A preprocessor for an English-to-Sign Language Machine Translation system

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE

OF THE UNIVERSITY OF STELLENBOSCH

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Andries J. Combrink

December, 2005

Supervised by: Dr. L. van Zijl

Declaration

I, the undersigned, hereby declare that the work contained in this thesis is my own original work and has not previously in its entirety or in part been submitted at any university for a degree.

Signature: ..... Date: .....

Abstract

Sign Languages such as South African Sign Language, are proper natural languages; they have their own vocabularies, and they make use of their own grammar rules.

However, machine translation from a spoken to a signed language creates interesting challenges. These problems are caused as a result of the differences in character between spoken and signed languages. Sign Languages are classified as visual-spatial languages: a signer makes use of the space around him, and gives visual clues from body language, facial expressions and sign movements to help him communicate. It is the absence of these elements in the written form of a spoken language that causes the contextual ambiguities during machine translation.

The work described in this thesis is aimed at resolving the ambiguities caused by a translation from written English to South African Sign Language. We designed and implemented a preprocessor that uses areas of linguistics such as anaphora resolution and a data structure called a scene graph to help with the spatial aspect of the translation. The preprocessor also makes use of semantic and syntactic analysis, together with the help of a semantic relational database, to find emotional context from text. This analysis is then used to suggest body language, facial expressions and sign movement attributes, helping us to address the visual aspect of the translation.

The results show that the system is flexible enough to be used with different types of text, and will overall improve the quality of a machine translation from English into a Sign Language.

Opsomming

Gebaretale, soos Suid-Afrikaanse Gebaretaal, is in eie reg natuurlike tale; hulle maak gebruik van hul eie woordeskat en gebruik elkeen hul eie taalreëls.

Des nie teen staande skep masjienvertaling vanaf 'n gesproke taal na 'n gebaretaal interessante uitdagings. Hierdie probleme word veroorsaak omdat die karakter van gesproke- en gebaretale verskil. Gebaretale word geklassifiseer as visueel-ruimtelike tale: 'n persoon wat gebaretaal gebruik, sal gebruik maak van die ruimte rondom hom en daarmee saam van liggaamshouding, gesigsuitdrukkings en handbewegings, om hom te help kommunikeer. Dit is die afwesigheid van hierdie elemente in die geskrewe vorm van gesproke tale, wat kontekstuele onduidelikheid tydens masjienvertaling veroorsaak.

Die werk wat in hierdie tesis aangebied word, is gemik daarop om die onduidelikhede uit te skakel wat tydens die masjienvertaling van geskrewe Engels na Suid-Afrikaanse Gebaretaal veroorsaak word. 'n Voorverwerker is ontwerp en geïmplementeer wat gebruik maak van areas in linguistiek, soos anafora-oplossing en 'n data struktuur, wat 'n toneelgrafiek genoem word, om te help met die ruimtelike gedeelte van die vertaling. Die voorverwerker maak ook gebruik van semantiese en sintaktiese ontleding en 'n semantiese verwantskapsdatabasis, om emosionele konteks vir geskrewe teks te vind. Hierdie analise help dan om die attribute van liggaamshouding, gesigsuitdrukkings en gebarebewegings voor te stel, wat help om die visuele gedeelte van die vertaling te bemeester.

Die resultate toon dat die stelsel buigsaam genoeg is om saam met verskillende tipes teks gebruik te word en dit sal die kwaliteit van 'n masjienvertaling van Engels na 'n Gebaretaal globaal verbeter.

Acknowledgements

There are many people that need to be thanked for their contributions to this work; in particular Dr. Lynette van Zijl, my supervisor, for inspiring me to always try harder and to do better, as well as for her patience and mentorship throughout this postgraduate study; for my parents and the rest of my family who also gave me their unconditional support. Lastly I want to thank Lynn for always believing in me, even when I found it hard to do myself.

The financial assistance of the National Research Foundation (NRF) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the author and are not those of the NRF.

Contents

List of Tables

List of Figures

Introduction

0.1 Motivation

Sign Languages such as *South African Sign Language* (SASL) are proper natural languages. SASL has its own vocabulary and grammar rules, the same as any other natural language [17].

Researchers at the University of Stellenbosch are working on tools to aid hearing people who are learning SASL. Their combined efforts fall under the *South African Sign Language Machine Translation* (SASL-MT) project [1, 40]. The SASL-MT project will provide tools to translate written English into SASL, and display the results as computer animation.

One of these tools is an application where an avatar will sign the SASL equivalent of an English document. This thesis describes the design and implementation of a preprocessor for such an application. The preprocessor performs two functions to improve the quality of an English-to-Sign Language translation. Firstly, differences between spoken and signed languages, which will be discussed in further detail, cause contextual ambiguity during translation. The preprocessor is responsible for resolving these ambiguities. Secondly, the preprocessor tries to find emotional context from text so that the translation will not only contain information about what is said, but also about how it was said.

0.2 Constraints

In this section the nature of Sign Languages is briefly looked at, and then the constraints and goals set for this work will be summarized.

Sign Languages such as SASL are *visual-spatial* languages [17]. The space in front of a person using Sign Language is important, and from here onwards will be referred to it as the *signing space*. The signing space has three functions: Words are signed in this space, it forms part of the way in which Sign Languages manage objects and their relationships to one another, and it is also used for the unique way in which pronouns are used. As an example, during a discourse the person *John* is mentioned for the first time. The signer will sign John's name somewhere in the signing space. Thereafter, every time the signer wants to make a reference to *John*, he will indicate the location where he signed John's name, and the watcher will know who is being talked about.

In the same way that spoken languages use sound, Sign Languages use visual clues as communication medium. These visual clues are observed as signs, facial expressions and body language. With spoken languages, *how* something is communicated is as important as *what* is being communicated. For Sign Languages this is even more so, and therefore it is said that Sign Languages are expressive languages.

The purpose of the preprocessor is to improve the quality of an English-to-Sign Language translation. A summary is given firstly, and it is then discussed how the preprocessor will make these improvements and why it is necessary to do so. The tasks of the preprocessor are:

- To find grammatical structure and part-of-speech information from text.
- To manage the spatial component of Sign Languages which will entail:
  - Resolving pronouns by finding their antecedents.
  - Managing the usage of the signing space.

- To generate prosodic information for the text which includes:
  - Body language.
  - Facial expressions.
  - Sign movement.
- To store the results in a universal and flexible format.

The first task of the preprocessor is to find grammatical structure and part-of-speech information from text. Many natural language processing algorithms use grammatical structure and part-of-speech information to help with *syntax*[1] and *semantic*[2] analysis.

As mentioned, Sign Languages are spatial languages. With regards to the spatial component, the preprocessor addresses two problems that arise during translation from English to SASL. The first problem is in the area of *anaphora*[3] [26] where pronoun resolution is performed to identify *antecedents*[4] [26] for pronouns. In other words, if a pronoun *he* is found in some text, the problem will be to find whether the *he* is referring to say, John or Peter. Referencing a person in Sign Language requires pointing to the location where that person was placed. Therefore, to point to the correct location, the antecedent needs to be known for the pronoun found in text. The second problem is keeping track of which objects occupy what part of space at any specific time during the discourse. In other words, it is important to manage the usage of the signing space.

Because of the expressive nature of Sign Languages, the generation of *prosodic*[5] [29] information is crucial to create a realistic translation. This means that for Sign Languages, prosodic information needs to be created that will describe body language, facial expression and the nature of the sign movement.

[1] Syntax refers to the grammatical structure of a sentence.
[2] Semantics refers to the meaning of a sentence.
[3] Anaphora describes the process of one object pointing back to another object. The pointing back is done to make a reference, or imply the first object.
[4] Antecedents are the object that anaphors, for example pronouns, refer to.
[5] Prosodic information refers to the non-lexical information contained within an utterance to convey meaning for example facial expressions.

Finally, the results generated by the preprocessor are stored in a flexible and universal format to simplify re-use.

In the next section an overview of the rest of the thesis will be given.

0.3 Overview of the rest of the thesis

The rest of this thesis consists of six chapters. The first four chapters describe the design of the preprocessor, while the final two chapters will discuss the analysis of the implementation, the conclusions from the work that was done, and suggested future work.

Chapter 1 describes the method that was used to find grammatical structure and part-of-speech information from text. Chapter 2 describes the use of a semantic relational database as central knowledge repository, and which plays an important role in the implementation of the preprocessor. In chapter 3 the constraints set on the spatial component of Sign Languages will be given, and a description of the algorithms to achieve these goals. Chapter 4 looks at the spatial constraints, or in other words, addresses the problem of creating prosodic information that will be used with the machine translation. An overview of how the output will be stored in a flexible and universal format can be found in Appendix G and H.

An analysis of an implementation of the preprocessor is given in chapter 6, and finally conclusions and possible future work in is discussed in chapter 7.

Chapter 1

Grammar and part-of-speech information

This chapter provides an overview of *Tree-adjoining grammars* (TAGs) [23]. TAGs are used in the SASL-MT project for text analysis and processing, the purpose of which is to find grammatical structure and part-of-speech information.

There are different approaches to perform the task of finding grammatical structure and part-of-speech information. However, TAGs were prescribed by the SASL-MT project and therefore other approaches are not considered in this work. We will discuss in this chapter however why TAGs are useful for natural language applications and why they are better that some other approaches. The interested reader may read a survey and critique of American Sign Language, natural language generation and machine translation systems by Matthew Huenerfauth [22]. In his technical report, he investigates four English-to-Sign Language machine translation systems, each system using a different approach to represent grammatical structure and part-of-speech information.

1.1 Introduction

One goal of the SASL-MT project is to provide a tool that will translate English into its SASL equivalent. Because SASL is a natural language with its own vocabulary and grammar rules, direct translation cannot simply be performed by saying the next English word found is the *gloss*[1] for the SASL sign. Such an argument is similar to changing every word in an English document to a French equivalent, and then claiming it to be a French document. The grammatical structure of a sentence, and part-of-speech information for each word is required to perform *machine translation* (MT).

Grammatical structure and part-of-speech information is equally important and necessary for other natural language processing algorithms. As an example, assume it is required to perform some form of semantic analysis on a sentence. Suppose then that the word *plant* is found in the sentence. Depending on the use of the word as a verb or a noun, the meaning of the word, and also that of the sentence will change.

The natural language processing algorithms implemented by the rest of the preprocessor all make use of grammatical structure and part-of-speech information to perform either syntactic or semantic analysis.

The rest of this chapter gives a brief overview of TAGs and also motivate why TAGs are useful to represent the nature of natural language sentences. TAGs and Sign Language is then discussed by considering another project where TAGs were used in an English-to-Sign Language MT system. The role TAGs played during that project is investigated, some of its shortcomings are considered, and finally it is argued how the preprocessor addresses these shortcomings.

[1] A Sign Language gloss is the English name given to a sign, and written in capital letters.

1.2 Relevance of Tree-adjoining grammars

In this section it is briefly stated what TAGs are, and why they are useful to represent the nature of natural language sentences. For a formal overview and definition of TAGs see Appendix A.

Unlike many other grammars, for example *context free grammars* (CFG), which are string generating systems, a TAG is a tree generating system [23]. In other words, TAGs are different from other grammars because they use tree objects instead of strings to represent grammatical structure. It is the properties of these tree objects that make it advantageous to use TAGs with natural language to describe the structure of a sentence.

TAGs have two main properties. The first one is called *extended domain of locality* (EDL), and the other is called *factoring recursion from the domain of dependencies* (FRD). All other properties of TAGs are derived from these two basic properties [23]. To see a formal explanation of EDL and FRD the reader is again referred to Appendix A.

In short, looking at the rules of a grammar as its domain of locality, the TAG property of EDL implies that a TAG can represent the same language as some CFG, but doing so with less, more complex grammar rules. The TAG property of FRD implies that after a sentence was derived from the grammar, it is possible to find the grammar rules from which the sentence was derived. This is not always possible in a CFG for example, and the advantage of such a property is the fact that context is preserved. To conclude, the TAG properties of EDL and FRD make it possible to include complex grammar rules while keeping the size of the grammar manageable, and also provide a way to preserve context during derivation. Both of these properties are necessary to represent natural language.

1.3 Tree-adjoining grammars and Sign Language

TAGs have been used in English-to-Sign Language MT systems before. At the University of Pennsylvania [45] a project called the *Translation from English to ASL by Machine* (TEAM) project was created. During this project, a system was developed to translate English into *American Sign Language* (ASL) using a *Synchronous Tree-adjoining grammar* [2] (STAG) [32]. In machine translation, a sentence in a *source language* is translated to a sentence in a *target language*. An STAG is a TAG with a corresponding grammar rule in the target language for each grammar rule from the source language. For each sentence that is parsed in the source language, a sentence in the target language is then created. For more information about English-to-Sign Language MT, about the TEAM project and STAG see Appendix E.

TEAM is one of the four systems that Huenerfauth investigated in his technical report of ASL, natural language generation and MT systems [22]. Following are some of the conclusions from his report.

Sign Language makes use of two types of signs: *manual signs* and *non manual signs* [11]. Manual signs are the signs created by hand, wrist and arm movements, while an example of a non manual sign would be the raising of eyebrows during a question.

The TEAM system took manual and non manual signs into account during translation by incorporating non manual sign generation into the STAG. Huenerfauth argued that although the inclusion of non manual signs improved the quality of the translation, the way the non manual signs were created was an over simplification of the real world. Huenerfauth stated that the idea that non manual signs are determined by the grammatical structure of a sentence was based on earlier research of ASL that has now been discarded by modern findings. It is currently known that the semantics of a sentence also plays a part in

[2] Synchronous Tree-adjoining grammar is a variant of TAG where two TAGs are used. The two TAGs are synchronous in the sense that adjunction and substitution operations are applied simultaneously to related nodes in pairs of trees, one for each language.

the generation of non manual signs.

Huenerfauth suggested that the TEAM system would improve if non manual signs were determined by semantic analysis. He further stated that the use of semantic analysis would also help to resolve some of the contextual ambiguity created during translation of English into Sign Language.

The preprocessor described in this work addresses both suggestions made by Huenerfauth. The preprocessor will resolve some of the contextual ambiguity that is created by translating English into Sign Language, and it will also use semantic analysis to determine the emotional state of a signer, improving the quality of non manual signs that will be created with the translation.

1.4 Summary

In this chapter it was stated that grammatical structure and part-of-speech information is important for natural language processing. It was mentioned that different approaches exist to find grammatical structure and part-of-speech information, but that TAGs were prescribed by the SASL-MT project.

It was briefly stated what TAGs are, and some of their useful properties that make them well suited for natural language applications were mentioned. Finally another English-to-Sign Language MT project that uses TAGs was investigated, the shortcomings of that project were highlighted, and it was stated that the preprocessor described in this work addresses those shortcomings.

In the next chapter a definition of a semantic relational database will be given. It is also motivated why such a database was used as a central knowledge repository in the preprocessor.

Chapter 2

Using a central knowledge repository

In this chapter the use of a semantic relational database as central knowledge repository is described. Together with grammatical structure and part-of-speech information, the knowledge repository plays an important part in the success of the natural language processing algorithms implemented by the preprocessor.

2.1 Introduction

A semantic relational database is a database that contains semantic and syntactical information, structured in a relational manner [2].

As an example, it is known that some words can have more than one meaning depending on their part-of-speech category, or the context in which the words are used. It is also known that some words have the same meaning and are called synonyms. Assume now that a relational database is created with two tables, the one to store meanings of words, and the other to store the words themselves. If one meaning is linked for each context that every word could be found in, then a simplistic semantic relational database has been created. The two tables can now be used to find synonyms by finding all the words that link to a specific meaning. The different ways in which one word could be used can now also be found.

This small example could be extended to show other semantic or syntactical relationships such as antonyms.

In the next section a well known semantic relational database will be looked at. Next some of the shortcomings of this database are considered in context with the preprocessor, and it is shown how the database can be modified to better suit the required needs.

2.2 The WordNet semantic relational database

WordNet is a semantic relational database that was developed at the Cognitive Science Laboratory at Princeton University [2]. The creator is Professor George Miller who based his design on theories of how people organize and store lexical items. See [3] for an online bibliography of papers and articles published on this subject. The reason why WordNet is such a good choice for a natural language application is because of the number of items and relationships that are contained within the database.

WordNet contains words from four part-of-speech categories: nouns, verbs, adjectives and adverbs. WordNet also contains descriptions of concepts or ideas or meanings of words. WordNet finally links words with meanings and meanings with other meanings through different types of relationships. To see a detailed description of the types of relationships included in WordNet see Appendix D.

The latest version of WordNet available is WordNet 2.0. A C library is included with the program for people who want to use it with their own applications.

2.3 Shortcomings and improvements

The WordNet database structure can be seen by means of an entity relationship diagram in Figure 1 on page 14. The diagram shows each table in the database, the columns of each table and the relationships between tables. Two of the

important tables in the database are called *wn_gloss* and *wn_synset*. The table *wn_gloss* contains the meanings of words and concepts, and the table *wn_synset* contains words themselves. The two tables are linked by a one-to-many relationship where each entry from *wn_synset* is linked to one meaning from *wn_gloss*, while an entry from *wn_gloss* could be referred to by many entries from *wn_synset*. The other tables are used to show different types of relationships between words and meanings, or between meanings and other meanings. In the light of an implementation this structure has some shortcomings. This includes the need to populate the database with additional information.

The first shortcoming of the WordNet database is the fact that it contains words from only four part-of-speech categories. Information is not only needed about nouns, verbs, adverbs and adjectives, but also about conjunctions, pronouns and proper nouns. In the original database there exists a column in the *wn_synset* table called *ss_type* which stores a letter to represent the part-of-speech category. In the new database a *Part_of_Speech* table was created to store a name and description of each part-of-speech category that is consider. The *sense*[1] of each word was then linked to a part-of-speech category.

The original database is not flexible enough to create new relationship types without changing the database structure and this created a second shortcoming. All the relationships of a specific type are stored in their own database tables. The same is true if words or meanings should be grouped together in a new manner. To see why this is a problem, consider the conjunction part-of-speech category. A conjunction can be one of three types, for example *and* which is a *simple co-ordinate conjunction*, *between* which is a *correlative co-ordinate conjunction*, or *after* which is a *subordinate conjunction*. If it is necessary to add this information to the original database structure, it would be required to add a new table called *wn_conjunction_frame*, where a frame encloses the grouping. To overcome this problem, the database structure was made more flexible by creating the *Relationship_Type*, and *Frame_Type* tables so that a new frame type, or relationship type can be added by populating the database with new content,

[1]The sense of a word is one sense in which that word could be used.

and not by changing the database structure.

