams dancs for browne' a correct sentence ?`
`> no`

hypotheses = {*(v_unerg_le, 'dance'), (v_unacc_le, 'danc'), (v_unacc_le, 'dance')*}

Q2.3 `Which of these words is most like your word ?`
`1. 'stand', 'orientate', 'bay'`
`2. 'bust', 'grow', 'blow'`
`> 1`

hypotheses = {*(v_unerg_le, 'dance')*}

...

5.3 Bilingual Multiple word example

The example in this section illustrates the use of multiple words and translation of words. Since *'yelloweyed penguin'* appears as a consecutive clause within *'the albatross chases the yelloweyed penguin'* the user is consulted in Question *Q3.1* whether or not this makes up one lexical entry, which it does. Since this still leaves two unknown words in the sentence the user is asked to make a sentence with only the unknown word *'albatross'*. This generates a set of hypotheses which are reduced in Question *Q3.3* to *Q3.5*. It is interesting to mention that alternative 1 of Question *Q.3.3* is incorrect because the ERG has a specific feature deciding whether or not a noun can be compared to be *'of'* something.

From Question *Q3.6* the translation for *'albatross'* is devised and based on the sentence a set of hypotheses is created. This set is small because the number of word types for the Māori grammar are small. In Question *Q3.7* the number of hypotheses are reduced, and both *'albatross'* and its translation in Māori are resolved.

Question *Q3.8* to *Q3.10* now resolves *'Yelloweyed penguin'* which is treated as one lexical entry and the only unknown word in the sentence *'the albatross chases the yelloweyed penguin'*. In Question *Q3.11* the translation of *'yelloweyed penguin'* is resolved and the original sentence can be translated.

`> the albatross chases the yelloweyed penguin`

Q3.1 I do not understand the words ALBATROSS, YELLOWEYED, PENGUIN. Is YELLOWEYED PENGUIN one word ?
> yes
Q3.2 Could you provide me an example sentence with the unknown word 'ALBATROSS' using words I know ?
> the albatross eats

hypotheses = {*(n_ppof_le, 'albatross'), (n_intr_le, 'albatross'), (n_proper_le, 'albatross'), (n_mass_count_le, 'albatross'), (n_mass_le, 'albatross'), (n_mass_count_ppof_le, 'albatross'), (n_day_of_month_le, 'albatross'), (n_holiday_le, 'albatross'), (n_mass_ppcomp_le, 'albatross'), (n_year_le, 'albatross'), (n_ppcomp_le, 'albatross'), (n_month_le, 'albatross'), (n_day_of_week_le, 'albatross'), (n_season_pp_le, 'albatross'), (n_intr_temp_le, 'albatross')*}

Q3.3 Which of these phrases illustrates the correct use of ALBATROSS ?
1. the albatrosses of the projects are trustworthy
2. mr albatross browne is the manager
3. browne was hired on january albatross 1984
4. None
> 4

hypotheses = {*(n_intr_le, 'albatross'), (n_mass_count_le, 'albatross'), (n_mass_le, 'albatross'), (n_mass_count_ppof_le, 'albatross'), (n_mass_ppcomp_le, 'albatross'), (n_holiday_le, 'albatross'), (n_year_le, 'albatross'), (n_ppcomp_le, 'albatross'), (n_month_le, 'albatross'), (n_day_of_week_le, 'albatross'), (n_season_pp_le, 'albatross'), (n_intr_temp_le, 'albatross')*}

Q3.4 Is 'he showed her an albatross' a correct sentence ?
> yes

hypotheses = {*(n_intr_le, 'albatross'), (n_mass_count_le, 'albatross'), (n_mass_count_ppof_le, 'albatross'), (n_ppcomp_le, 'albatross')*}

Q3.5 Which of these words is most like your word ?
1. 'sec', 'counter narcotics', 'paycheck'
2. 'school', 'black', 'distance'
3. 'delivery', 'record', 'treatment'
4. 'specification', 'subscriber', 'deviation'
> 1