The major shortcoming of the original database structure is that even though the meaning of a word is stored, together with relationships to other words, it is either difficult, or impossible to find world knowledge, or attributes of a certain word. Consider the word *brother*. The gender of this word is male and it can be found by examining the meaning of the word which is *a male with the same parents as someone else*. The meaning of the word *cousin* is stored as *the child of your aunt or uncle*. To determine that the gender of the word cousin could be either male or female now becomes a difficult linguistic problem. Therefore, the biggest addition to the original database structure was to create tables that would store metadata about a sense of a word. An example of metadata that was added to the database was to store the gender of a noun as *male*, *female*, *male or female*, or *male and\or female*. Languages such as German contain nouns that could have both the gender of male and female, and the option of *male and or female* is therefore added for future use.

The new database structure can be seen by means of an entity relationship diagram in Figure 2 on page 15.

2.4 Summary

In this chapter a semantic relational database was described as semantic and syntactic information stored in a relational manner. A description of the WordNet semantic relational database was given, and how it was shown how the database was modified to be used in the preprocessor as central knowledge repository. In the following chapters the role that the semantic relational database played in the different areas of the preprocessor will be described in further detail. The next chapter describes how the preprocessor addresses the spatial component of Sign Languages and its management.

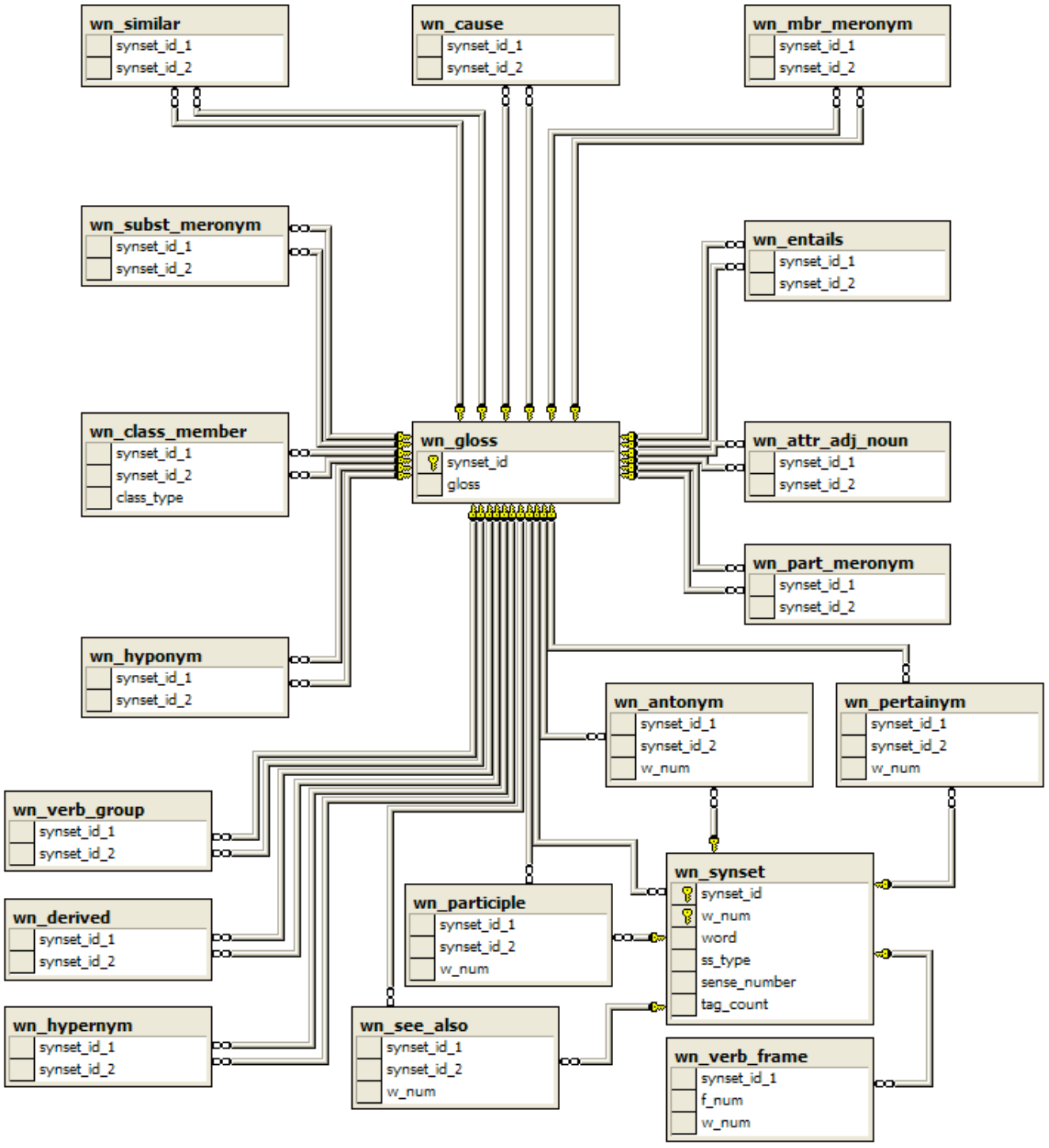

Figure 1: Entity relationship diagram of WordNet semantic relational database.

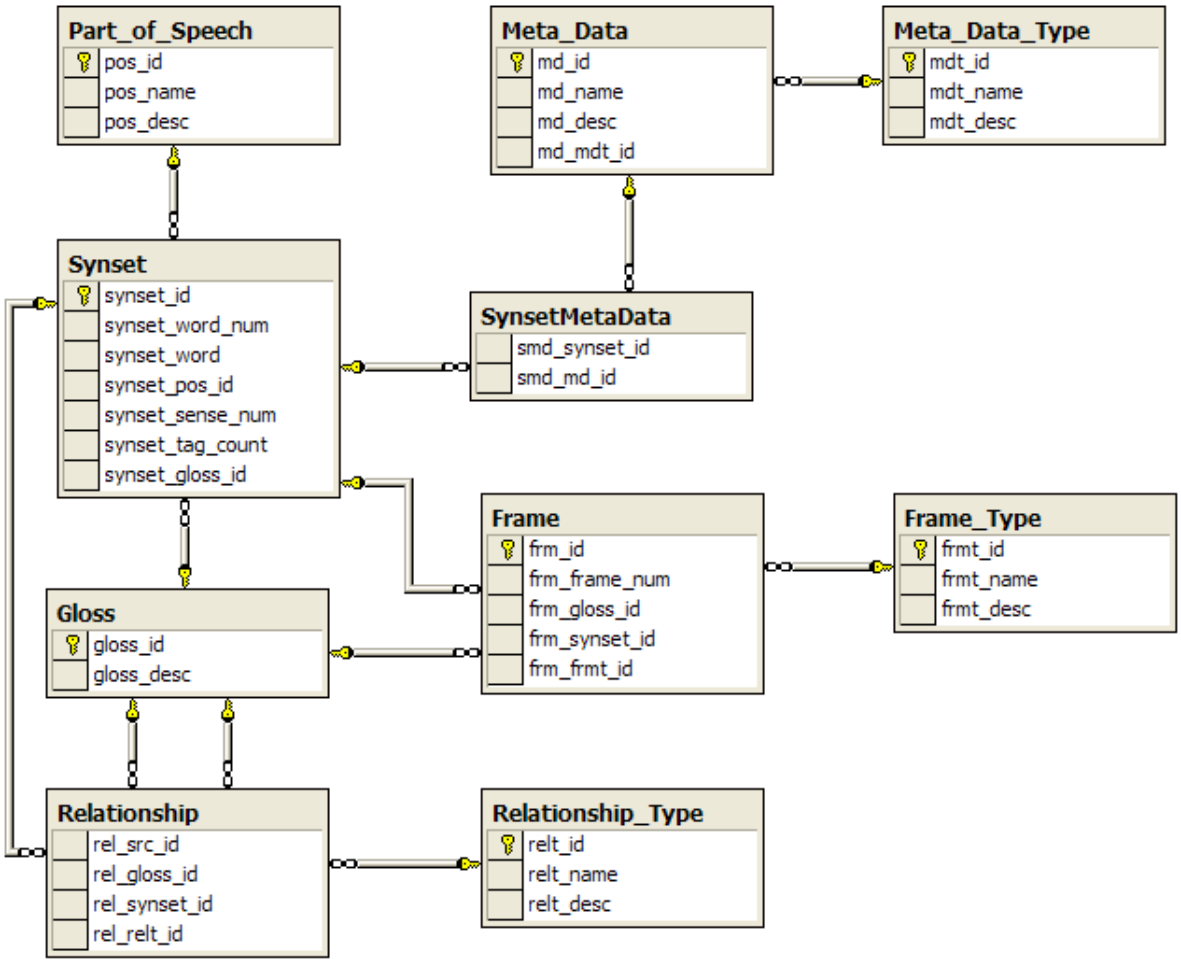

Figure 2: Entity relationship diagram of new semantic relational database.

Chapter 3

The spatial component of Sign Languages

Sign Languages are visual-spatial languages. This chapter focuses on the spatial component of Sign Language. The different functions of the signing space and the linguistic challenges it creates during English-to-Sign Language machine translation are investigated. Finally a description of an implementation of managing the unique way in which pronouns are used for Sign Languages is given for the preprocessor.

3.1 Introduction

The signing space is important to a person using Sign Language. The signing space has three functions: Words are signed in this space, it forms part of the way in which Sign Languages manage objects and their relationships to one another, and it forms part of the unique way in which pronouns are used (see Figure 3 on page 17).

One component of the signing space, as defined in [17], is called the *sight line*. The sight line is described as an imaginary line that extends outwards from the middle of the signer's chest, parallel to the floor. This line divides the space in front of the signer into a left and a right side.

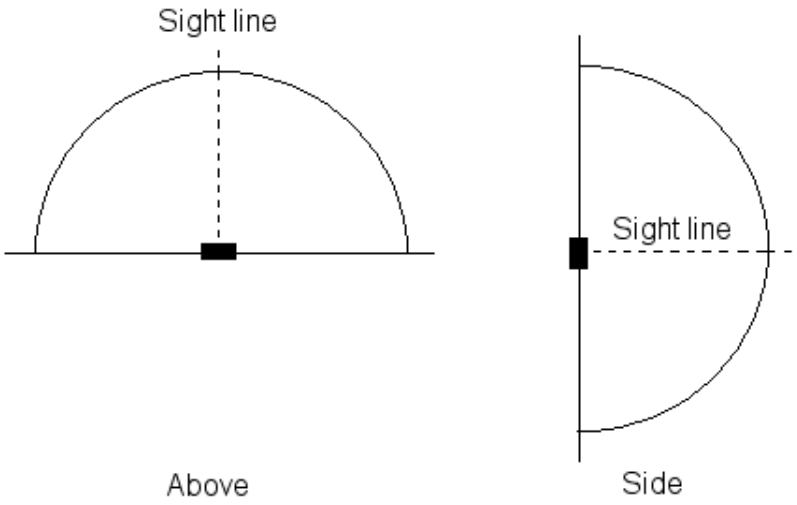

Figure 3: Signing space

The sight line itself is reserved for reference to the observer. Assume a signer is in conversation with an observer and he wants to ask the observer a question directly, "When did you go?" To show the observer that the *PAST WHEN GO* was directed at him, the signer will point on the sight line towards the observer creating the sentence *PAST WHEN GO YOU*. The space close to the signer is used for signing, and the space on left and right side of the sight line is also used for placing objects such as people. These objects may then be referenced at a later stage.

To illustrate this, imagine a conversation between a signer and an observer. During the conversation, the signer explains to the observer how their friend John, got lost in the city. When John is mentioned for the first time, the signer signs John's name-sign[1] somewhere in the signing space. John was thus *placed* in that space, and will occupy it until he is re-placed or replaced by another person. As the conversation continues and the signer wants to refer to John, he will point to the position John is occupying. The observer will then know to whom the signer is referring.

[1]A name-sign is the sign given to a person and is used to refer to that person instead of finger spelling the person's name.

Relationships between objects can also be shown through the use of the spatial component. Assume that during the conversation about John getting lost, the signer wants to show that John eventually parked his car next to a building. It is important that the signer *show* the *car* next to the *building*. If he signs the *car* above the position where he signed the *building*, the observer might conclude that the car ended up on top of the building.

To see another example of these subtleties involved when using the signing space, imagine a conversation between a signer and an observer. The signer explains to the observer that he will be leaving soon. For the observer to understand correctly, the signer must sign the verb in the direction of the door, otherwise the observer might wonder why the signer wants to leave the room through the wall.

Role playing is another component of communication which is done by the help of the signing space. To explain this, consider the following sentence, *John said: 'I cannot find the way home!'*. When this sentence is spoken, the word *I* refers to *John*, and not the speaker. In this instance the speaker plays the role of John. In Sign Language, if a signer wants to sign the sentence given above, he changes his body position to where John was placed in the signing space. He therefore plays the role of John, and signs what John said, as if he is John. In this way the observer will know that what was signed was said by John.

3.2 Linguistic challenges and other problems

The usefulness of the signing space and its importance in terms of Sign Language semantics is clear. The way in which this space is used and managed for an English-to-Sign Language MT system is therefore equally important.

Considering the examples from section 3.1, if it is necessary to implement all the different functions of the signing space in an English-to-Sign Language MT system, then several well-known but difficult linguistic problems needs to be addressed and solved.

A brief outline is given of the linguistic problems that would need to be addressed to implement the different functions of the singing space. For further information on natural language processing and problems in the area of natural language and linguistics, including the problems stated here, see [7, 8, 27].

3.2.1 Pronoun resolution

The first example that was given of how the signing space is used, showed a conversation about John getting lost in the city. The signer placed *John* in the signing space, and thereafter referred to John by pointing at the location John occupied.

To implement this function of the signing space for an English-to-Sign Language MT system, one of the linguistic problems that that needs to be addressed can be seen from the following example:

Example 1 *Bob greeted Sue this morning. She just smiled.*

The linguistic problem consists of finding the person to whom the *she* refers. In the example given, the pronoun *she* would refer to *Sue*. *Pronoun resolution* [26] is the process of finding the words to which pronouns refer. These *words* are known as the *antecedents* of the pronouns.

d'Armond Speers [16] discussed many different problem areas concerning machine translation of Sign Languages. One of these problem areas is contextual ambiguity. To illustrate how the use of pronouns cause contextual ambiguity and why it is one of the reasons pronoun resolution is such a difficult problem, consider the following example:

Example 2 *John visited Bob yesterday. He showed him the red car.*

From the example, it is unclear who showed the red car to whom. In other words, because the pronouns *he* and *him* were used, the context has become ambiguous. When ambiguity is found, disambiguation rules are used, or *preferences*

to decide antecedents for pronouns. In pronoun resolution algorithms syntax is often used for resolve the type of ambiguity found in the given example. The disambiguation rule states that because in the first sentence *John* was mentioned first and *Bob* second, the first pronoun *he* refers to John, and the second pronoun *him* refers to Bob.

3.2.2 Efficient use of the signing space

Using the signing space efficiently requires careful consideration.

During the conversation about John getting lost, the signer had to place *John* in the signing space. Theoretically, there are an infinite number of locations where John could be placed. To create a practical solution a finite number of locations have to be defined, and rules must be created that will govern where and when objects are placed, and how long they may occupy that space. An optimal solution of dividing the signing space into a finite number of locations is unknown. A description of how the signing space was divided in this work is given in Section 3.5 on page 32.

If the signing space is divided into locations and an object management strategy is implemented, then directional verbs and the showing of relationships between objects can also be addressed. Relationships between objects are discussed in further detail in the next section.

3.2.3 Relationships between objects

The signing space is used to show relationships between objects.

To implement this feature in an English-to-Sign Language MT system, a way of finding relationships between objects from text is needed.

Several difficulties become clear when investigating this problem. Firstly, objects need to be found, and should be kept track of. Syntax can be used to find

objects by using grammatical structure and part-of-speech information from sentences. Keeping track of objects is complicated however by the possibility that an object is not always referred to in the same way for a given piece of text. One example is the use of a proper noun the first time a person is mentioned and thereafter using pronouns to refer to that person. In this case, pronoun resolution forms only a special case of a larger problem called *anaphora resolution* [26]. Here an *anaphor* will show back to the antecedent. An anaphor could be a pronoun, a common noun, or even a noun phrase.

Assuming that anaphora resolution is performed and track is kept of objects, then the difficulty remains that relationships can be given at any instance during text, and between any objects. In other words, the objects involved between which the relationship exists, are not always kept close together. Consider the difference between the two sentences in the example:

This is John's car.

The car, which was in an accident, belongs to John.

Another difficulty arises from the fact that many relationships are never explicitly stated, but can be deduced from logical reasoning. See the following example:

John drove into a tree.

Using logic, it is concluded that John must have been driving a vehicle when he drove into the tree. No mention of the vehicle is ever made, but in Sign Language it is important to show that John did not hit the tree, but that his vehicle did.

3.2.4 Role playing

To implement role playing in an English-to-Sign Language MT system, it is necessary to resolve for direct words, the speaker and the addressee. Resolving the speaker and addressee requires the use of *discourse analysis* and *logic*.

To investigate the complexity of this problem fragment of text where John is

in conversation with Mary is examined:

Example 3 *"What are you doing this holiday?" John asked.*
"I really don't know."
He looked bemused and said: "Really? Not even an idea!"
"Well, " Mary smiled, "Like Peter said: 'We will see when we get there.'."

Each sentence shows a different way how, and at what time the speaker can be announced when he is speaking. In the first sentence, the speaker is announced after the words he spoke. In sentence two, the speaker is never announced. Sentence three announces the speaker before he speaks his words, and in the last sentence, the speaker is announced during the words she is speaking.

In the example it is only explicitly mentioned twice that a person is speaking, *John asked*, and *John said*. Logic is needed to conclude that Mary is also speaking.

Finally, from the fourth sentence it can be seen that role playing can be used recursively.

3.2.5 Towards an implementation

The different functions of the signing space have been discussed, and the problems mentioned to reproduce them in an English-to-Sign Language MT system.

For this thesis only the problem of pronoun resolution, and the efficient use of the signing space will be addressed. The finding of relationships between objects, and role playing is left to be addressed in future work.

In the next section related work done in on pronoun resolution is discussed.

3.3 Related work

Pronoun resolution falls into a larger category of problems known as anaphora resolution. A detailed overview of anaphora is given in Appendix B. In this

section the strategies that have been proposed to perform anaphora resolution with pronoun resolution in particular is discussed.