hypotheses = {*(n_intr_le, 'albatross')*}

Q3.6 What is the translation of 'the albatross eats' in Māori.
> ka kai te toroa

hypotheses = {*(m-noun-lxm, 'toroa'), (m-aha-noun-lxm, 'toroa')*}

Q3.7 Is 'kua pai te toroa' a correct sentence ?
> yes

hypotheses = {*(m-noun-lxm, 'toroa')*}

hypotheses = {*(n_ppof_le, 'yelloweyed penguin'), (n_intr_le, 'yelloweyed penguin'), (n_proper_le, 'yelloweyed penguin'), (n_mass_count_le, 'yelloweyed penguin'), (n_mass_le, 'yelloweyed penguin'), (n_mass_count_ppof_le, 'yelloweyed penguin'), (n_day_of_month_le, 'yelloweyed penguin'), (n_mass_ppcomp_le, 'yelloweyed penguin'), (n_holiday_le, 'yelloweyed penguin'), (n_year_le, 'yelloweyed penguin'), (n_plur_xmod_le, 'yelloweyed penguin'), (n_ppcomp_le, 'yelloweyed penguin'), (n_plur_le, 'yelloweyed penguin'), (n_month_le, 'yelloweyed penguin'), (n_day_of_week_le, 'yelloweyed penguin'), (n_plur_ppcomp_le, 'yelloweyed penguin'), (n_season_pp_le, 'yelloweyed penguin'), (n_intr_temp_le, 'yelloweyed penguin')*}

Q3.8 Which of these phrases illustrates the correct use of YELLOWEYED PENGUIN ?
1. the yelloweyed penguins of the projects are trustworthy
2. mr yelloweyed penguin browne is the manager
3. browne was hired on january yelloweyed penguin 1984
4. there are five yelloweyed penguin in the room aren t there
5. None
> 5

hypotheses = {*(n_intr_le, 'yelloweyed penguin'), (n_mass_count_le, 'yelloweyed penguin'), (n_mass_le, 'yelloweyed penguin'), (n_mass_count_ppof_le, 'yelloweyed penguin'), (n_holiday_le, 'yelloweyed penguin'), (n_mass_ppcomp_le, 'yelloweyed penguin'), (n_year_le, 'yelloweyed penguin'), (n_plur_xmod_le, 'yelloweyed penguin'), (n_ppcomp_le, 'yelloweyed penguin'), (n_plur_le, 'yelloweyed penguin'), (n_month_le, 'yelloweyed penguin'), (n_day_of_week_le, 'yelloweyed penguin'), (n_season_pp_le, 'yelloweyed penguin'), (n_intr_temp_le, 'yelloweyed penguin')*}

Q3.9 Is 'list yelloweyed penguins who have bookcases' a correct sentence ?
> yes

hypotheses = {*(n_intr_le, 'yelloweyed penguin'), (n_mass_count_le, 'yelloweyed penguin'), (n_mass_count_ppof_le, 'yelloweyed penguin'), (n_ppcomp_le, 'yelloweyed penguin')*}

Q3.10 Which of these words is most like your word ?
1. 'sec', 'counter narcotics', 'paycheck'
2. 'school', 'black', 'distance'
3. 'delivery', 'record', 'treatment'
4. 'specification', 'subscriber', 'deviation'
> 1

hypotheses = {*(n_intr_le, 'yelloweyed penguin')*}

Q3.11 What is the translation of 'the albatross chases the yelloweyed penguin' in Māori.
> ka whai te toroa i te hoiho

hypotheses = {*(m-noun-lxm, 'hoiho')*}

'the albatross chases the yelloweyed penguin' translates to:
ka whāia te hoiho e te toroa
ka whai te toroa i te hoiho

Chapter 6

Summary and future works

The first part of this chapter summarizes the main limitations (Section 6.1) and the various gains (Section 6.2) of the project.

The second part explores future works to be done to improve the system (Section 6.3) and further expand its functionality (Section 6.4). Finally future applications of Tauira are discussed (Section 6.5).

6.1 Project limitations

The quality of Tauira's natural language dialogue suffers from two major limitations, explained in this Section. First of all the questions asked to the user rely on the quality of the sentences of the original test suite (Section 6.1.1); secondly the questions are ristricted by the grammar used (Section 6.1.2).

6.1.1 Test suites

In Section 1.4 it is mentioned that one of the characteristics of a test suite is that every grammatical phenomenon is covered systematically. If we see word types as grammatical phenomena then it can be seen from Table 4.2 that this characteristic does not apply to the MRS test suite. The first benchmark for test suites, the number of times a word type occurs in the of tests suite, rates 0 for 2 out of the ten most frequent word types defined by the ERG. Of the ten most frequent word types that are used in the MRS test suite only one is used such a way that it is indistinguishable from any other word type (*v_particle_np_le* thanks to the test sentence *'Browne squeezed the cat in'*).