An excellent survey on the state of the art in anaphora resolution research is given by Ruslan Mitkov [26]. In his report he discusses the importance and relevance of *constraints* and *preferences* for anaphora resolution. A constraint refers to a condition that must be met when possible candidates for an antecedent are selected. A preference is a prescribed rule of selecting an antecedent from the possible candidates. Mitkov also gives a categorization of the different anaphora resolution approaches: Traditional approaches, which include algorithms such as the focussing algorithm of Candice Sidner [33], alternative approaches such as statistical algorithms, and knowledge-poor approaches.

Most anaphora resolution algorithms make use of similar concepts and ideas. If the algorithm finds an anaphor, a set of possible *candidates* are selected. This selection is subject to constraints such as *person*, *number*, and *gender* agreement. By applying these constraints, if a pronoun *he* is found for example, it is then clear that *Sue* cannot be a candidate because there is no gender agreement. After the candidates have been identified, the algorithm uses preferences to select the most likely candidate as the antecedent. Preferences are different from one algorithm to the next.

In the research of Bruce Wooley [43], he experimented with a simple rule based system to resolve only the pronouns *they* and *them*. His model took a piece of text and identified noun phrases with a part-of-speech tagger. From the noun phrases, he then identified what he called *plural noun phrases*. A plural noun phrase is a noun phrase that contains plural nouns and singular nouns connected by *and* or by comma delimiters. He tested two simple rules for his system called *nearest prior plural noun* and *the first prior plural noun*. *Nearest prior plural noun* was a rule stating that the pronoun always pointed to the plural noun phrase that was closest to the pronoun, prior in the text. Using this rule he achieved a success rate of just over 50%. The other rule that he used was *the first prior plural noun*, which meant he assumed that the pronoun always pointed to the plural noun phrase closest to the beginning of the sentence that the pronoun was found in.

With this rule he achieved a success rate of over 75%. Wooley noted that the one major problem he had was when the text had a subject that was referenced throughout the text by either *they*, or *them*, but where that plural noun phrase was only given at the beginning of the text. When he disregarded those pronouns, he achieved a success rate of over 90%.

Most of the traditional methods for pronoun resolution uses the work of Sidner [33], which itself was built on that of Barbara Grosz [19]. Sidner stated that a discourse always has a central theme, which she called the focus of the discourse. In addition, she stated that anaphors found in a text refer to the focus, and that if it is known what the focus is at all times then anaphora resolution can be done. Her algorithm had focus registers to keep track of the current focus, and these registers were updated at the end of each sentence. The four registers she used were the *current focus*, the *alternate focus list* where old foci was stored, the *focus stack*, and the *actor focus*. When an anaphor was found, *interpretation rules* were used to identify candidates from the registers, and using syntax and semantics, the most likely antecedent was then selected. Sidner's approach, and similar algorithms are called *centering algorithms*.

Saliha Azzam, Kevin Humphreys and Robert Gaizauskas [10] slightly modified Sidner's algorithm. Instead of using whole sentences, they identified *elementary events*, which are only a part of a sentence, and added two additional registers to the algorithm. They added an *actor focus stack*, and an *intrasentential alternate focus list*, used only in the current elementary event to identify its candidates. Their conclusions of this method showed that a good analysis of the syntactic and semantic structure is needed to yield good results. This conclusion can be expected for all similar approaches because all the decisions are based on the results of the syntactic and semantic analysis.

Marilyn Walker [42] extended Sidner's ideas by saying that the focus of the discourse was not always local to a sentence, or discourse segment, but to the discourse itself. In the algorithm that she proposed, she replaced the stacks that Sidner used with a cache. The cache model is based on the principle used by a cache in a computer. Once some data is accessed, it is stored in the cache. If the

cache is full, it will replace data that has not been accessed for the longest time. The model implies that the most recent foci will almost always be stored in the cache. It also shows that these foci are not dependent on the discourse segment that is being resolved. In other words, if a discourse segment is being resolved, the possible candidates stored in the cache are not limited to a specific discourse segment, but are the most recent foci from all previous text.

Michael Strube [37] took the idea that a discourse is centered round a focus point in another direction. His algorithm is modelled after a *hearer's attentional state*. In other words, he argues that during a discourse, as the hearer receives information, he keeps and updates a list of what the discourse is about and prioritize the items in the list. If then for instance a pronoun is used in the discourse, the hearer uses the list to determine the antecedent for the pronoun. Instead of all the registers that Sidner use, Strube's model has only one structure called the *S-list*. The S-list contains all the discourse elements of the current and previous sentences. When a new discourse element is found, using specified rules, it is given a ranking and is then inserted into the list according to that ranking. When a pronoun is encountered, a lookup is made through the list until the first element is found that matches the constraints of the pronoun. The list is then updated again. Strube assessed that in the worst case, his algorithm performed as well as other centering algorithms.

In some alternative systems, Roland Stuckardt [38] used and modified Noam Chomsky's *binding theory* [9] to perform pronoun resolution. His algorithm identifies and resolves *non-reflexive pronouns*, *reflexive pronouns* and other common nouns. The pronouns *he* and *she* are examples of non-reflexive pronouns while the pronouns *himself* and *herself* are examples of reflexive pronouns. Constraints are again categorized as either *morphosyntactic agreement*, in other words, person, number and gender agreement, or *syntactic constraints*. The interested reader can see [7] for a formal overview of binding theory. Short binding theory states that reflexive pronouns will always have an antecedent present to its *local domain*. Non-reflexive pronouns on the other hand will have an antecedent that is located outside of its local domain. *Nonpronominal nouns* are observed to have

their antecedents located possibly inside or outside of the local domain. A non-pronominal noun is an anaphor which is a noun, and not a pronoun. The observation of these syntactic constraints is defined by Chomsky as the three *binding principles*. He defined a *binding relation*, a *c-command relation* and a *binding category* that states how a local domain for an element is identified. Stuckardt used these binding principles to define an algorithm consisting of three phases. Firstly a candidate list is generated by applying constraints. Next, *preference criteria application* and *plausibility sorting* is performed. Finally the antecedent is selected.

Niyu Ge, John Hale and Eugene Charniak [18] developed a statistical approach for anaphora resolution. As their constraints, they calculated probabilities for all candidates that were found in the current and previous two sentences. For example, if the pronoun found was *he*, a candidate *Bob* would end up with a higher probability than *Sue*. This would be due to the gender agreement of the *he*, and *Bob*. With their model they achieved an accuracy of about 85%. In their findings, they concluded that gender and number agreement was one of the most important components in pronoun resolution. When they excluded this information from their model, the model yielded poor results.

Because of the time needed, and complexity of semantic analysis to do anaphoric resolution, the latest trend is to develop robust, knowledge-poor systems for pronoun resolution. Ruslan Mitkov [25] developed one such system for technical manuals, and achieved a success rate of almost 90%. His approach can be seen as closely related to the system of Ge, Hale, and Charniak. When a pronoun was found, Mitkov looked for all possible candidates in the current and previous few sentences. He then applied what he called *antecedent indicators* to the candidates to generate a score for each one, and the candidate with the highest score was considered the most likely antecedent. Antecedent indicators again include rules such as gender agreement. If the gender of the pronoun was correct a score of 1 was added, if it was wrong the candidate was penalized by a score of -1.

In the next section an overview is given of the pronoun resolution method that is implemented by the preprocessor.

3.4 A pronoun resolution algorithm

The algorithm that is described in this section was designed by combining ideas from centering algorithms, binding theory, and robust methods mentioned in the previous section.

The algorithm receives a TAG tree object as input, and its first step is to tag the tree with morphosyntactic information. The morphosyntactic information will be used to verify constraints, and include person, number and gender classification for *proper determiners*, nouns and noun phrases. A proper determiner is a determiner that consists of a proper noun. For example, in the noun phrase *John's brother*, the proper noun *John* is used as a determiner to show that the *brother* is his brother. The morphosyntactic information is stored as metadata in the semantic relational database.

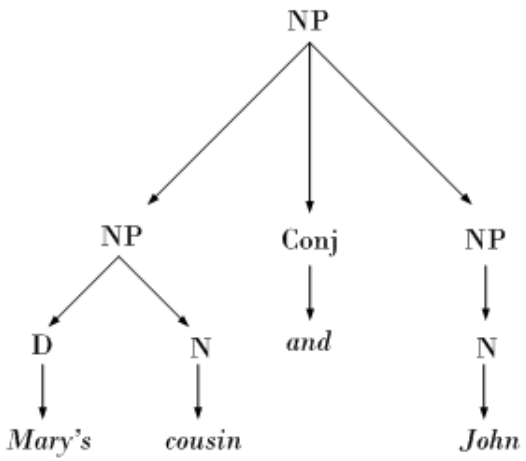

Figure 4: Tree object of *Mary's cousin and John*.

The nodes of a TAG tree can have any number of children. To simplify the traversing of the trees, the children of each node is divided into two groups, one group being called the *left children*, and the other group called the *right children*. The left children consist of the first $\lceil \frac{n}{2} \rceil$ child nodes, where n is equal to the number of child nodes, and the right children will consist of the rest. During the morphosyntactic tagging process, first the left children, then the right children,

and then the node itself will be tagged. The example of the noun phrase, *Mary's cousin and John* contains four nodes that will be tagged with morphosyntactic information (see Figure 4 on page 27 and Table 1). The algorithm tags leaves of the tree with morphosyntactic information from the semantic relational database. A node higher in the tree is tagged by considering the morphosyntactic information of its child nodes. For example if a noun phrase node has the child nodes *John* and *Mary*, then the morphosyntactic information from those nodes will be used to tag the noun phrase node with the gender tag of *male and female*.

Part-of-Speech	**English**	**Person**	**Number**	**Gender**
Determiner	Mary	person	singular	female
Noun Phrase	Mary's cousin	person	singular	male or female
Noun	John	person	singular	male
Noun Phrase	Mary's cousin and John	person	plural	male and female

Table 1: Nodes tagged with morphosyntactic information.

The possible morphosyntactic tags for person, number and gender can be seen in Tables 2, 3, and 4 on page 29.

Person
Person
Object
Both

Table 2: The possible person tags.

Number
Singular
Plural

Table 3: The possible number tags.

Once the TAG tree contains morphosyntactic information, it is traversed again and the algorithm will attempt to resolve pronouns it encounters. The classification of pronouns can be seen in Table 5 on page 29. The classification was

Gender
Male
Female
Male or Female
Male and Female
Male and/or Female
Neither

Table 4: The possible gender information tags.

done with consideration of Sign Language. Pronoun resolution is performed to identify where a signer must point, so that he may indicate the correct person. First and second person pronouns are special cases where the signer will always point to himself or the observer, and is therefore grouped in there own groups. The non-reflexive and reflexive pronoun classifications are taken from Chomsky's binding theory [9]. Pronouns used as determiners are again seen as a special case and they are grouped together.

First person	**Second person**	**Non-reflexive**	**Reflexive**	**Determiner**
I	You	Us	Himself	His
Me	Your	He	Herself	Her
We	Yours	She	Ourself	Hers
My	Yourself	Him	Ourselves	Our
Mine	Yourselves	Her	Themself	Ours
Myself	They	Them	Themselves	Their
				Theirs

Table 5: Pronoun classification

As the tree objects are traversed, two history lists are updated with all noun phrases, nouns, proper determiners, and resolved pronouns which were marked with a *person* tag. The first list contains the items found in all the sentences, the second list, called the *robust list*, is a list containing only the items from the current and previous two sentences. There are two motivations for using the robust list. The robust list will contain the most recently mentioned people, and if the text focuses on a person, then the probability will be good that the

person is contained in the robust list. Secondly, using a robust list will save time. Considering every item in the history list when a pronoun is found will take a lot longer than just considering the last few items found. When an item is added to the history lists, it is first checked for *definiteness*. To clarify definiteness, consider the difference between *the man*, and *a man*.

Once a pronoun is encountered, the algorithm proceeds with finding a list of possible antecedents called the *candidate list*. The candidates are selected from the history lists. The robust history list is searched first for candidates. If no candidates are found, the full history list will be searched. The way in which candidates are selected from the history lists depend on their pronoun classification.

For a first person pronoun such as *I*, only one candidate is considered, and is called the *speaker*. The implementation that was created assumed that the speaker would always be the signer. Note that this is only because role playing was not implemented. If role playing is implemented, the first person pronoun will show to the speaker at that moment.

The same argument is used for second person pronouns. Only one candidate is considered called the *addressee*. Again, in the implementation the addressee was always assumed to be the observer.

The candidates for reflexive pronouns are selected by looking at items from the current domain. If the items match, or could possibly match the morphosyntactic constraints, they are selected as candidates. To clarify this, say that the pronoun *himself* was found. The gender classification for *himself* is male. If one of the nodes contains a gender tag *mail or female*, for example the node *cousin*, and the other constraints are satisfied, then the item will be selected as a candidate.

The candidates for non-reflexive pronouns are selected by looking at items outside the current domain. Candidates for pronominal determiners are selected as any item that match, or possibly match the morphosyntactic constraints. Pronouns used as determiners give extra information during a discourse about some

noun, for example *his brother*. The antecedent of the pronoun could potentially have been mentioned anywhere previously in text, and therefore all possible candidates must be considered.

When the candidate list is created, preferences are used to select an antecedent. If the candidate list consists of only one candidate, then that candidate is accepted as the antecedent. If more than one candidate was selected, a score is calculated for each candidate. Scores are calculated in the following manner: for an exact match of a morphosyntactic constraint, a weight of two is added to the score. A possible match will receive a lesser weight of one. If the candidate has definiteness, a further value of one is added to the score. The candidate with the highest score is selected as the antecedent.

If more than one candidate has the highest score, then syntax is used to select the antecedent. For example, assume the candidates *John* and *James* are identified with *John* occurring first in the sentence. If the pronouns *he* and *him* are then identified in the sentence with *he* occurring first, then the disambiguation rule will select *John* as the antecedent for *he* and *James* as the antecedent for *him*.

It could happen that no candidates are selected for a pronoun. If the first word in a piece of text is a pronoun, then there will be no candidates to select. See the following example:

They all went to school together.

If this sentence was the first sentence in a piece of text, then there would be no candidates of possible antecedents for the pronoun *they*. In this case, the pronoun is entered into the history lists, and its definiteness becomes false.

If a pronoun is resolved, the pronoun is added to the history list, given definiteness, and the selected antecedent is used as the pronoun's *pseudo name*. The pseudo name is necessary because pronouns can now also be selected as antecedents for other pronouns. This implies, that if a resolved pronoun *he* with pseudo name *John* is selected as the antecedent of pronoun *him*, then it can be

determined that *him* is also referring to *John*.

The pronoun resolution algorithm is given as:

1. Starting from the root node n of a tree or sub-tree t and while n is not null:
   (a) If n.part-of-speech is S then increase the domain counter
   (b) If n.part-of-speech is not NP or N and n has left children then resolve pronouns for n.first-left-child
   (c) If n.part-of-speech is NP or N and n has a pronominal determiner on its left side then resolve pronouns for n.first-left-child
   (d) If n.word is a pronoun then
      i. Create a candidate list for n
      ii. Calculate score for candidates
      iii. Select the preferred candidate as antecedent
      iv. Update the history list
   (e) If n.word is not a pronoun but n.part-of-speech is NP, N or Det consisting of a proper noun then update the history list
   (f) If n.part-of-speech is not NP or N and n has right children then resolve pronouns for n.first-right-child
   (g) If n.part-of-speech is NP or N and n has pronominal determiner on its right side then resolve pronouns for n.first-right-child

In the next section the algorithm is described that was designed to help manage the usage of the signing space.

3.5 Creating a scene graph

In this section a design is given of an algorithm that use a data structure called a *scene graph*, to help manage objects in the signing space.

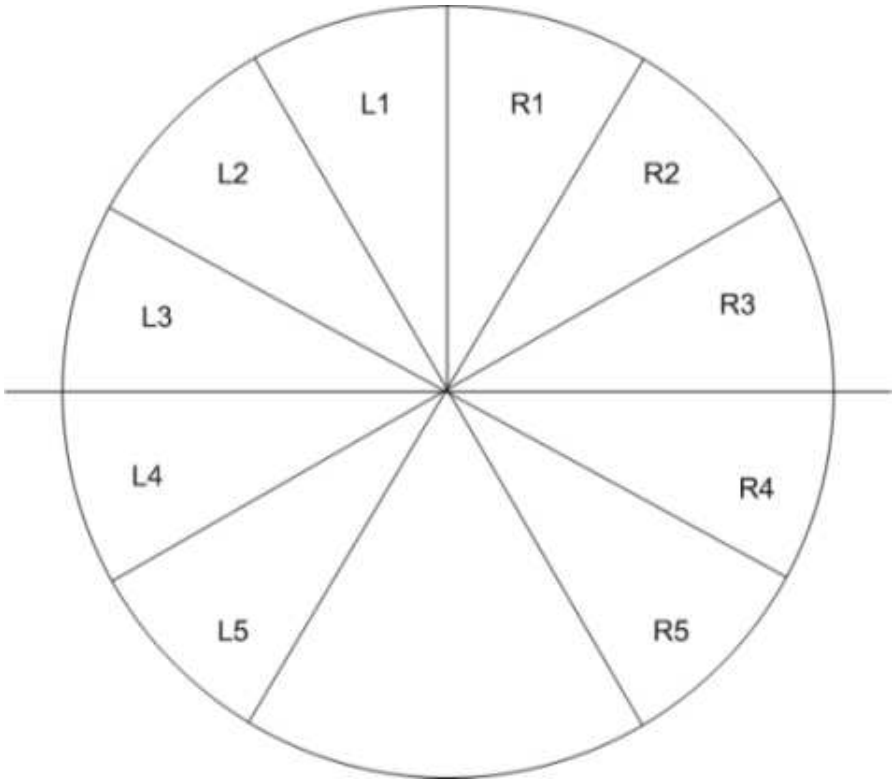

Figure 5: Signing space areas

The scene graph algorithm works as follows: The signing space, if viewed from above the signer, is divided into ten areas. Six of these areas are in front of the signer; the first three are placed to the left, and the last three areas to the right of the sight line. These six areas are *active*. The two locations to the far left and the two to the far right will be *dormant* (see Figure 5). The dormant areas are used as emergency areas that will only come into play if more than six different objects must be placed on the signing space at the same time.

The algorithm receives TAG trees as input, and traverses the trees to find objects that must be placed in the signing space. In the current implementation, only people are placed in the signing space, in future work this will be extended to other objects. For each new object found, a location is assigned with an action. Actions can either be *place*, or *refer*. The refer action will be assigned if the object already occupies a location in the signing space.

The algorithm searches for noun phrases, nouns, or determiners that satisfy person agreement. Singular objects are assigned one area of the signing space. Plural objects are assigned two areas. The exception happens when a plural noun phrase is found that contains either a comma conjunction, or an *and* conjunction. If such a plural noun phrase is found, the number of areas needed by the children of the noun phrase is calculated. If more that six areas are needed, and less that eight, two of the dormant areas become temporarily active. If more that eight areas are needed, but ten or less, all four dormant areas become temporarily active. If more that ten areas are needed, the areas needed are recounted, where a plural object will now occupy only one area.

A history list is kept to help assign areas to objects. If an object is found, the history list is searched to see whether the object has been found previously. If the object is not in the history list, it is added. After an area is assigned to the object, the history list will be updated with the starting location as well as the number of areas that the object now occupies. A group identifier is also assigned to the object. The group identifier is important to help determine whether plural objects still occupy the same area in the signing space as when they were placed. To clarify this, assume that a sentence exists with the noun phrase *John and Sue*. *John* will be placed at one location in the scene graph, and *Sue* in another, with both the same group identifier. If a pronoun is then found, for example *they* that refer to *John and Sue*, it will be known that *John* and *Sue* still occupy their places in the scene graph if the group identifier at each location is still the correct one. If the group identifiers are incorrect then a new location has to be determined for the object. The area that the object will occupy is determined by searching for an open area in the signing space. If no open area is found, then the object that has been in the scene graph for the longest time will be replaced by the new object.

If the object was found in the history list, the scene graph is examined to determine whether the object still occupies that area. If it still occupies the location then a referenced is made to the locations. If it does not, a new location is selected for the object, and the history list is then updated.

To keep the system simple, the scene graph is cleared after every paragraph. The current implementation assumes that when a first person pronoun is found, the signer will point to himself on the sight line. It is also assumed that when a second person pronoun is found, the signer will point to the observer on the sight line.

Together with the area and action, the scene graph also stores an *absolute position*, and a *directional vector* reserved for future use. The absolute position will be used to help adjust objects to be relative to other objects. This will be used to indicate that one object is above another one for example. The directional vector will be used to help indicate directional verbs.

The scene graph resolution algorithm is given as:

1. Starting from the root node n of a tree or sub-tree t and while n is not null:
   (a) If n has left children and n.part-of-speech is not NP or N start allocating locations for n.first-left-child
   (b) If n.part-of-speech is NP or N, n.number is singular and n.word is not a first or second person pronoun then assign location and action for n
   (c) If n.part-of-speech is NP or N, n.number is plural, n.word is not first or second person and sub-trees contain comma or and conjunction then
      i. Count items in sub-trees of n to see if dormant locations should become active
      ii. Set the in_conjunction_flag = true and current_start_location to first active location
      iii. Start allocating locations for n.first-left-child
      iv. Start allocating locations for n.first-right-child
   (d) If n.part-of-speech is NP or N, n.number is plural, n.word is not first or second person and has sub-trees contain no comma or and conjunction then assign action and location for n

(e) If n has right children and n.part-of-speech is not NP or N start allocating locations for n.first-right-child

Assigning location and action:

1. If n.person is not NONE or OBJECT and in_conjunction_flag = true then
   (a) PLACE n at *current_start_location*
   (b) Increment current_start_location
   (c) Update history list
2. If n.person is not NONE or OBJECT and in_conjunction_flag = false then
   (a) If n is in the history list and still occupy its last location then REFERENCE n at last location
   (b) If n is in history list and its last location is open then PLACE n at last location
   (c) If n is new or last location has been taken, PLACE n at a new location and update the history list

3.6 Summary

In this chapter the different roles of the signing space were described. Linguistic and other problems were discussed that needs to be addressed if all the different roles of the signing space have to be implemented in the preprocessor. Finally the pronoun resolution algorithm, and the scene graph algorithm were given that was implemented by the preprocessor.

The next chapter discusses how to create prosodic information from text for a Sign Language implementation.

Chapter 4

The visual component of Sign Languages

This chapter considers the *emotional dynamics* of a language, and argue that the meaning of language is not only contained in *what* is communicated, but also in *how* it is communicated. The preprocessor therefore improves the quality of an English-to-Sign Language machine translation by finding the *how* component. Three different ways of creating non-lexical information for text is described, and the *expressiveness function* is defined as a way to mathematically describe emotional dynamics. This provides a way to evaluate how well the non-lexical information that we create, will compare to real world expectations.

4.1 Introduction

It is not always clear what people mean, when one only know the words they used. Often there is also a need to know how those words were used. To clarify this, consider the following example of two questions that can be asked and answered in an English discourse.

Example 4
Q: Do you know that John drank all of the coffee?
A: He did?
Q: Do you know whether John drank all of the coffee?
A: He did!

The answers to both questions contain the exact same string of words, but their implied meanings are different. The person answering the first question is unsure whether John drank all of the coffee. The person answering the second question is sure. The hearer, listening to the two responses, can tell the difference by extracting intonational elements from what he hears. These elements include pitch, accent, syllable length, the loudness of each word, and the rhythm of how the sentence was spoken. When all these elements are grouped together, it is called *prosody* for spoken languages [28]. From the above example it is clear that prosody plays an important part in meaning, and therefore that meaning is not only contained in the lexical items of a sentence.

Considering the questions from Example 4 on page 37, a hearer will have difficulty to interpret the two responses correctly if they are both spoken monotonically. The same problem exists for an observer in Sign Language. Without any prosodic features, it becomes impossible to interpret the correct meaning of what the signer is signing.

Currently, researchers have a good understanding of prosody and its role for spoken languages. More importantly, the relationship between how a sentence is spoken, and what is meant by how it is spoken, is well understood [14]. To give an example, it is known that if a yes-no question is asked, the pitch of the sentence will go up at the end. Therefore it is known how the pronunciation of a word must change if some specific meaning must be conveyed. Imagine that a person wanted to investigate yes-no questions, and in particular was interested in the intonation of a speaker at the end of each question. Assume further that the person noted that the pitch of the speaker's voice rose at the end of each question. The pattern that the person observed is called a *stress pattern*. The relationship between prosodic information and its implied meaning is called, in general, *stress patterns*. Research is still required to understand such stress patterns in Sign Languages as well as stress patterns for spoken languages [14].

In Sign Language words are signed and not spoken and it is impossible to use vocal inflexion as a way of generating prosody. A signer will use body language,

facial expressions, and the speed and movement of the signs he executes to create the same effect as intonation [17]. It is mostly not difficult to see when a signer stresses a sign. This is true even to people who do not understand Sign Language [14].

Although it is easy to observe prosodic elements when a person is signing, it is more difficult to find stress patterns for Sign Languages than it is for spoken languages. In a spoken language, the letters of each word defines that word, and changing a single letter changes the whole meaning of the word. This means that prosodic elements such as pitch are not required to define the word, and the boundary between the lexical item and its prosodic information is clear. For Sign Languages, due to non-manual signs, this boundary between the lexical item and its prosodic information becomes less clear, which makes it harder to observe patterns.

For a while, researchers were unsure whether Sign Languages contained stress patterns, or whether this phenomenon was common only to spoken languages [14]. Furthermore, if Sign Languages did contain stress patterns, it was unclear whether there would then be similarities in stress patterns between the two types of languages. In a pilot study by Geoffrey Coulter [14] on emphatic stress in ASL, he states that the original studies on this matter led researchers to believe that there were no similarities. Coulter argues that due to inconsistencies in the way these tests were set up, the results were inaccurate. These inconsistencies include for instance the same word being stressed in syntactically different sentences. One problem in doing so, is that signs made prior to the one being stressed, influences the sign to follow in terms of starting position and starting moment, which are all factors the researcher is trying to observe. Because the way signs are signed can physically be different, it caused the observed, stressed sign to look as if it behaved inconsistently.

Coulter claims that in his experiments he restricts these factors to a minimum, and from the results of his tests it is clear that the two types of languages do share similarities. This finding of Coulter is important to this work because it motivates an approach to use semantics to generate prosodic information. In

other words, the same argument can be used as for spoken languages to say that if it is known what is said, then it is also known how it should be said, or in the case of this work how it should be signed.

In the next section related work is discussed that make use of, or include prosody.

4.2 Related work

Scott Prevost [29] gives a full literature overview of prosody and different areas of research related to the subject. Appendix C contains a description of components that constitute prosody for spoken languages, for example what a *prosodic phrase* is.

The goal of this section is to discuss the different approaches that have been used to define prosody, and its relation with natural language. In many instances where prosody is used, the application will either be natural language synthesis or natural language recognition. The automatic prosodic generation of *text-to-speech* (TTS) applications gives us useful insight into the similar problem of finding prosodic information for an English-to-Sign Language MT system.

There are two main approaches for automatic prosodic generation: using syntactic analysis or using semantic analysis. In the overview of TTS synthesis by Thierry Dutoit [4] he describes the model of many commercially developed TTS systems. The aim of these systems is not to generate the most appropriate intonation for the speech being synthesized, but rather acceptable intonation in the sense that what is said should be plausible. This can be done by using syntax. The first phase of TTS consists of natural language processing, and one of its components is the *syntactic-prosodic* parser, which is responsible for automatic prosodic generation from the syntax.

John Bear and Patti Price [12] describe a method of including prosodic features in the grammar of a language. In this case the prosodic element that they focus

on is the rhythm of a spoken utterance. Their approach is to modify a grammar to include these prosodic features. For example, suppose that the grammar was $A \rightarrow BC$. They will then modify the grammar to $A \rightarrow B\ link\ C$, with *link* defined as a new grammatical category, called *prosodic break indices*. The *link* gives an indication of what the length of the break must be between spoken utterance B and C. This approach of adding prosodic information into the grammar has also been used in Sign Language to incorporate non-manual signs into its grammar [45].

In the work done by Laurance Danlos, Eric Laporte, and Francoise Emerand [15], they constructed a *syntactic-prosodic grammar* for French. They call their system a *syntactic-representation-to-speech* system. In this system, they define fifteen prosodic markers to mark an analyzed sentence. Each marker assigns melody and rhythm to each syllable of the preceding word. The assignment of markers is done according to the part-of-speech category and the position in the sentence of the relevant word. They furthermore define the markers in a hierarchy, so that the markers are dependent on one another. In the work done by Pete Sharp [31], he gives a similar approach for Tokyo Japanese.

Richard Oehrle [28] states that prosody is determined by the manner in which the grammar of a language controls the phonetics of each word in a sentence. He notes that prosody is global, in the sense that the intonation of a sentence would contribute to help clarify the meaning of that sentence. However, the actual prosodic elements are found locally, because stress and pitch are focussed on specific words. He further argues that (a) a prosodic phrase can be seen as a unit of speech that can be analyzed, and that (b) the type of prosodic phrase, and therefore the type of prosodic elements that can be found, are restricted by the underlying grammar of that phrase.

Mark Steedman [34] worked on intonational and syntactic structure, and he argues that in most cases it is not appropriate to use syntax to generate intonation. He motivates this by saying that the syntactical structure looks nothing like the intonational structure of the language. He then shows that a *combinatory*

categorial grammar [1] can be used to construct a grammar for a language that would have the same structure as the intonational structure, and can then be used together with semantics to generate intonation.

In other work by Mark Steedman [35] he gives a model to show how semantics can be used to generate intonation for a sentence. The model is based around the structure of a conversation. When two people communicate, one person speaks and in reply the second person may respond. When generating the intonation, three factors are considered: Who is speaking, does the speaker agree with himself, in other words, is he sure or unsure of what he is saying, and thirdly, does he agree with what the other person had said.

Concept-to-Speech (CTS) systems use semantic information to create realistic sounding speech from text [21]. The methodology is based on the argument that certain stress patterns in sentences give an indication of the meaning of that sentence. Stated in reverse, if it is known what the meaning of a sentence is, then it will be a clear what the intonation should look like. In this case, meaning refers for instance to the knowledge that a question is asked, or that a contrast is given, or that new information is stated. The concept of a CTS system is in principle straightforward: Find some way to characterize a given sentence, know which stress pattern is associated with that classification, and then assign the correct prosodic features accordingly.

One CTS model was created by Laurie Hiyakumoto, Scott Prevost and Justine Cassell [21]. Their CTS system consists of two modules. The first module is made up of three algorithms. The first algorithm identifies and marks *new information*, the second algorithm identifies *contrastive information*, and the third algorithm divides sentences into smaller constituents and then uses the words marked for accent to classify them as either *rhemes*, or *themes*. Rhemes contain new information, while themes contain information already given. The second module uses the themes and rhemes to map intonational information onto sentences.

Other methods of finding intonation include computational models. In the work

[1] Combinatory categorial grammar is a grammar formalism based on logical type-theory.

done by Anand Venkataraman, Luciana Ferrer, Andreas Stolcke and Elizabeth Shriberg [41], they give a computational model to generate prosody for dialogs. They created a *Hidden Markov Model* based dialog act tagger and trained it with unlabelled data. Computational models are not considered in this thesis, but the interested reader may refer to [29] for more information on this topic.

4.3 The expressiveness function

Sign Languages are expressive languages. A signer will make use of his whole body, facial expressions and the movement of his signs to help express meaning when he is signing.

Assume there is a person who does not understand Sign Language. If he was asked to observe a conversation between two signers, he might not follow what the conversation is about, but what he would probably be able to say, is the emotion that the signer is expressing at a specific instance in time. A similar view can be taken on written work. A writer will use the words available to him, to express himself.

Assume that the extent to which a person expresses himself at a given moment can be measured. *Expressiveness* is then the measurement of expression. The different emotions a person could use to express himself are called *expressiveness states*. The *expressiveness function for emotion* is then defined as the change of expressiveness over time.

Using the expressiveness function in a practical way, it can be argued that some emotions, for example *love*, are positive, some are negative such as *anger*, and an emotions such as *contentedness* is neutral. By assigning a numerical value to each expressiveness state, a mathematical function can be created that will describe the change in expressiveness over time.

Example 5 *Suppose that there are only three emotions through which a person could express himself: love, hate, or contentedness. A value is assigned to each of these three possible expressiveness states as:*

$$love = 1,$$
$$hate = -1, and$$
$$contentedness = 0$$

The following text shows John answering a question about his views on computer games:

I love computer games, but not when they are too difficult, those ones I hate. It is fine to play adventure titles, but driving simulators I really enjoy.

The expressiveness function for this text can be seen in Figure 6 on page 45. The graph indicates how the expressiveness of the speaker changes over time. In the example it can be seen that the person begins by expressing himself in a positive manner by using the word love. *Later on he uses negative words to express himself by using* difficult, *and* hate. *Finally, he ends by positively expressing himself through the use of the word* enjoy.

Looking at Sign Language, the movement of signs also plays a role in terms of size of movement and the speed of the movement. Assume the speed of movement could be standardized, and therefore the time of execution, and also the size of any sign movement to one unit. In other words, under normal conditions, the sign will execute at one unit speed and will use one unit of space. It is now possible to assume that for some reason, the speed of movement might increase or decrease, or that the space used might be more or less than one unit, for example, because a word is being stressed. Observing the speed of movement and space used over time, an expressiveness functions can be created that describe the rhythm of a sentence.

Considering the expressiveness function of form in terms of the speed of movement for the given example (see Figure 7 on page 45), then each bar in the graph represents a word. The higher the bar in the graph, the faster the movement will be. In the example, five words are faster and two slower than they would normally be. Intonational breaks such as when punctuation is found, is also included in the graph. The height of the bar in this case will indicate whether there should be a longer or a shorter break than usual.

Considering the expressiveness function of form in terms of space used for each

movement (see Figure 8 on page 46), then again each bar in the graph represents one word. Each bar indicates how much space is used. If the height of the bar is zero, then there was no movement. This will indicate an intonational break for example after punctuation. The higher the bar in the graph, the bigger the movement will be. In the example, seven words were bigger than their average, while one word took less space.

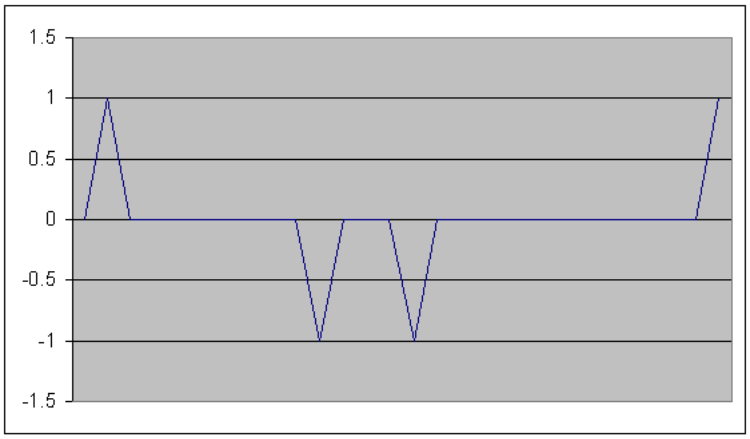

Figure 6: Expressiveness function: Emotional class

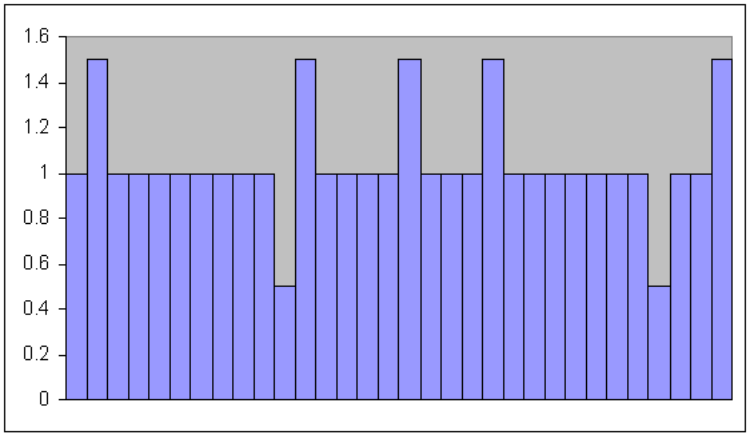

Figure 7: Expressiveness function: Form (speed)

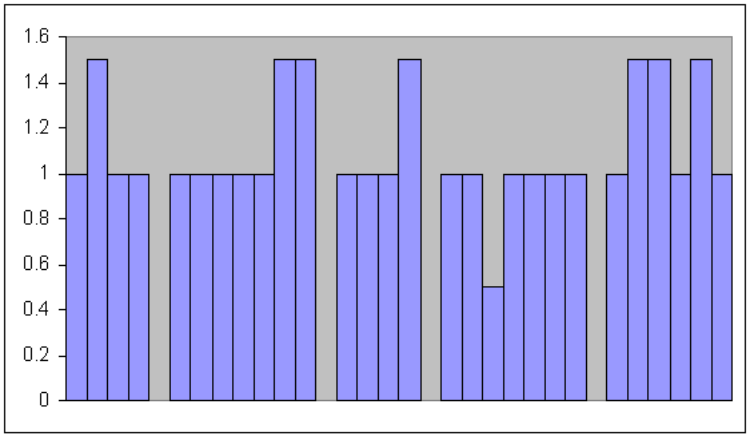

Figure 8: Expressiveness function: Form (space)

4.4 A prosodic model for Sign Language

There are four factors that play a role in Sign Language prosody. Firstly, signs are created by the use of hands, wrists and arm movements. These movements can change in terms of the speed of execution, and the range of the movement. Secondly, facial expressions are used to create emotional information. Body language plays a similar role to facial expressions, and lastly the rhythm of what is signed is important in terms of pauses between words, and how the words flow from the one to the other [17].