The CSLI test suite is over nine times larger and should be able to cover more word types. Because of the complexity (and not the multitude!) of the sentences

preproccessing becomes too complex. Fouvry (2003) mentions this in his paper when he attempts unknown word analysis of long sentences. Table 4.3 does provide encouraging results based on a total of 400 test sentences that were preprocessed. Of the ten most frequent words there are two uniquely distinguishable and eight are only indistinguishable from up to one other word type.

For a smaller grammar such as Te Kaitito's Māori grammar the a small test suite is already enough to distinguish between different types. Questions using example sentences will also not show up a lot of the time within the dialogue since the original set of hypotheses created is quite small.

6.1.2 Grammar

The usefulness of Tauira's dialogue is dependent on the grammar used. Sometimes the grammar is too generalized (*overgeneration*); the ERG for example does not distinguish between male and female proper names which means the ERG will accept the sentence *'Mary knows that Sara will sleep doesn't he ?'* . When the user is asked whether or not the previous sentence is correct he or she should answer affirmatively, against her or his natural intuition. In other instances the grammar is too restrictive for a user (*undergeneration*). The ERG will allow the construction *'The managers of the projects are trustworthy'* for *'manager'* but not for *'dog'*. This although searching for the phrase *'dogs of war'* returns over 137.000 hits from the Google internet search engine. So in order for *'albatross'* to be of the same lexical entry as *'dog'* the phrase *'the albatrosses of the project are trustworthy'* must be refused, which again can go against the user's intuition.

Overgeneration of a grammar can be tested by randomly selecting words of the same word-type as a word in a sentence and replacing this word with the original word and see if this still leaves an acceptable sentence. For instance replacing *'Bob'* with *'Mary'* in *'Bob knows that John will sleep doesn't he ?'*. Undergeneration of a grammar can be tested by placing a word of different word type then the original wordtype in a sentence and see if this still leaves an acceptable sentence for instance replacing *'dog'* with *'manager' 'The managers of the projects are trustworthy'*. The testing method for undergeneration could be restricted to wordtypes differing only in a limited number of features from the original word type (preventing verb types replacing noun types).

6.2 Project Gains

In this section the gains of the project are summarized: the functionality of the LKB system has been extended (Section 6.2.1); Evaluation criteria for grammars

and test suites have been established (Section 6.2.2); Unknown word processing techniques have been improved (Section 6.2.3) and a lexical acquisition tool that operates within a (bilingual) dialogue has been created (Section 6.2.4).

6.2.1 LKB system

As described earlier the LKB system was not developed with dynamic addition of words in mind. So in order to achieve this a number of new functions were written. These functions could be used for numerous other applications besides Tauira, and consits of:

- A function for creating a lexical entry based on spelling and word type.
- A function for saving a lexicon to a lexical file readable by the LKB and grammar writers.
- A function to add new words to a lexicon.

Additional re-usable extensions of the LKB system include a function to set the global variable for open word types (**unknown-word-types**) using a cut-off and a function to store well-formed tsdb test sentences (Oepen, 2001) in files of a specified size.

6.2.2 Evaluation criteria

Two evaluation criteria have emerged from the Tauira project for grammar writers to evaluate their work. First of all the benchmarks of Section 4.2.2 can be used to test the coverage of test suite sentences on the grammars word types. Secondly the ability of non-linguist users to correctly handle the dialogues created by Tauira (such as those in Chapter 5) can exemplify weak and strong points of a grammar like those discussed in Section 6.1.2.

6.2.3 Processing of unknown words

Tauira extends the theory of Erbach (1990), Barg and Walther (1998) and Fouvry (2003) in two ways. Firstly it simplifies the creation of a new lexical entry by explicitly formulating a set of simple hypotheses, which can be eliminated one by one, in stead of creating a disjunct feature structure which needs to be filtered.

Secondly, Tauira takes morphological information into acount in unknown word processing. Using morphological stems as opposed to using word spellings allows for a truly robust processing of unknown words not offered by any of the previous works.

6.2.4 Lexical acquisition in a dialogue context

The lexical acquisition tool created in the course of this project conforms to the project goals stated in Section 1.5. Because Tauira is a lexical acquisition tool in the context of a dialogue more then one unknown word per sentence can be resolved; this is something Erbach, Barg and Walther and Fouvry could not as easily work around.