The approach to generate prosodic information is based on the idea that sentences can be seen as having two levels of detail. At one level specific words are considered, while at the second level phrases are looked at. During prosodic generation, the TAG tree objects will be tagged with metadata. Different types of tags are used to address the four factors of Sign Languages prosody, some at word level, and others at phrase level. The semantic relational database given in this work is used to tag the tree objects with metadata.

The prosodic generation phase of the preprocessor consists of three modules, each performing its own task, but all of them creating prosodic information. The first module considers words and sentences, and tags the TAG tree object

with *expressiveness information*. The second module identifies and tags words that should receive accent. The third module examines sentences, and identifies phrase boundaries to indicate rhythm.

Three types of expressiveness information are defined. The first is called *form*. Form is information that gives a guideline to define the parameters for sign movement. To address the speed of the movement, form could be tagged as *explosive* or *delayed*, and to address the range of movement, form could be tagged as *exaggerated* or *understated* movements. The second type of expressiveness information is called the *emotional class*. The emotional class determines the emotional state of the signer when the sign is signed. The emotional state is associated with the facial expression of the signer but also influences speed of sign movement and body language. Thirdly a sentence is classified as having a *type*. The type of a sentence is tagged as a *statement*, *question*, or *exclamation* and is classified by considering the punctuation of the sentence. The type of the sentence is used to help indicate body language. The semantic relational database is used to tag the TAG tree objects with form and emotional class information. Punctuation is used to define sentence type.

The second module is based partly on a CTS model created by Hiyakumoto, Prevost, and Cassell [21]. The reason why ideas from this CTS module is used, is because the approach of CTS modules focus on semantics and not syntax. Two of their algorithms are used; one to extract and mark new information, and one that finds contrasts. The words identified, are tagged to receive accent.

The first algorithm identifies new information. The input text is scanned, and every time a noun, verb, adverb, or adjective is encountered, a lookup of that word is done in a *word history list*, and in all the *equivalence lists* of each word in the history list. The history list contains all the words that have already been found in the text, and which have been marked as stressed. If a word is not found in the history list, or any equivalence list, then the word is marked for stress. Equivalence lists are created with the help of the semantic relational database, where every word in a list, is similar to a word found in the history list. As an example, if the word *dog* was found, marked for stress, and placed

in the history list, then the words *pooch*, *hound*, and *heel* amongst others would be placed in its equivalence list. If a lookup of a word is made in an equivalence list, a match can only be made if the part-of-speech category of both the *lookup word* and the words of the equivalence list match. If the word is found in an equivalence list, then the word is marked as *inferable*, which means it should still be stressed, but less than that of words marked as new information. After every paragraph, both the history and equivalence lists are cleared.

The first algorithm, as taken from [21], is given as:

For each word w

1. If w is a noun, verb, adjective, or adverb, and $w \notin history()$, and $w \notin equiv(x)$, for any $x \in history()$:
   (a) tag w as a focused item
   (b) add w to $history()$
   (c) create $equiv(w)$
2. If w is a noun, verb, adjective, or adverb, and $w \in equiv(x)$, tag w as inferable.

The purpose of the second algorithm is to find contrasts, or contrastive relationships between words. In the sentence *John was sad, but Sam was happy*, the word *happy* is contrastive to the word *sad*, and is therefore marked to receive accent. Every time a word is encountered, it is compared to every other word of the same part-of-speech category to find a possible contrast. This is done from most recent to least recent word in the list.

The second algorithm, as taken from [21], is given as:

1. For each word u:
   (a) For each word v on the history list, from most to least recent:
      i. For each word in $\{w : w \in contrast(v)\}$:

A. if $u = w$ then mark u as a contrastive focus;
   end the search

B. else add u to the history list;
   generate and store $\{w : w \in contrast(u)\}$

The words that are identified to receive accent amplify their expressiveness information. To give an example, if a word was tagged with *exaggerated* expressiveness information, then if the word is also identified to receive stress, then it is known that the word should be stressed by making the sign movement bigger. If it was marked with *understated* expressiveness information, then the movement would be made smaller.

The third module of the preprocessor finds phrase boundaries in sentences. These boundaries help to indicate the rhythm of the sentence.

Boundaries are calculated by finding either punctuation in a sentence, or conjunctions. A conjunction is a part of speech that connects two words, phrases, or clauses. There are three types of conjunctions: *simple co-ordinate conjunctions*, *correlative co-ordinate conjunctions*, and *subordinate conjunctions*. A simple co-ordinate conjunction will connect two parts that have the same importance. For example, looking at *John and Mary*, the word *and* is a simple co-ordinate conjunction. A correlative co-ordinate conjunction is found when a pair of words connects two parts that are similar. For example, if looking at *both Mary and John*, then *both* plus *and* forms the co-ordinate conjunction. The co-ordinate conjunctions can be seen in Table 6. If one phrase is subordinate to another, the conjunction will be a subordinate conjunction. In the sentence *John went to bed after he got home*, the word *after* is an example of a subordinate conjunction.

Boundaries are chosen after punctuation occurs, after the first part of a correlative co-ordinate conjunction is found, and before simple co-ordinate conjunctions are found. A break is created before the first part of the correlative co-ordinate conjunction to indicate that the parts that will be connected is about to follow. A break is created before a simple co-ordinate conjunction to indicate that the one part of the two that are connected has ended. The break after a correlative

Correlative co-ordinate conjunction	Simple co-ordinate conjunction
both	and
either	or
neither	nor
not only	but
whether	for
	yet
	so

Table 6: Co-ordinate conjunction classification

conjunction should be shorter than a break after punctuation, or the break before a simple co-ordinate conjunction. If punctuation is found right before a simple co-ordinate conjunction, it will count as only one standard break. The sentence *Neither Mary, John or Peter went on holiday*, will be broken up as *Neither (1) Mary (2) John (2) or Peter went on holiday* where the numbers show the breaks and their length the sentence. A shorter break is indicated by one while a standard break is indicated by two. The algorithm to do boundary creation is given as:

1. For each tree t and each node n in t:
   (a) If n is punctuation create a standard break after n
   (b) If n is a correlative co-ordinate conjunction create a short break after n
   (c) If n is a simple co-ordinate conjunction and no break have been created before n, create a standard break before n

4.5 Summary

This chapter introduced prosody for spoken and signed languages, and discussed its importance in terms of meaning. It was argued that because of Coultier's findings, a semantic approach to create prosody can be used. The expressiveness

function was defined as a means to describe how expressive a piece of text is. Finally the different algorithms that are implemented by the preprocessor to create prosodic information was described.

In the next chapter an analysis is done of the results from an implementation of the preprocessor in a real world application.

Chapter 5

Analysis

This chapter investigates how the preprocessor performs in a real world implementation.

5.1 Introduction

To test the preprocessor, several different software packages were either used or had to be written. To find grammatical structure and part-of-speech information, a software package called *Lem14.0* [5], and which was developed at the University of Pennsylvania was used. The WordNet 2.0 API [6] was incorporated into the main software program to find morphological information about words, and to create word lists such as the equivalence lists used in the prosodic generation modules. The semantic relational database structure was created with SQL, and a special database data transfer utility was written to copy the data stored in the original WordNet database structure to the new structure. Any additional information was added to the new database with SQL. A program was written to read the files created by the Lem software package. The algorithms described in this work were implemented, and the output generated was stored as XML files.

5.2 Analysis approach

The testing of the preprocessor was approached in the following manner: each component of the preprocessor was tested and the results analyzed separately, where different types of text were used to test each component.

The process for finding grammatical structure and part-of-speech information is firstly looked at, next the pronoun resolution algorithm is analyzed, followed by the scene graph algorithm, and finally an analysis is done of the results created by the prosodic generation algorithms.

Different types of text needed to be tested during analysis. As a result, document were manually classified and grouped into three categories: *media*, *academic*, and *general*. In general the *media* category was defined as all text documents written to be published in the media, including newspaper articles, magazine articles or internet sites. The *academic* category refers to all text documents written for academic purposes, including text books, papers or dissertations. The *general* category refers to all other text documents, including fictional books.

Two samples from each of these three categories were selected and tested with the preprocessor. These six pieces of text were slightly modified when needed. In the next section it will be shown how and why this had to be done. The modified text together with citations of the originals can be found in Appendix F.

5.3 Tree-adjoining grammar analysis

To find grammatical structure and part-of-speech information a TAG was used.

For shorter sentences with about 15 words or less, the results per sentence were usually good and manageable. The number of derivations varied from a few to a few hundred, and in virtually every case an acceptable derivation could be found. When a sentence did not yield any derivation, it was mostly caused by an unrecognized word. The Lem package accepts unrecognized words to have a

noun part-of-speech category, which was sometimes untrue, and the assumption caused the sentence to be grammatically incorrect.

Sentences of greater length, and especially sentences exceeding 20 words, created more problems. The TAG parsing algorithm is calculated to have a time complexity of $O(n^3)$ [23], which became evident as the time of execution became unpractical. For example, a sentence with 13 words could parse in 10 to 120 seconds, and would yield between 30 and 300 derivations, while a sentence with 22 words could take between 900 and 1200 seconds, and yield 1500 to 15000 derivations. After waiting such a long time for a sentence to parse, it still happened at times that an unrecognized word caused no results to be yielded, forcing us to slightly modify the sentence by temporarily replacing the unrecognized word with a recognized one, and then parsing the sentence again. Searching through thousands of derivations to find an adequate one was also a big task.

Sentences found in open text are often longer than 20 words, and will regularly exceed even 30 or 40 words. For the test documents, the original text was modified so that most sentences never contained more than 25 words. This was done by breaking longer sentences into two or three shorter ones.

5.4 Pronoun resolution analysis

Testing the pronoun resolution algorithm with different types of text documents meant that a variety of different pronouns were tested. At worst, the pronoun resolution algorithm performed as well as other pronoun resolution algorithms.

Overall, the pronoun resolution algorithm resolved 87.17% of pronouns correctly in the test cases (see Table 7 on page 55). If any pronouns are removed from the results that did not show back to a person, which happened twice, then the algorithm was successful 89.47% of the time. The algorithm was designed so that when a pronoun was resolved, that resolved pronoun would then be stored in the history list, and was then eligible to be selected as an antecedent for some other pronoun. This means, that if a pronoun was resolved incorrectly, and was

then selected as the antecedent of another pronoun, the other pronoun would also resolve incorrectly. One possible way of eliminating these errors would be to change the algorithm so that resolved pronouns are not added to the history lists. Technically speaking, a pronoun that chooses its antecedent as an incorrectly resolved pronoun ca still be considered to have resolved correctly if the pronoun it chose as its antecedent was indeed an antecedent. In other words, both pronouns show back to the same entity. For interest sake, considering pronouns that selected incorrectly resolved pronouns as their antecedent, and given that those pronouns are legitimate antecedents, then the algorithm was successful 97.30% of the time.

Text	Correctly resolved	Incorrectly resolved	Success
Baby murder suspect	12	0	100%
India's Sharapova	9	6	60%
Fant	7	1	87%
TAGs	12	0	100%
Birthday of the world	6	1	86%
Five days in Paris	22	2	92%
Total	68	10	87%

Table 7: Results from pronoun resolution algorithm.

The document that yielded the worst results was *India's Sharapova* (see Table 8 on page 56). Two of the pronouns that resolved incorrectly were caused by disambiguation rules that selected the wrong antecedent, while the other mistakes were all caused by the two pronouns that resolved incorrectly. This document showed that the algorithm could still be improved by improving the disambiguation rules, and it also showed a major pitfall of centering algorithms, in the sense that one mistake could potentially cause more mistakes.

Disambiguation rules that do not always choose the correct antecedent is a problem that many pronoun resolution algorithms share. This is not unexpected because many of the pronoun resolution algorithms use syntax to resolve pronouns. Researchers have found however that incorporating semantics into their algorithms, greatly increased the complexity of the algorithm while it only

marginally improved the results [7].

The last document, *Five days in Paris*, showed again that the algorithm was sensitive to grammatical structure (see Table 9 on page 57). The last pronoun *himself* resolved incorrectly because it is a reflexive pronoun, and its antecedent is searched for within its current domain. Due to the grammatical structure of the sentence, the domains of the sentence were chosen incorrectly, and therefore the pronoun resolved incorrectly. This is not unexpected due to the use of Chomsky's binding theory principles in the algorithm which makes use of the grammatical structure of a sentence. If the grammatical structure is incorrect then the results of the algorithm will suffer as a consequence.

Anaphor	Selected antecedent	Correct antecedent
she	Teenager Mirza	Teenager Mirza
she	Mirza	Mirza
her	Mirza	Mirza
her	Mirza	Mirza
her	Kuznetsova	Kuznetsova
her	Kuznetsova	Mirza
she	Kuznetsova	Mirza
her	Kuznetsova	Mirza
she	Kuznetsova	Mirza
her	Sharapova	Mirza
her	Sharapova	Mirza
she	Mirza	Mirza
she	Mirza	Mirza
she	Mirza	Mirza
they	people	people

Table 8: Resolved pronouns: India's Sharapova

5.5 Scene graph analysis

The next step was to investigate the results of the Scene Graph algorithm.

During the testing of the scene graph, it was found that the dormant locations

Anaphor	Selected antecedent	Correct antecedent
he	“he”	Peter
him	Peter	Peter
he	Peter	Peter
him	Peter	Peter
he	Peter	Peter
he	Peter	Peter
his	Peter	Peter
he	Peter	Peter
he	Peter	Peter
him	Peter	Peter
her	”her”	”her”
he	Peter	Peter
he	Peter	Peter
he	Peter	Peter
their	children	children
they	children	children
he	Peter	Peter
he	Peter	Peter
he	Peter	Peter
he	Peter	Peter
her	“her”	“her”
he	Peter	Peter
he	Peter	Peter
himself	“himself”	Peter

Table 9: Resolved pronouns: Five days in Paris

of the signing space were never used. This showed that it was acceptable to divide the space in front of the signer into six locations.

After a person was placed in the signing space, no one person was ever moved more than once from its original placement. This means that objects were often re-placed on the location they previously occupied. The whole active signing space was also more or less evenly used (see Figure 9 on page 65 and Figure 10 on page 66). The figures should be interpreted as follows: The x-axis represent the locations that are found in the signing space, while the y-axis represent all the people that was place at some point in the signing space. The y-axis represent

how many times a person would then occupy a certain position. In short, the results imply that that the goal to make it as easy as possible for the observer to remember where objects were placed in the signing space was achieved.

The scene graph did however show two vulnerabilities that become evident from *Baby murder suspect*. In this text, the way that baby Jordan-Leigh is referenced changes all the time, as she is referred to as *the infant*, *the baby girl*, and *the baby* (see Table 10 on page 59). The scene graph does not recognize these three objects as the same person, and therefore places them at different locations, which is an undesirable effect. Items in a discourse that represent the same entity are called *coreferential* and the entity that they represent is called the *referent* (see Appendix B). Finding coreferential items from a discourse is a linguistic problem that would need to be addressed in the scene graph to solve this problem and it is left as part of future work. The second vulnerability is the mistakenly included *Cape police station* in the scene graph. The mistake is caused by the word *police*, which satisfy person agreement. The second problem is caused because the preprocessor focuses on individual words, and does not attempt to identify concepts or ideas as a whole. Mining concepts from text is a linguistic problem many researchers are working on. If a successful solution is found to this problem then many different areas of the preprocessor could be improved.

	L5	L4	L3	L2	L1	R1	R2	R3	R4	R5
Jordan-Leigh's father	0	0	0	0	0	0	1	1	0	0
His daughter Natasha	0	0	0	0	0	1	1	0	0	0
Norton	0	0	2	0	0	0	0	0	0	0
The baby	0	0	0	0	0	0	0	1	0	0
Jordan-Leigh's grandfather	0	0	0	1	1	0	0	0	0	0
The Norton family	0	0	0	0	0	1	1	0	0	0
Cape police station	0	0	0	1	1	0	0	0	0	0
Dina	0	0	4	0	1	0	0	0	0	0
The baby girl	0	0	1	1	0	0	0	0	0	0
The infant	0	0	0	0	0	0	0	1	0	0
Four men	0	0	0	0	0	2	2	0	0	0
Police	0	0	0	1	1	0	0	0	0	0
The 24-year-old suspect in the murder of six-month-old Jordan-Leigh	0	0	1	0	0	0	0	0	0	0

Table 10: Scene graph (placed): Baby murder suspect

5.6 Prosody generation analysis

The prosodic generation algorithms proved to be flexible and could be extended to suite different types of text. It should be noted that only a percentage of words in the semantic relational database was populated with metadata, and because of the little research available on prosodic information for Sign Languages, the database was populated with realistic metadata, but not necessarily the correct metadata. For the purpose of this work, it was important to demonstrate that the system could be used to create realistic prosodic information to be used in a Sign Language application, and therefore the population of the database as is, was viewed as acceptable. The population of the database with correct metadata is left as future work.

The prosodic information was divided into three categories, *type* which was determined at sentence level, and *form* and *class* which was determined at word level. Because all the sentences of all six documents were *statements* further analysis of the *type* prosodic information generated will not be done. In general,

	L5	L4	L3	L2	L1	R1	R2	R3	R4	R5
Jordan-Leigh's father	0	0	0	0	0	0	0	0	0	0
His daughter Natasha	0	0	0	0	0	0	0	0	0	0
Norton	0	0	2	0	0	0	0	0	0	0
The baby	0	0	0	0	0	0	0	0	0	0
Jordan-Leigh's grandfather	0	0	0	0	0	0	0	0	0	0
The Norton family	0	0	0	0	0	0	0	0	0	0
Cape police station	0	0	0	0	0	0	0	0	0	0
Dina	0	0	4	0	1	0	0	0	0	0
The baby girl	0	0	0	0	0	0	0	0	0	0
The infant	0	0	0	0	0	0	0	0	0	0
Four men	0	0	0	0	0	1	1	0	0	0
Police	0	0	0	0	0	0	0	0	0	0
The 24-year-old suspect in the murder of six-month-old Jordan-Leigh	0	0	1	0	0	0	0	0	0	0

Table 11: Scene graph (referenced): Baby murder suspect

the *type* prosodic information is created by considering whether a sentence is a statement, question or exclamation. This means that an analysis of the *type* prosodic information would only show how the author of some text made use of these different types of sentences. In the following sections the expressiveness functions created by the *form* and *class* prosody is examined.