Within its dialogue context Tauira provides, given the limitations of Section 6.1, a simple natural language dialogue through which a non-linguist can author new words. The questions asked in order to author the unknown words are generated automatically and therefore evolve together with the development of the grammar and test suite. Even large test changes to the grammar such changing the way a lexical entry is represented can be accounted for in Tauira (See Apppendix A for the grammar writer's guide).

The whole system also operates in and remains true to the bilingual structure of Te Kaitito: for each word its translation must also be resolved and the entire dialogue can be conducted by both Māori and Pakeha in their own language.

6.3 Improving usability

The current version of of Tauira could have some improvement for ease of use. The question strategies used require an evaluation (Section 6.3.1); errors made by the users could be taken into account (Section 6.3.2); statistics could reduce the number of hypotheses generated (Section 6.3.3); example sentences could be more carefully selected (Section 6.3.4); homonyms of equal word type are not considered (Section 6.3.4); the checks on sentence translation need improvement (Section 6.3.5) and the re-indexing process required for generation could be improved (Section 6.3.6).

6.3.1 User evaluation

The actual quality of the dialogues of Tauira in human-computer interaction has not been tested. To do this an independent user evaluation has to take place. This evaluation could consist of a randomised controlled trail in which users would be asked to enter sentences containing unknown words. The various groups could be (a) users using the Tauira system; (b) Users using Tauira without the example sentence stategies (Section 4.5.1 and 4.5.2); (c) users authoring words by selecting the right type name and (d) the grammar writer. Results (number of correct types, duration of the authoring process) could then be compared to find out how the different groups match up to the grammar writer.

6.3.2 User errors

The current setup pays little attention to possible errors made by users. Some simple improvements could be made to give the user a chance to correct these mistakes such as a spellchecker and undo/abort functionality. A spell checker could easily check whether an unknown word is closely related to a existing word in the lexicon and ask for feedback from the user as in the following dialogue.

```
> The dog braks
Did you mean 'The dog barks' ?
```

When the user wishes to reconsider its last answer calling *'undo'* could set the system back to the previous step. Analogously *'abort'* could abort the entire subdialogue as if the original sentence had never been entered.

6.3.3 Statistics

The current design is completely symbolic. Fouvry (2003) notes that by using his Part of Speech tagger that even when still considering highly unlikely odds (1 to 10.000) the PoS tagger is still able to make a significant reduction to the number of words types. It would be worth to see if a PoS tagger would be able to decrease the number of initial hypotheses. Of course one would have to find a tagged Māori corpus to train the PoS tagger on.

6.3.4 Selecting example sentences

At the moment the only criteria on which test-items are selected is the number of hypotheses they eliminate. To select the user friendliest test items a heuristic could be calculated to select the least complex sentence. This heuristic could be calculated by the preprocessor based on one or more of the following:

- The length of the sentence.
- The number of rule applications.
- The size of the feature structure.
- The degree of ambiguity of the sentence.

Also in itsdb one has the ability to define phonomenon categories such as noun phrase agreement and link a phenomenon category to each test suite sentence. In the TSNLP, MRS and CSLI test suites there is no category for word types. Defining

such a category and linking these to test suite sentences would make sure that sentences of this category are used preferentially in Tauira's questions.

Homonyms of equal word type

The first type of homonyms have equal part of speech like the Dutch *'ezel'* translating both to *'donkey'* as *'easel'*. Consider the following dialogue where Tauira is working with a Dutch-English grammar and the word *'easel'* is unknown but *'ezel'* and *'donkey'* are known.

```
> The easel was rather old
I do not understand the word 'EASEL'
...
What is the translation of 'The easel was rather old' in Dutch?
> De ezel was nogal oud
'The easel was rather old' paraphrases as:
de ezel was nogal oud
the easel was rather old
the donkey was rather old
```

In this previous example Tauira defines *'donkey'* and *'easel'* as synonyms in stead of homonyms. The dialogue should therefore come up with some type of question to resolve wether or not we are dealing with a hymonym or synonym of the existing word.

```
...
> De ezel was nogal oud
Does 'EASEL' mean the same as 'DONKEY' and is it not a
different sense of 'EZEL' ?
```

6.3.5 Translation

Currently relations between source and translation sentence are only checked between based on sets of predicates, not between the variables which appear as the arguments of predicates. This means that *'the dog chases the cat'* is considered identical to *'the cat chases the dog'*. MRS (Copestake, 2001) is however able to destinguish between these two sentences. To improve the translation checking process the MRS structures must therefore be compared more thoroughly.

6.3.6 Generation

Before newly added words can be actually used by the sentence generator a process called re-indexing is required. This re-indexing is (like most of the LKB system) not designed for dynamically adding and removing semantic relations. The current function removes all relations and indexes the relations from scratch. For speed purposes a reindexing procedure for a case where one or two new words are added should be implemented.

6.4 Expanding functionality

In this section phenomena that Tauira does not yet cope with are explored. These are irregular morphology (Section 6.4.1), homonyms (Section 6.4.2) and advanced forms of translation (Section 6.4.3).

6.4.1 Irregular morphology

Although the current setup deals with regular morphology, irregular morphology has not yet been dealt with. The sentence *'The sheep sleep'* will generate the following hypotheses: {*(n_plur_xmod_le, 'sheep'), (n_plur_ppcomp_le, 'sheep'), (n_plur_le, 'sheep')*} for words can only exist in plural such as *'media'* (e.g. one cannot say *'the media chases the cat'*). If one would take irregular morphology into account extra hyptheses such as *(n_intr_le, IRREG* {*(plur_form, 'sheep'* }*)* could be created. Indicating this could be an intransitive noun with the plural form *'sheep'* and an unknown singular form.

These hypotheses should probably end up at the back of the *hypotheses* list since irregular morphlogy is not very likely, although in irregular morphology for a word with a large occurence could be pretty likely, compared to regular morphology for words with a small occurence. The challenge is of course to develop a method to confirm or deny such hypotheses. One method could be asking questions such as:

```
Could you say 'the *blank* chases the cat' for an inflection
of SHEEP?
> Yes
How would you say 'the *blank* chases the cat' for SHEEP?
> the sheep chases the cat
```

Of course unknown words with irregular morphology could also occur in sentences where there is no inflection (e.g. *'a sheep eats'*). This would mean that for

every word type an irrigular hypotheses must be generated as well. The confirmation or refusal of these hypotheses could become even more complex.

6.4.2 Homonyms

Besides improvements made for dealing with homonyms of the same word type (Section 6.3.4) the system could also be extended to be able to cope with different types of homonyms. The current complications for two other types of homonyms are shown in this section.

Homonyms of different word type

The phrase *'This beach is both sandy and hot'* will not parse when lexical item *'Sandy'* is only known as a proper name and not as an adjective. For these homonyms of different word types there are no words marked as *unknown word* and Tauira's subdialogue will not be initiated.

Multiword homonyms

A third type of homonym Tauria currently does not deal with are homonyms which are part of multiword expressions. For example if one mentions the word *'sea lion'* which is unknown but *'sea'* is known and *'lion'* is not, it will be impossible to set the right word type for *'sea lion'* and link it to the Māori *'kakerangi'*.

6.4.3 Translation

Besides improving the current translation capabilities as described in Section 6.3.5, the translation part of the algorithm could also be extended to handle more complex translations, such as a case where multiple non-consecutive words make up the translation of a single word.

Even improved translation features would go beyond the current capabilities of the Māori-English grammar. As this grammar further develops it is however possile that more advanced translation features become availbable, such as allowing features from the source word in context to be taken into account in the translated word (Copestake and Sanfilippo, 1993).

6.5 Additional applications

The Tauira system could be applied to do tasks out of its original subdialogue context. Two possible applications are an authoring tool (Section 6.5.1) and the

assimilation of the Ngata English-Māori dictionary (Section 6.5.2).

6.5.1 Authoring mode

Currently the only method of authoring words with Tauira is by entering an unknown word within a sentence context. It would be possible to let words be authored in a so-called authoring mode. In stead of generating hypotheses from the sentence context hypotheses could be generated from specific commands. The command *'add noun penguin'* could loadup tauria using all possible noun hypotheses (*n_*_le*) for that word or *'add to dance'* would generate all possible verb hypotheses (*v_*_le*).

6.5.2 Ngata dictionary

The Ngata (1993) English-Māori dictionary distinguishes itself from other dictionaries by illustrating the use of Māori words within a sentence context rather that just an English translation. For example the definition for the word *'dog'* is illustrated with the phrase *'The child loved the dog.'* and its English translation *'I arohatia e te tamaiti te kurī'*. Recently the entire dictionary has become available on-line with the support of the ministry of education.