5.6.1 Form analysis

Analysis of the *form* prosodic information created by the preprocessor was evaluated by considering the results, against the content of each piece of text. The results generated for *Coming of age in Karhide* will be discussed.

Coming of age in Karhide is a fictional story about the oldest civilization in the world. The story is told by one of the citizens living in the city, and describes the history and culture of his nation. He also mentions day to day events and activities. The story teller describes in great detail the vast landscape with mountains and rivers, and how the houses are so widely spaced, that no one

house blocks the view of another. He also describes the age of the city and nation in terms of thousands of years.

Considering the result of the *form* prosodic generation in terms of space, two things can be noted: firstly, the sign movements of most of the accented words are exaggerated, and secondly, there are many pauses during the telling of the story (see Figure 11 on page 67). Pauses can be seen in the bar chart when the height of the bar is zero, in other words, when no space was used. This is consistent with the content of the text, and created the idea of greatness, and something that happened over a long period of time.

Looking at the *form* prosody in terms of speed, it can be seen that there is a mixture of signs created faster, or slower than normal, when they are accented (see Figure 12 on page 67). This is also consistent if one considers how the story teller jumps between the history of the nation, and the every day events taking place in the city.

5.6.2 Class analysis

Analysis of the *class* prosodic information created by the preprocessor was done in two phases. The emotional set used was defined firstly as the emotions defined by WordNet plus the emotion *contentment*. It should be noted that any number of classifications could have been made based on research about facial expressions and emotions found in text. The classification however already encapsulated many of the expressions in some of the other classifications, and was viewed as an acceptable first classification (see facial expressions generated by Dean Barker in [11]). The first emotional class set and assigned values can be found in Table 12 on page 62.

The initial categorization of the different pieces of text, was based on the knowledge that different texts serve different purposes. For example, newspaper articles are written to portray events, which could be emotional, and try to be entertaining to a degree. On the other hand, academic material is usually written to only convey facts, and not to stir emotion, or to be entertaining. It is therefore

Emotion	Value
Love	4
Happiness	3
Contentment	0
Surprise	-1
Anxiety	-2
Fear	-3
Anger	-4
Hatred	-5

Table 12: First emotional class set classification.

expected that the prosodic information generated by the preprocessor would correctly show the difference in expressiveness due to the difference in the type of document.

Examining the expressiveness functions of the newspaper articles, and comparing them to the expressiveness functions of the academic material, it shows that the preprocessor did correctly generate functions that were realistic to the types of document (see Figures 13 on page 68, and 15 on page 69). The newspaper article showed emotion throughout the text, while the academic material did not show any emotion.

Examining the expressiveness function for *Baby murder suspect* more closely, it can be seen that the emotions found at the beginning of the text are all negative, while the emotions at the end of the text are more positive. In this case, the article is about a baby girl called Jordan-Leigh who was murdered, and in the beginning of the article they discuss the murder and the suspects, while at the end of the article they discuss the family of the baby, and how they want to honour the baby, and how much support they received from other people. In other words, the preprocessor also generated results realistic to the nature of the document.

Using the first class set definition, although it worked well still showed a shortcoming: although the academic material did not portray any emotion, it was

not devoid of expression. The facts given in the documents were written with a degree of *seriousness*, and the actions described were shown to have some *determination*. The emotional class set was therefore extended to also contain the elements *serious* and *determined* (see Table 13).

Emotion	Value
Love	4
Happiness	3
Determined	2
Serious	1
Contentment	0
Surprise	-1
Anxiety	-2
Fear	-3
Anger	-4
Hatred	-5

Table 13: Second emotional class set classification.

After creating the expressiveness functions for the test documents using the second emotional class set it is found that the academic documents are now more expressive, but also realistic to the nature of those documents (see Figure 16 on page 69).

Therefore, the system implemented is flexible, and given the correct classification of the emotional set, could create prosodic information for many different types of text.

The approach for generating prosodic information however, was partly based on work that generated prosodic information for spoken languages. Ideas were used from a CTS system to identify words that should receive accent. This implies that the algorithms will also suffer some of the shortcomings that these systems experience. Although the prosodic information resembles the nature of the document as a whole, it does not always accurately resemble certain individual sentences. The preprocessor does not attempt to identify language phenomena such as *sarcasm*, and therefore the results cannot be expected to be correct all

the time. One way of overcoming these shortcomings would be to add an element of *discourse analysis* to the preprocessor. The interested reader can see [8] for an in depth study of how people are using discourse analysis to improve their natural language applications.

5.7 Summary

In this chapter an analysis was done of the results from the preprocessor in a real world implementation. Finding grammatical structure and part-of-speech information for a sentence proved to work well for shorter sentences, but less so for longer sentences. The pronoun resolution algorithm yielded good results, and the scene graph yielded acceptable results. The approach for generating prosodic information from text proved to be a flexible system that could create good and realistic results.

It should be noted here that the preprocessor could not be tested for generated Sign Language, as the rest of the SASL-MT system is not yet complete.

In the next chapter conclusions and possible future work will be discussed.