If one were able to select the sentence-pairs from the dictionary which the Māori-English grammar is able to translate and sort the sentence-pairs in such a manner that there is only one unknown word per sentence (i.e. the sentences with one unknown word first). One could run the sentences through Tauira and compile a list of question for a Māori and English speaker.

This Māori or English speaker could then structurally contribute to the lexicon by answering the list of questions. This will provide a large lexical increase for Te Kaitito.

Bibliography

Barg, P. and Walther, M. (1998). Processing unknown words in hpsg. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, Montreal, Quebec, Canada.

Copestake, A. (2001). An algebra for semantic construction in constraint-based grammars. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL 2001)*, Toulouse, France.

Copestake, A. (2002). *Implementing Typed Feature Structure Grammars*. Centre for the Study of Language and Information, Stanford, California, USA.

Copestake, A. and Flickinger, D. (2001). An open-source grammar development environment and broad-coverage english grammar using HPSG. In *Proceedings of the Second conference on Language Resources and Evaluation (LREC-2000)*, Athens, Greece.

Copestake, A. and Sanfilippo, A. (1993). Multilingual lexical representation. In *AAAI Spring Symposium on Building lexicons for machine translation*, Stanford, California, USA.

Erbach, G. (1990). Syntactic processing of unknown words. In P. Jorrand and V. Sgurev, editors, *Artificial Intelligence IV - methodology, systems,applications*, pages 371–382. North-Holland, Amsterdam, the Netherlands.

Fouvry, F. (2003). Lexicon acquisition with a large-coverage unification-based grammar. In *10th Conference of the European Chapter of the Association for Computational Linguistics, Research notes and demos*, Budapest, Hungary.

Harding, P., Bain, C., and Bedford, N. (2002). *Lonely Planet New Zealand. 11th edition*. Lonely Planet publications Pty ltd, Victoria, Australia.

Kamp, H. and Reyle, U. (1993). *From discourse to logic*. Kluwer Acedemic Publishers, Dordrecht, the Netherlands.

Knight, K. (1996). Learning word meanings by instruction. In *Proceedings of the 13th National Conference on Artificial Intelligence*, Portland, Oregan, USA.

Knott, A., Bayard, I., de Jager, S., and Wright, N. (2002). An architecture for bilingual and bidirectional nlp. In *Proceedings of the 2nd Australasian Natural Language Processing Workshop*, Canberra, Australia.

Knott, A., Moorfield, J., Meaney, T., and Ng, L.-L. (2003). A human-computer dialogue system for māori language learning. In *Proceedings of the World Conference on Educational Multimedia, Hypermedia and Telecommunications (ED-MEDIA)*, Honolulu, Hawaii, USA.

Ngata, H. (1993). *Ngata English-Māori Dictionary/Pukapuka Taki Kupu a Ngata*. Learning media ltd, Wellington, New Zealand.

Oepen, S. (2001). [incr tsdb()] — competence and performance laboratory. User manual. Technical report, Computational Linguistics, Saarland University, Saarbrücken, Germany.

Oepen, S., Netter, K., and Klein, J. (1997). TSNLP — Test Suites for Natural Language Processing. In J. Nerbonne, editor, *Linguistic Databases*, pages 13 – 36. CSLI Publications, Stanford, Californa, USA.

Villa, D. (2002). Integrating technology into minority language preservation and teaching efforts: An inside job. *Language Learning & Technology*, **6**(2), 92–101.

Appendix A

Grammar writer's guide

A.1 Introduction

This guide describes how the system known as Tauira can be used on a grammar written for the LKB system. As explained in the Tauira report Tauira can create benchmarks on which the grammar can be evaluated and provides a lexical authoring tool within the context of a dialogue.

This guide consits of three parts. Section A.2 describes how the script of the grammar should be extended, Section A.3 describes how to operate the preprocessor and Section A.4 describes how to use Tauira as part of a dialogue system.

A.2 Extending grammar script

In order to run Tauira's preprocessor and authoring tool some grammar specific variables need to be set. We recommend you set these variables in a separate file called `unknowns.lisp` in your script directory and adding the following line to your grammar:

```
(lkb-load-lisp (this-directory) "unknowns.lisp")
```

An example of this file is (for a Māori-English grammar) is shown in Figure A.1. The functions used and variables set are all located in the *lkb*-package. Therefore add the following as the first line of your file:

```
(in-package :lkb)
```

You need to specify which types of your grammar are open class. Tauira uses the LKB system's standard global variable for this: *lkb::*unknown-word-types**.

```
(in-package :lkb)

;; --- *unknown-word-types* ---

(set-unknown-word-types 10)

;; --- make-unknown-word-sense-unifications ---

(defun make-unknown-word-sense-unifications (word-string)
 (when word-string
  (let ((orth (split-into-words word-string)))
    (append
     (make-orth-fs orth)
     (list
       (make-unification
        :lhs (create-path-from-feature-list
              '(SEM HOOK KEYPRED LIST FIRST PRED))
        :rhs (make-u-value :type
              (format nil "~{~A~^_~}_rel" orth)))
       (make-unification
        :lhs (create-path-from-feature-list
              '(SEM HOOK KEYPRED LAST))
        :rhs (create-path-from-feature-list
              '(SEM HOOK KEYPRED LIST REST)))))))))

(defun make-orth-fs (orth)
 (let (rests-path orth-fs)
  (loop for word in orth do
   (pushnew (make-unification
             :lhs (create-path-from-feature-list
                   (append '(ORTH LIST) rests-path '(FIRST)))
             :rhs (make-u-value :type word))
    list)
   (push 'REST rests-path))
   orth-fs))

;; --- translation variables ---

(setf *language-feature* 'LANG)
(setf *language-feature-values* '(english maori))
(setf *predicate-path* '(SEM HOOK KEYPRED LIST FIRST PRED))

(setf *index-for-generator-p* T)
```

Figure A.1: Example `unknowns.lisp` file

This variable must be set to a list of word types. Tauira includes the function *lkb::set-unknown-word-types* with which *lkb::*unknown-word-types** will be set to all words types used in the lexicon at least *n* times. Add either one of the following lines to your file to specify the open class word types:

```
(setf *unknown-word-types* 'list of open class word types)
(set-unknown-word-types minimal occurences)
```

In order for Tauira to create new lexical entries the form lexical entries within your grammar must be described. Within the LKB system a feature structure is represented as a *list* of *lkb::unification* structures. You must include a function called *lkb::make-unknown-word-sense-unifications* which creates a feature structure in which the required features for a lexical entry (ussally just the semantics and orthography) are set based on a word string. If your lexical entries are described in the lexicon as:

```
word := word-type &
[
...
feature-path feature-value
...
].
```

Your *lkb::make-unknown-word-sense-unifications* will look something like:

```
(defun make-unknown-word-sense-unifications (word-string)
(list
...
(make-unification :lhs (create-path-from-feature-list
'(feature-path) :rhs feature-value)
...
))
```

Please note that setting the previous global variable and function will enable the LKB system's unknown word mechanism during normal parsing. This mechanism of the LKB system's sentence parser will generate entries of *lkb::*unknown-word-types** using *lkb::make-unknown-word-sense-unifications*.

If you want to use the bilingual capabilities of Tauira the variable *lkb::*language-feature** must be set. If set *lkb::*language-feature-values** must be set to a possbile languages for this grammar and *lkb::*predicate-path** to a list representing the path to the first predicate.

```
(setf *language-feature* language-feature)
(setf *language-feature-values* 'list of languages)
(setf *predicate-path* 'list of path to first predicate)
```

Finally the variable *lkb::*index-for-generator-p** will define whether or not re-indexing is for the LKB system's generator is required for this grammar. By default it is set to *NIL* (Lisp's false value). If the new words added are also required to be generated on the fly this variable must be set to *T* (representing true).

```
(setf *index-for-generator-p* T or NIL)
```

A.3 Preprocessing test suites

Tauira uses test suites to come up with questions for the user. You can describe your test suite as a file with one sentence per line or you can extract test suites from the itsdb database (Oepen, 2001). To extract sentences from the itsdb start up the itsdb front-end and select the test suite you want to extract sentences from. Now load the lisp file `tauira/tsdb-util.lisp` and run the function *store-selected-tsdb* applied to a filename. You can enter an optional argument to specify the maximum number of sentences per file. The second and further files with have a number added to the filename.

```
(in-package :tsdb)
(store-selected-tsdb filename optional-sentences-max)
```

To run the preprocessor run the LKB, the Tauira files and your grammar with the modified script file from your `LKB/src` directory. (Make sure your Tauira files are in the `LKB/src/tauira` directory).

```
(load "general/loadup")
(load-system "mrs")
(load "tauira/loadup")
(read-script-file-aux "script file location")
```

To start the preprocessor switch to the *lkb* package and run the function *lkb::precache3* with an input and output file.

```
(in-package :lkb)
(precache3 input-file output-file)
```

Note that this process might be quite slow (Tested on a pentium for a 100 sentences can take up to an hour to preprocess).