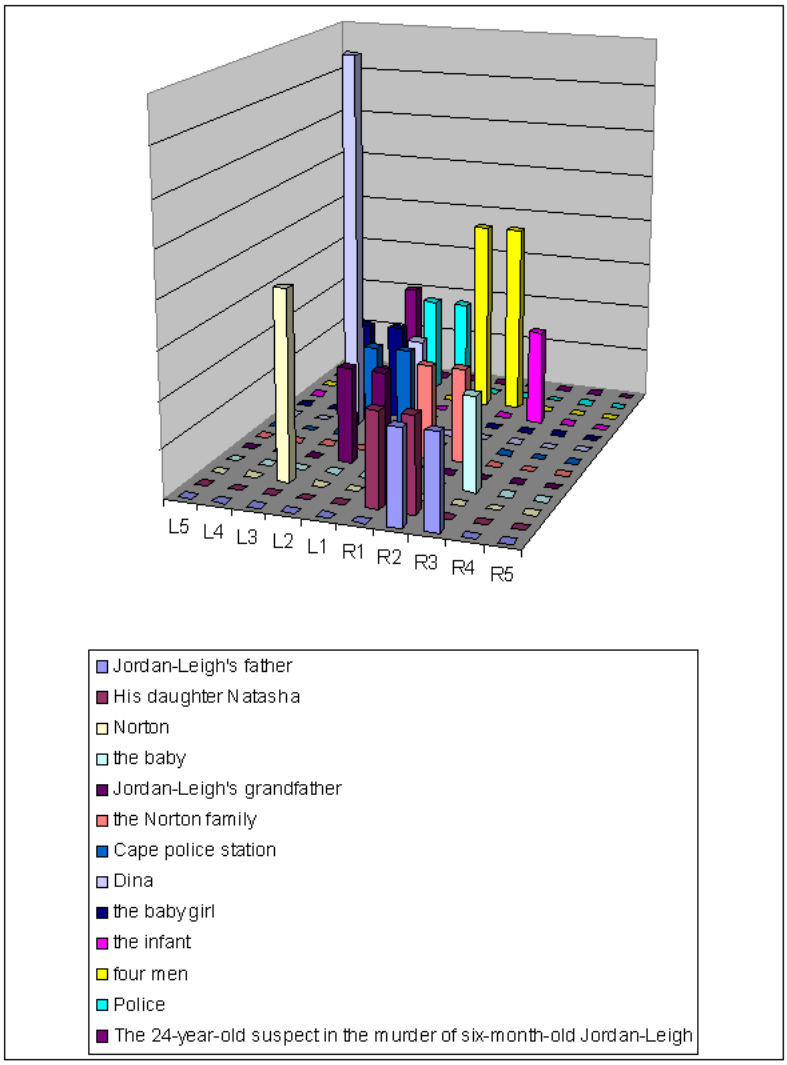

Figure 9: Scene graph (placed): Baby murder suspect

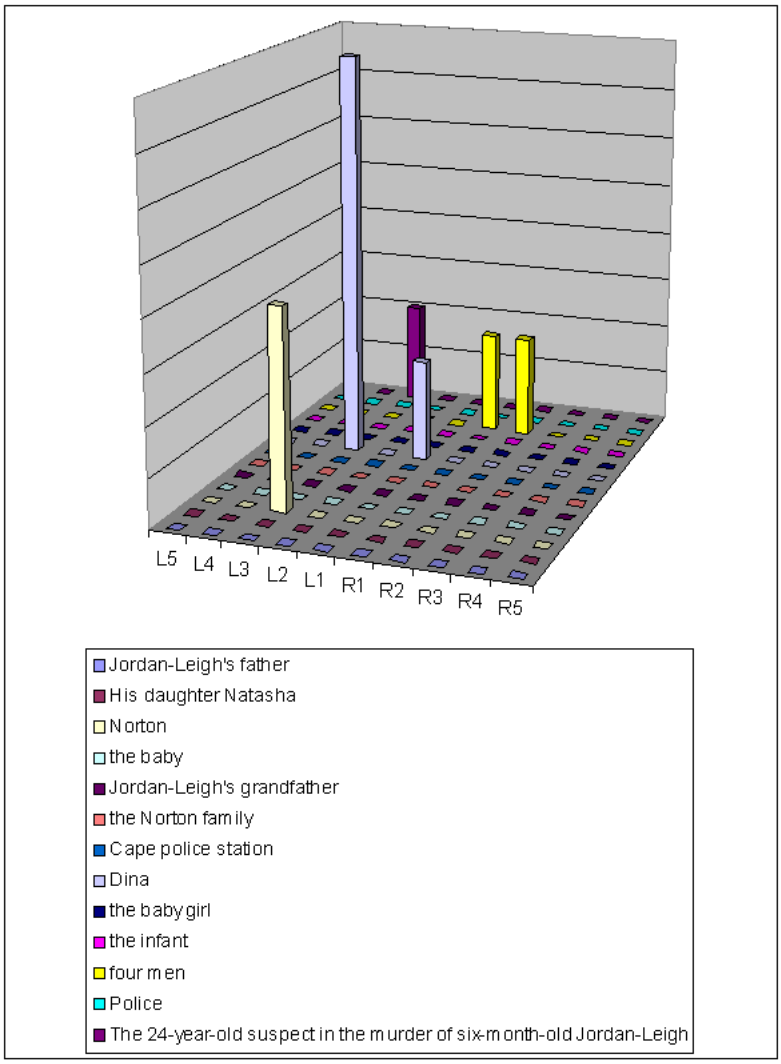

Figure 10: Scene graph (referenced): Baby murder suspect

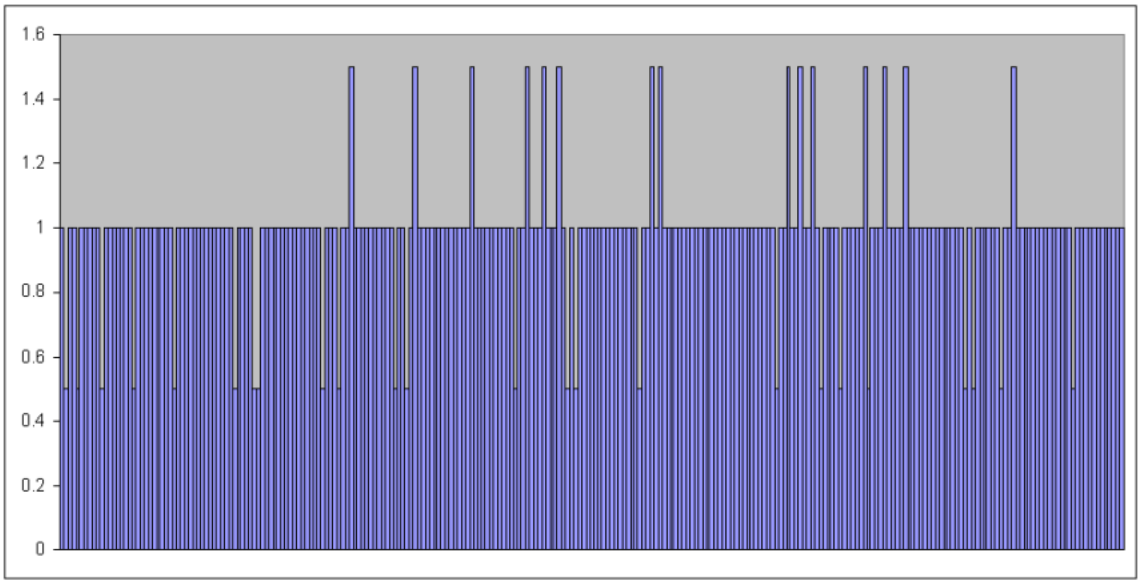

Figure 11: Form (speed): Coming of age in Karhide

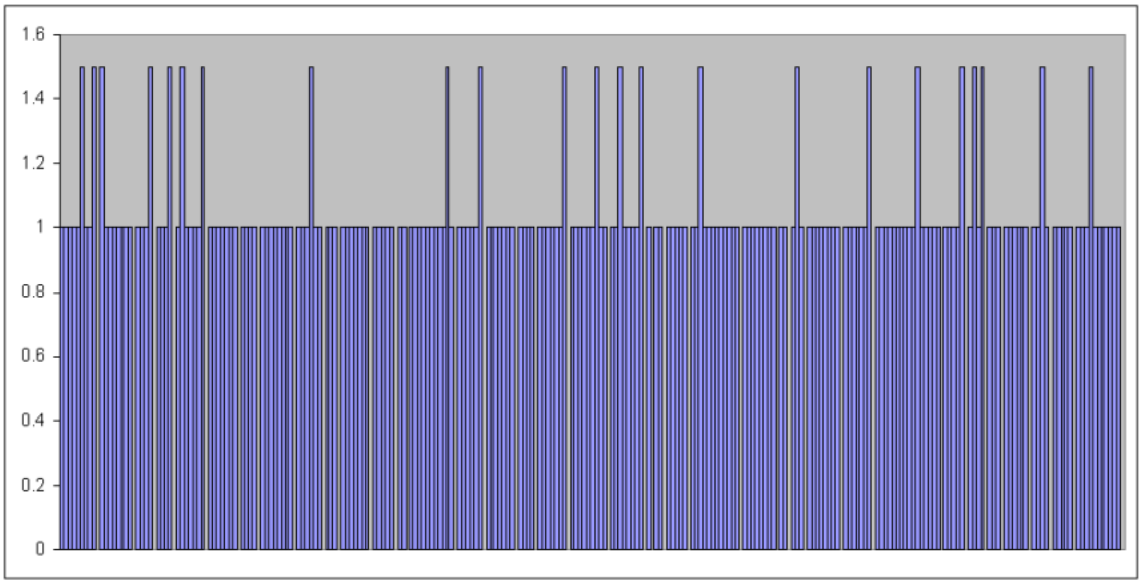

Figure 12: Form (space): Coming of age in Karhide

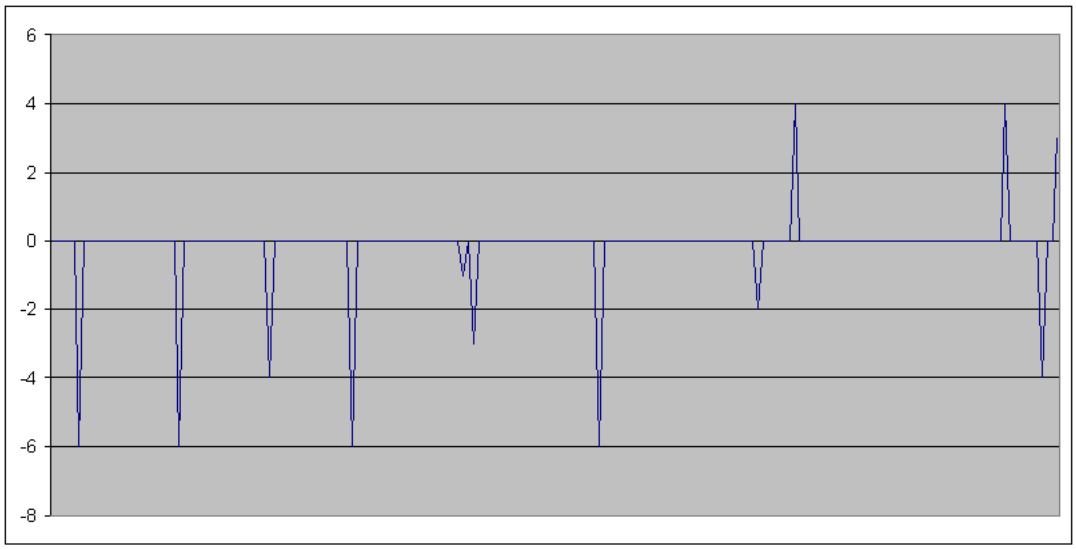

Figure 13: Class (emotional set 1): Baby murder suspect

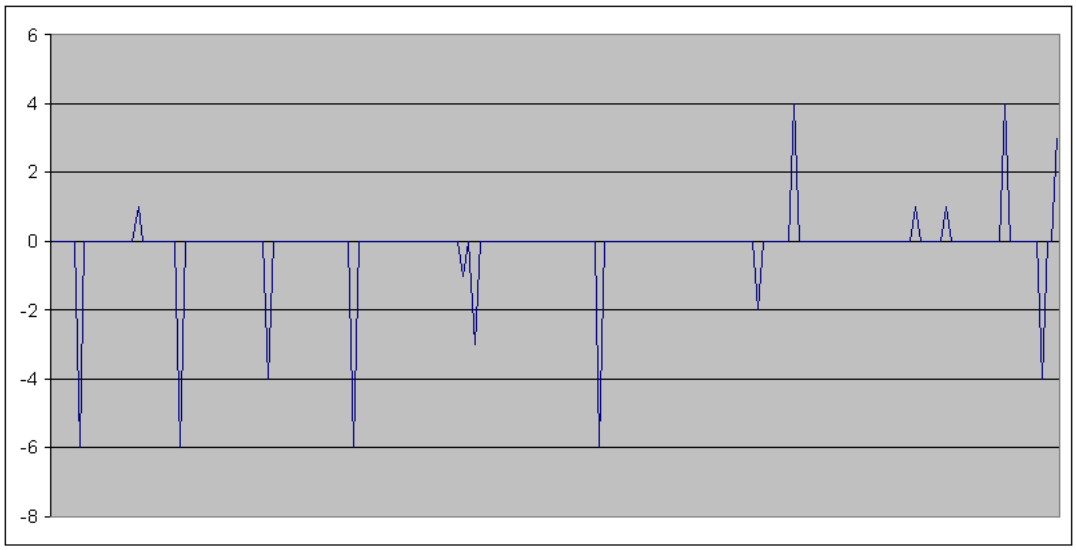

Figure 14: Class (emotional set 2): Baby murder suspect

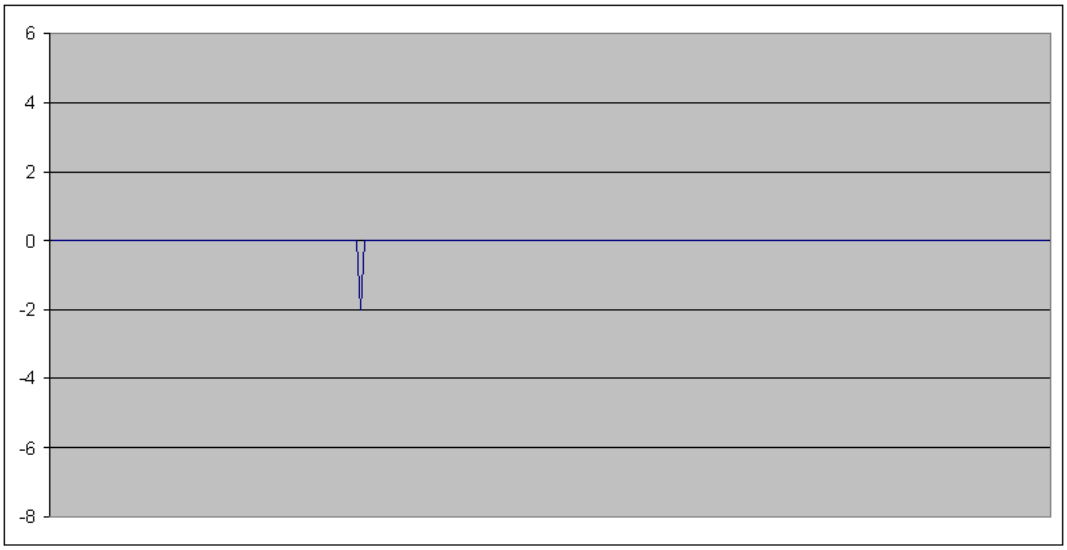

Figure 15: Class (emotional set 1): TAGs

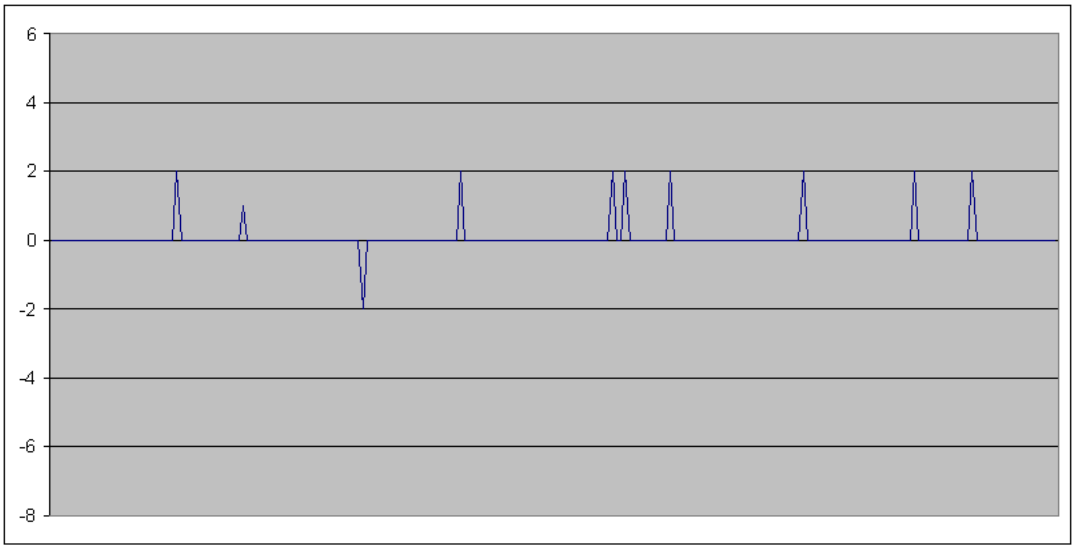

Figure 16: Class (emotional set 2): TAGs

Chapter 6

Conclusions and future work

In this chapter the conclusions from the work that has been done is discussed. Possible future work is also suggested.

6.1 Introduction

A successful design and implementation of a preprocessor for an English-to-Sign Language machine translation system was given that had to comply with specified constraints. These constraints were described in section 0.2 as:

- To find grammatical structure and part-of-speech information from text.
- To manage the spatial component of Sign Languages which will entail:
  - Resolving pronouns by finding their antecedents.
  - Managing the usage of the signing space.
- To generate prosodic information for the text which includes:
  - Body language.
  - Facial expressions.
  - Sign movement.
- To store the results in a universal and flexible format.

The remainder of this chapter will discuss the conclusions on these constraints, and future work. The chapter concludes with a summary.

6.2 Natural language processing

To find grammatical structure and part-of-speech information from text, the preprocessor used a Tree-adjoining grammar as prescribed by the SASL-MT project. The physical implementation was done by a package called *Lem14.0* [5]. This package was created at the University of Pennsylvania.

Due to the nature of the Lem package, it was impossible to automate this part of the preprocessing process with the rest of the preprocessor. The reason for this, was that for each sentence the parser parsed, all possible derivations were given. The number of derivations varied from a few to many thousands depending on the nature of the sentence. In other cases, the sentences could not be derived by the grammar, and therefore the parser yielded no derivations. This meant that sentences had to be parsed one at a time. If there were derivations, the best one would then be selected by hand, and if no derivations were produced, the sentence had to be modified, and parsed again. A future improvement of the preprocessor would be to address these issues.

The suggestion for future research on the TAG parsing process is to incorporate a statistical approach into the parser. To explain this, consider the word *bank* which could either be used as a verb or as a noun. When the parser encounters such a word, all derivations are calculated for the word as a verb, and as a noun. Assume for a moment that *bank* is used as a noun and is then substituted with another word *chair*. Because the word *chair* only belongs to one part-of-speech category, the number of derivations will be less. By statistically determining part-of-speech categories for words, results might be created better suited for real world applications. Building on this idea, it might further be possible to statistically extract concepts from sentences. For example, the probability of the words *grand slam* being used as a noun, and not as an adverb and a verb in a sentence, might be high, and by accepting that assumption could then help

reduce the number of derivations.

Once the sentences were parsed, and the appropriate derivations were selected, the results proved to be well suited for this type of natural language application.

6.3 Semantic relational databases

The semantic relational database proved to be a key component to implement the preprocessor. The database was used in every step of the preprocessing process after grammatical structure and part-of-speech information was found.

Modifying the WordNet database structure and adding the metadata storing capabilities for each sense of a word yielded a tool flexible enough to be used with diverse types of implementations; from a pronoun resolution algorithm to prosodic generation. The database proved powerful because a means was given to easily access relevant information.

For this work, the database was only partially populated with metadata. This was done because of the size of the task to fully populate such a database, and also because finding the correct metadata, especially for this implementation, is considered a research project on its own.

6.4 Pronoun resolution

The pronoun resolution algorithm yielded good results. The algorithm was constructed by using concepts from robust algorithms, centering algorithms and other algorithms.

As this algorithm inherited the good aspects of the different types of algorithms, it also suffered from the vulnerabilities the others suffer from. The algorithm is syntactically driven, which means it is sensitive to the grammatical structure and part-of-speech information generated by the English TAG. When the TAG derivations were inaccurate, the results of the pronoun resolution algorithm were

poor; when the derivation was a good description of the nature of the sentence, the results were good. Secondly, at times, syntax did not provide enough information to select an antecedent, and disambiguation rules had to be used. These rules still did not always resolve the ambiguity correctly. One way of improving disambiguation might be to consider using discourse analysis, which might also make the algorithm less sensitive to the grammatical structure found for the sentence. Lastly, the centering concepts created the situation where one mistake had the effect of causing more mistakes.

For future work, a bigger and more diverse corpus should be created to test and improve the pronoun resolution algorithm. The algorithm should possibly be modified to be less sensitive to the grammatical structure created by the TAG parser, and the disambiguation rules could be refined.

6.5 Scene graph

For this implementation, the scene graph algorithm was adequate. During analysis, the four dormant locations in the signing space were never used, proving that dividing the space in front of the signer into six locations was a fair assumption. Most of the objects placed in the signing space never occupied more than one location, and objects were spread more or less evenly across the signing space. The goal to make it as easy as possible for the observer to remember where each object was located was therefore achieved.

For future work, there are still improvements to be made to the scene graph algorithm. In the current implementation, the scene graph algorithm does not see *Mirza* and *teenager Mirza* as the same object, and will place them at different locations. This still needs to be addressed. Furthermore, showing relationships between objects, and performing role playing must also still be added to the algorithm.

6.6 Prosody generation

The proposed method to create prosodic information for text proved to be a flexible and good approach. It was shown that it is possible to extend the types of expressiveness information, in this case by enlarging the emotional class set. This meant expressiveness information could be created for different types of text, and that the results generated were well suited for that type of text.

For future work, two improvements must still be made. To explain the first improvement, it is noted that the semantic relational database distinguishes between different senses of a word. The metadata is also stored per word sense. Theoretically this means that if a word is used in one sense, the emotional class would be A for example, and if the word was used in another sense, the emotional class would be B. In the current approach the sense of the word that was used is not determined, and it is assumed that the expressiveness information is the same for all word senses.

Secondly, in the current implementation, the focus lies mostly on individual words and it is not taken into account the effect they might have on the words following them. For example, if a word should be spoken with a *happy* expression, and the next word is *neutral*, these sudden changes in expressiveness could seem unnatural. Future work would be to find a method where the expressiveness functions are not only determined by the words themselves, and the senses in which they are used, but also by the expressiveness of what happened in the sentence before. This will create a more natural and elegant flow of emotion and movement throughout the sentence.

6.7 Summary of conclusions

There are still improvements to be made for the preprocessor. However, the purpose of this project, to design and implement the foundation of such a system for the SASL-MT project, was fulfilled.

Bibliography

[1] http://www.cs.sun.ac.za/~lynette/SASL/.

[2] http://wordnet.princeton.edu/.

[3] http://engr.smu.edu/~rada/wnb/.

[4] http://tcts.fpms.ac.be/synthesis/introtts.html.

[5] http://www.cis.upenn.edu/~xtag/swrelease.html.

[6] http://wordnet.princeton.edu/obtain.

[7] Jean Aitchison. *Linguistics: an introduction*. Hodder and Stoughton, Great Britain, 1995.

[8] James Allen. *Natural Language Understanding*. The Benjamin/Cummings Publishing Company, United States of America, 1995.

[9] Ash Asudeh and Mary Dalrymple. Information-structural semantics for English intonation, 2004. http://www.ling.canterbury.ac.nz/personal/asudeh/pdf/asudeh-dalrymple-b%inding.pdf.

[10] Saliha Azzam, Kevin Humphreys, and Robert Gaizauskas. Evaluating a focus-based approach to anaphora resolution. In Christian Boitet and Pete Whitelock, editors, *Proceedings of the Thirty-Sixth Annual Meeting of the Association for Computational Linguistics and Seventeenth International Conference on Computational Linguistics*, pages 74–78, San Francisco, California, 1998. Morgan Kaufmann Publishers. http://www.citeseer.nj.nec.com/azzam98evaluating.html.

[11] Dean Barker. Computer Facial Animation for Sign Language Visualization. Master's thesis, University of Stellenbosch, Department of Computer Sciences, 2005.

[12] John Bear and Patty Price. Prosody, syntax and parsing, 1990. `http://portal.acm.org/citation.cfm?id=981826`.

[13] Ronald Cole, Joseph Mariani, Hans Uszkoreit, Annie Zaenen, and Victor Zue. *Survey of the state of the art in human language technology*. Cambridge University Press, Italy, 1997.

[14] Geoffrey R. Coulter. Emphatic stress in ASL. In *Theoretical Issues in Sign Language*, volume 1, pages 109 – 125. Springer, 1997.

[15] Laurance Danlos, Eric Laporte, and Francoise Emerand. Synthesis of spoken languages from semantic representation. `"http://www.citeseer.ist.psu.edu/588038.html"`.

[16] d'Armond L. Speers. *Representation of American Sign Language for Machine Translation*. PhD thesis, Georgetown University, Graduate School of Arts & Sciences, 2001.

[17] Lou Fant. *The American Sign Language phrase book*. McGraw-Hill Companies, 1994.

[18] Niyu Ge, John Hale, and Eugene Charniak. A statistical approach to anaphora resolution. `http://www.citeseer.nj.nec.com/ge98statistical.html`.

[19] Barbara J. Grosz. The representation and use of focus in a system for understanding dialogs. In *Procedings of the Fifth International joint conference on Artificial Intelligence*. Camebridge, 1977.

[20] Ursula K. Le Guin. *The birthday of the world and other stories*. Gollancz London, Great Britain, 2002.

[21] Laurie Hiyakumoto, Scott Prevost, and Justine Cassell. Semantic and discourse information for text-to-speech intonation, 1997. `http://www.citeseer.nj.nec.com/hiyakumoto97semantic.html`.

[22] Matthew P. Huenerfauth. A survey and critique of American Sign Language Natural Language Generation and Machine Translation Systems, 2003. `http://www.seas.upenn.edu/~matthewh/publications/huenerfauth-2003-ms-ci%s-03-32-asl-nlg-mt-survey.pdf`.

[23] Aravind K. Joshi and Yves Schabes. Tree-adjoining grammars. In *Handbook of Formal Languages*, volume 3, pages 69 – 124. Springer, 1997. `http://www.citeseer.nj.nec.com/joshi97treeadjoining.html`.

[24] Ronald M. Kaplan and Joan Bresnan. *Lexical-functional grammar: A formal system for grammatical representation.* The MIT Press, 1982.

[25] Ruslan Mitkov. Robust pronoun resolution with limited knowledge. In *COLING-ACL*, pages 869–875, 1998. `http://www.citeseer.nj.nec.com/mitkov98robust.html`.

[26] Ruslan Mitkov. Anaphora resolution: The state of the art, 1999. `http://www.citeseer.nj.nec.com/mitkov99anaphora.html`.

[27] Ruslan Mitkov. *The Oxford Handbook of Computational Linguistics*. Oxford University Press, United States of America, 2003.

[28] Richard T. Oehrle. Prosodic constraints on dynamic grammatical analysis, 1991. `http://www.citeseer.nj.nec.com/oehrle91prosodic.html`.

[29] Scott Allan Prevost. A semantics of contrast and information structure for specifying intonation in spoken language generation, 1995. `http://www.citeseer.nj.nec.com/226643.html`.

[30] Pritha Sarkar. Indias Sharapova reaches second round. Cape Times, Tuesday 21 June 2005.

[31] Pete Whitelock Sharp. What sort of trees do we speak? a computational model of the syntax-prosody interface in tokyo japanese. `"http://www.citeseer.ist.psu.edu/588083.html"`.

[32] Stuart Shieber and Yves Shabes. Synchronous tree adjoining grammars. In *Proceedings of the 13th International Conference on Computational Linguistics*, pages 253–258, 1990. `http://talana.linguist.jussien.fr/~lkalmey/LTAG-Semantics`.

[33] Candice L. Sidner. *Towards a computational theory of definite anaphora comprehension in English*. PhD thesis, Massachusetts Institute of Technology, Department of Electical Engineering & Computing Science, 1979.

[34] Mark Steedman. Structure and intonation in spoken language understanding. In *Meeting of the Association for Computational Linguistics*, pages 9–16, 1990. `http://www.citeseer.nj.nec.com/steedman90structure.html`.

[35] Mark Steedman. Information-structural semantics for English intonation, 2003. `http://www.ltg.ed.ac.uk/magicster/deliverables/steedman3.pdf`.

[36] Danielle Steel. *Five days in Paris*. Bantam Press, 1995.

[37] Michael Strube. Never look back: An alternative to centering. In *COLING-ACL*, pages 1251–1257, 1998. `http://www.citeseer.nj.nec.com/strube98never.html`.

[38] Roland Stuckardt. Anaphor resolution and the scope of syntactic constraints. `http://www.citeseer.nj.nec.com/article/stuckardt96anaphor.html`.

[39] Nina Suszczanska, Przemyslaw Szmal, and Jaroslaw Francik. Translating Polish texts into Sign Language in the TGT system. `"http://www.citeseer.ist.psu.edu/suszczanska02translating.html"`.

[40] Lynette van Zijl and Dean Barker. A Machine Translation System for South African Sign Language. 2003. `http://www.cs.sun.ac.za/~lynette/publications/SASLGraphics.ps`.

[41] Anand Venkataraman, Luciana Ferrer, Andreas Stolcke, and Elizabeth Shriberg. Training a prosody-based dialog act tagger from unlabeled data. In *Proc. IEEE International Conference on Acoustics, Speech and Signal*

Processing, volume 1, pages 272–275. Springer, 2003. http://www.speech.sri.com/people/stolcke/publications.html.

[42] Marilyn A. Walker. Centering, anaphora resolution, and discourse structure, 1997. http://www.citeseer.nj.nec.com/93789.html.

[43] Bruce A. Wooley. Pronoun resolution of "they" and "them". http://www.citeseer.nj.nec.com/309465.html.

[44] Staff Writer. Baby murder suspect, 24, in custody. Cape Times, Tuesday 21 June 2005.

[45] Liwei Zhao, Karin Kipper, William Schuler, Christian Vogler, Norman Badler, and Martha Palmer. A Machine Translation System from English to American Sign Language. http://www.citeseer.ist.psu.edu/539314.html.

Appendix A

Tree-adjoining grammars (TAGs)

The definitions in this appendix are summarized from the work given in [23]. A Tree-adjoining grammar (TAG) is the sum of five elements (Σ, NT, I, A, S).