Section 4.2.2 of the Tauira report describes various benchmarks for test suites. To create a file of these benchmarks for your preprocessed test suite files use *append-tests-from-file* and *write-tests-results*. You can use *append-tests-from-file* multiple times to add more files.

```
(in-package :lkb)
(append-tests-from-file input-file)
(write-tests-results output-file)
```

A.4 Running Tauira

To run Tauira first loadup the LKB, the Tauira files and your grammar as in the previous section. Then load your preprocessed test suites sentences using the *lkb::append-tests-from-file* function. For effeciency run the function *lkb::set-word-occ* afterwards. The function *(lkb::wrapper)* (no arguments) provides an environment in which prompted sentences are parsed and paraphrased (leave the prompt with *'end'*). If an unknown word is encountered Tauira is invoked.

During your session with Tauira you can change the language of Tauira's dialoges. *lkb::*lang** defines the current language for Tauira's dialogue. *lkb::*lang** can be set at any time to any value, but there are currently only dialogues available for English (*english*) and te reo Māori (*maori*). Extend the file `dialogue.lisp` with sentences for your language if desired.

The lexical files created during a Tauira session are stored in the global variable *lkb::*user-lexicon**. You can store this lexicon to a file using the function *lkb::store-lex-database*. You can reload this using the function *lkb::load-lex-database*. Your session could proceed as follows:

```
(load "general/loadup")
(load-system "mrs")
(load "tauira/loadup")
(read-script-file-aux "script file location")
(in-package :lkb)
(append-tests-from-file input-file) (repeat for every file with test-items)
(set-word-type-occ)(optional, for efficiency)
(load-lex-database *user-lexicon* user-lexicon-filename) (optional)
(setf *lang* 'language)
(wrapper)
(store-lex-database *user-lexicon* user-lexicon-filename) (optional)
```

There are some aditional switches with which you can determine the behaviour of Tauira. Unless mentioned otherwise the default setting of these switches is *T* (true). You can set these switches as follows:

`(setf` *switch T or NIL*`)`

When *lkb::*tauira-use-morph-p** is set Tauira will create different morphological stems for an unknown word (e.g. *'walk'*, *'walke'* and *'walked'* for *'walked'*), otherwise only the original unknown word's spelling is used as a stem. When *lkb::*tauira-use-sentence-p** is set the number of hypotheses is reduced using information from the sentence context.

When *lkb::*tauira-use-multiple-example-p** is set multiple choice example sentences questions may be asked to the user. Analogues when *lkb::*tauira-use-single-example-p** is set single choice example sentence questions may be asked to the user. When *lkb::*tauira-use-word-example-p** is set the user will have to select the word type for the unknown word out of example words for the remaining hypotheses' word types (e.g. *'dog'*, *'credit card'* and *'apology'*), otherwise the user will have to select the word type based on the word type's name (e.g. *'n_intr_le'*).

*lkb:: *tauira-show-hypotheses-p** is the only switch that is *NIL* by default. If set it will print the remaining hypotheses on the screen before a question is given to the user.

Appendix B

User's guide

B.1 Introduction

The following Section (B.2) serves as a template of what should be included in a user's guide of a dialogue system using Tauira. Before the user is able to use Tauira he or she must be familiar with the limitations of the grammar. Thefore a user's guide for a system using the ERG grammar should for example include the fact that the system does not destinguish between male and female proper names. This shall be known as the **grammar disclaimer**.

B.2 Unknown words

The system you are about to use has only a limited vocabulary. Like humanbeings the system can learn new words that are not yet in its vocabulary. When you mention a new word to the system the system will require some assistance from you in order to learn more about the word. These questions will be either multiple choice, which you should answer with a number or single choice, in which case you should reply *'yes'* or *'no'*.

The system might ask you whether or not certain sentences are correct. These sentences might not make any sense but the you should reply whether or not these sentences are grammatically correct. Pay special attention to the spelling of the word for single choice questions: if a word is not spelled correctly refuse it. The system however cannot handle words that have irregular inflection.

insert your grammar disclaimer here.