- Σ is a finite set of terminating symbols.
- NT is a finite set of non-terminal symbols so that $\Sigma \cap NT = \emptyset$.
- I is a finite set of finite trees, called *initial trees*.
- A is a finite set of finite trees, called *auxiliary trees*.
- S is a special non-terminal symbol with $S \in NT$

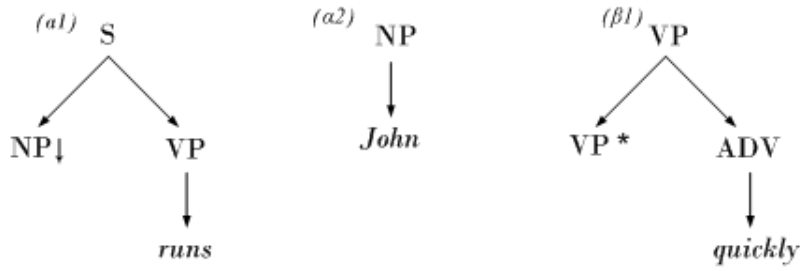

Figure 17: A Tree-adjoining grammar

In Figure 17 a TAG is given with $\Sigma = \{John,\ runs,\ quickly\}$ and $NT = \{S,\ NP,\ VP,\ ADV\}$.

The set I of initial trees have non-terminal symbols on all internal nodes, and non-terminal or terminal symbols on the leafs of the trees. The non-terminal symbols on the leafs can be substituted by other trees, and are marked for substitution by ($\downarrow$). From Figure 17 on page 80 the set $I = \{\alpha 1,\ \alpha 2\}$ can be found.

The set A of auxiliary trees have non-terminal symbols on all internal nodes, and non-terminal or terminal symbols on the leafs of the trees. The non-terminal symbols on the leafs can be substituted by other trees, except for one node, called the *foot node*. The foot node is marked by an asterisk ($*$), and its label must be the same as the label of the root node. From Figure 17 the set $A = \{\beta 1\}$ can be found.

The elements of the set $I \cup A$ forms a set of trees called the *elementary trees*. Any tree that was constructed from two or more trees by means of some operation is called a *derived tree*.

A.1 Tree-adjoining grammar operations

Adjoining and *substitution* are two operations that can be used to create a derived tree.

A.1.1 Adjoining

A derived tree can be constructed by *adjoining* an auxiliary tree α with another tree β. The tree β could be auxiliary, initial, or derived.

Assume β has a node X, and α has as its root X. To create γ by adjoining (See Figure 18 on page 82):

1. The sub tree of β with X as root is removed, with a copy of node X left behind. The spliced tree is called t.
2. α is now attached where the copy of X was left.

3. t is attached to the foot node of α.

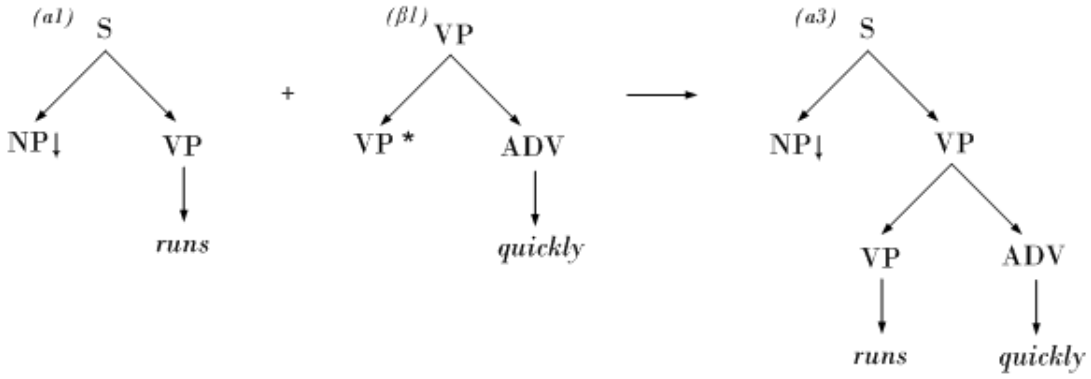

Figure 18: TAG adjoining operation

A.1.2 Substitution

A derived tree can be constructed by *substitution* of a tree α, with a node from a tree β. Substitutions can only take place on leafs marked for substitution.

Assume α has a node X marked for substitution, and a tree β has a root node X. To create γ by substitution (See Figure 19):

1. Replace the node X of α by β.

Only trees derived from initial trees can be used for substitution.

A.1.3 Constraints

Constraints can be defined saying which auxiliary trees may be adjoined at which nodes of other trees to form derived trees.

One of three constraints can be specified: *Selective adjunction* means that only auxiliary trees from a set T, where $T \subset A$, can be used to be adjoined to a specific

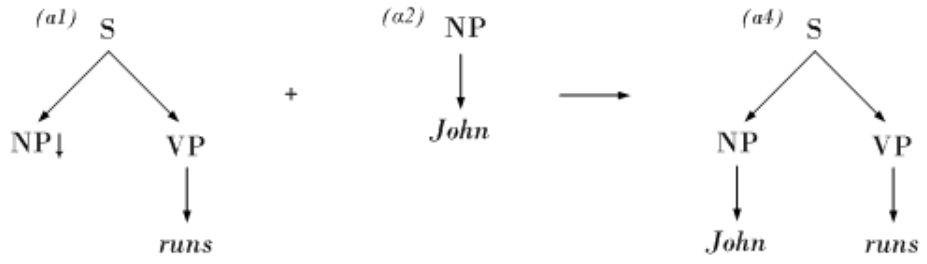

Figure 19: TAG substitution operation

node n. Adjunction on that node is not mandatory. *Null adjunction* means that the set T is empty, or $T = \emptyset$. *Obligatory adjunction* means that adjunction is mandatory at node n, with an auxiliary tree from the set T.

A.2 Derivation tree

The operations of adjoining and substitution is used for derivation in TAGs. It cannot be seen from a derived tree what operations were used to construct it. However, an object called a *derivation tree*, uniquely describes how the derivation tree was created. In a derivation tree, solid lines are used to show adjunction, and dashed lines are used to show substitution. In Figure 20 on page 84 the derived tree can be found for the sentence *John runs quickly*, and next to it is the derivation tree for the same sentence. The derivation tree should be interpreted as follows: Node 1 of tree $\alpha 1$ was substituted by tree $\alpha 2$, and then tree $\beta 1$ was adjoined to tree $\alpha 1$ at node 2.

A.3 Toy languages

In the following sections examples are used to clarify some of the properties of TAGs, and highlight their usefulness. For this purpose a small grammar called *Toy English* has been created (see Figure 21 on page 84).

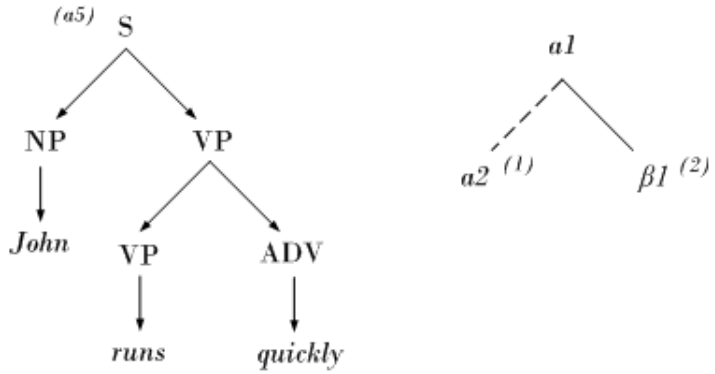

Figure 20: Derived tree and derivation tree for *John runs quickly.*

Some attributes of TAGs can also better be seen by looking at the shortcomings of other representations such as a *context free grammar* (CFG). The CFG for the toy English is given in Figure 21 on page 84.

(1)	S	⟶	NP	VP
(2)	VP	⟶	V	VP
(3)	VP	⟶	VP	ADV
(4)	NP	⟶	*John*	
(5)	NP	⟶	*home*	
(6)	V	⟶	*arrives*	
(7)	ADV	⟶	*safely*	

Figure 21: CFG of Toy English

A.4 Relevance of Tree-adjoining grammars

TAGs are different from other grammars because it uses tree objects to represent grammatical structure. As it will be shown, the properties of these tree objects

make it advantageous to use TAGs with natural language to describe the structure of a sentence.

TAGs have two main properties. The first one is called *extended domain of locality* (EDL), and the other is called *factoring recursion from the domain of dependencies* (FRD). All other properties of TAGs are derived from these two basic properties [23].

A.4.1 Extended domain of locality

Example 6 *Using the CFG given in Figure 21 on page 84, the sentence* John arrives home *is derived:*

$$\begin{aligned} S &\Rightarrow NP\ \ VP \\ &\Rightarrow NP\ \ V\ \ NP \\ &\Rightarrow John\ \ V\ \ NP \\ &\Rightarrow John\ \ arrives\ \ NP \\ &\Rightarrow John\ \ arrives\ \ home \end{aligned}$$

To derive the sentence *John arrives home*, begin with an S, and use the first two rules of the CFG: $S \rightarrow NP\ \ VP$, and $VP \rightarrow V\ \ NP$. This will gives an $S \rightarrow NP\ \ V\ \ NP$. Considering the given CFG, a person might want to simplify the grammar by replacing the first two rules with an immediate description of a sentence as $S \rightarrow NP\ \ V\ \ NP$. However, it is impossible to make such a substitution without changing the set of possible sentences that this grammar can derive. To prove this, consider another sentence that is derived from the original CFG.

Example 7 *Using the CFG given in Figure 21 on page 84, the sentence* John

arrives home safely *is derived:*

$$\begin{aligned} S &\Rightarrow NP\ \ VP \\ &\Rightarrow NP\ \ VP\ \ ADV \\ &\Rightarrow NP\ \ V\ \ NP\ \ ADV \\ &\Rightarrow John\ \ V\ \ NP\ \ ADV \\ &\Rightarrow John\ \ arrives\ \ NP\ \ ADV \\ &\Rightarrow John\ \ arrives\ \ home\ \ ADV \\ &\Rightarrow John\ \ arrives\ \ home\ \ safely \end{aligned}$$

Assume the two rules $S \rightarrow NP\ \ VP$, and $VP \rightarrow V\ \ NP$ were exchanged for a more compact $S \rightarrow NP\ \ V\ \ NP$ in the grammar. In other words, the CFG now starts with $S \rightarrow NP\ V\ NP$. The new CFG can be seen in Figure 22 on page 86.

(1)	S	⟶	NP	V	NP
(2)	NP	⟶	*John*		
(3)	NP	⟶	*home*		
(4)	V	⟶	*arrives*		
(5)	VP	⟶	V	VP	
(6)	VP	⟶	VP	ADV	
(7)	ADV	⟶	*safely*		

Figure 22: New CFG of Toy English

Closer inspection of the new CFG show that the sentence *John arrives home safely* cannot be derived. This is because of the rule $S \rightarrow NP\ V\ NP$, which means there is no way to derive from S a sentence that contains a VP. The result is that the rule $VP \rightarrow VP\ ADV$, and consequently $ADV \rightarrow$ *arrives*, is lost in the grammar. Therefore, it is not possible to simplify the original CFG by describing the relationship of a NP then V followed by another NP as one rule without abandoning the VP in the grammar.

If the rules of a CFG are considered as its domain of locality, then in this case, the domain cannot contain the rules of $S \rightarrow NP\ V\ NP$ and $VP \rightarrow VP\ ADV$ at the same time for the given CFG. This is however possible for a TAG.

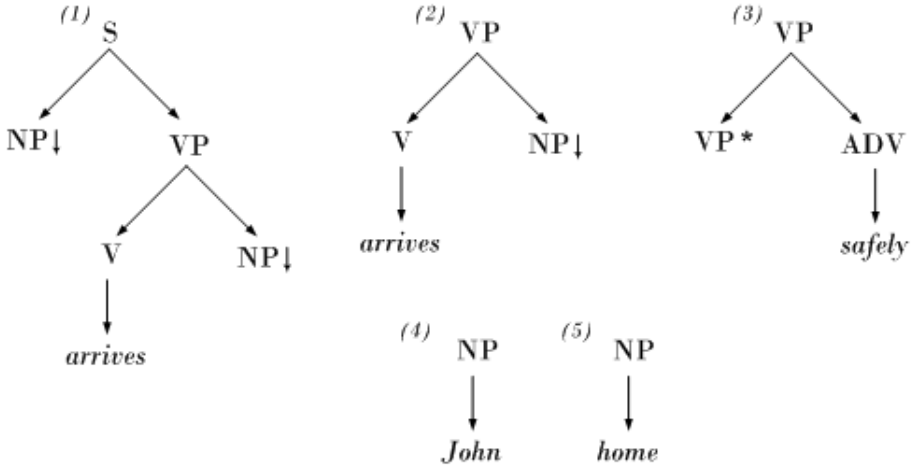

Figure 23: TAG of Toy English

Figure 23 on page 87 shows that with a TAG, the Toy English can be represented by five grammar rules, where the original CGF needed seven. The sentence *John arrives home* can be derived by using trees 1, 4, and 5, and using the *substitution* operation. The sentence *John arrives home safely* can be derived by first *adjoining* tree 1 and 3, and then substituting tree 4 and 5 into the result.

The example shows that if for a TAG the tree objects are considered as its domain of locality, then TAGs have an extended domain above that of CFGs. This is because the relationship of a NP, V, NP can be incorporated into one tree object without abandoning the VP, and therefore the ADV, from the grammar.

A.4.2 Factoring recursion from domain of dependencies

For a CFG, the rules of the grammar show dependencies between the elements of those rules.

Example 8 *Looking at the CFG rule $S \rightarrow NP\ VP$, the NP and the VP share a dependency.*

The dependency given in Example 8 on page 88, shows that for the grammar rule $S \rightarrow NP\ VP$, the NP will always be followed by a VP. Note that this dependency can be lost, or broken.

Consider the CFG:

1. S → aa | ab
2. a → aa | ab

Investigating the derived sentence *abab*, it is noted that the sentence could be derived either from

$$\begin{aligned} S &\Rightarrow aa \\ &\Rightarrow aba \\ &\Rightarrow abab \end{aligned}$$

$$\begin{aligned} S &\Rightarrow ab \\ &\Rightarrow aab \\ &\Rightarrow abab \end{aligned}$$

It is impossible to determine which method was used by looking at the derived sentence alone. In other words, there is no way to be sure whether the sentence was derived from a rule stating that an a is dependent on another a following, or a rule stating that an a is dependent on a b following.

The tree objects of TAGs are also used to show dependencies.

In Figure 24, one of the dependencies in the tree exists between the $NP(wh)$ and the verb *arrives*. Similar to CFGs, the dependencies are defined automatically by defining the structure of the rule. The tree in the example describes a sentence S as a $NP(wh)$ followed by another S. This S will consist of a NP followed by

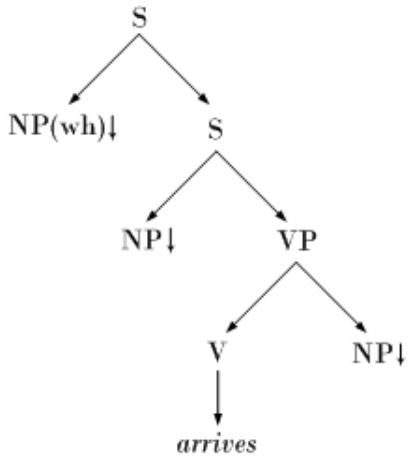

Figure 24: A TAG tree object.

a VP, and the VP will consist of the verb *arrives* and another NP. Due to the operation of adjoining, it might seem that these dependencies are lost.

In Figure 25, the verb *arrives* that was associated as being at a specific location in the tree from the $NP(wh)$, has now moved to a position further away from the $NP(wh)$ due to adjoining. However, by factoring out recursive information from one tree into separate trees, it can be seen that these dependencies are kept together (see Figure 26 on page 91).

A.4.3 Mildly context sensitive grammars

TAGs fall into the class of *mildly context sensitive grammars*. This context sensitivity is a direct result of the tree objects and the properties of EDL and FRD [23].

To see why these properties make it advantageous to describe the grammatical structure of natural language, consider the sentence:

When does Jack say John arrives home?

Without punctuation to help clarify the meaning, the question being asked could either be when John arrives, or when Jack says. d'Armond Speers [16]

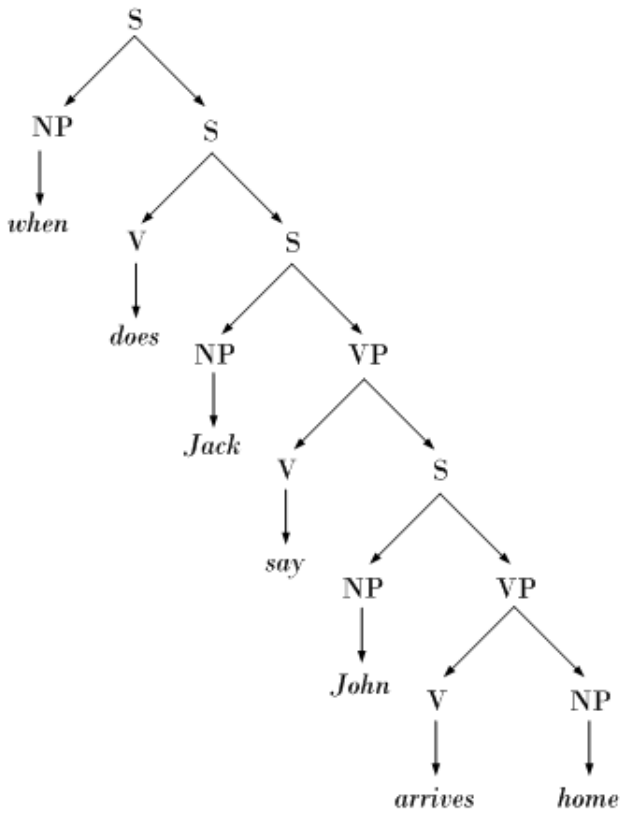

Figure 25: A modified TAG tree object.

identifies one of the problem areas of machine translation as *syntactic ambiguity*. This happens when, as can be seen from the example, a sentence could have different meanings depending on how the syntax is viewed. Speers argues that syntactic ambiguity should be resolved by a syntactic parser. If a CFG is used to describe the above sentence, it would not be possible to disambiguate the sentence due to the fact that CFGs are context free. It is similar to the problem with the example of finding the starting rule for *abab*, not knowing whether the starting rule was $S \Rightarrow aa$, or $S \Rightarrow ab$. TAGs on the other hand are mildly context sensitive and dependencies are not lost. Assume the sentence is the one derived by the TAG in example, then it would be possible to determine that the dependency was between $NP(wh)$ and the verb *arrives*, and therefore, it is known that the question is, when does John arrive.

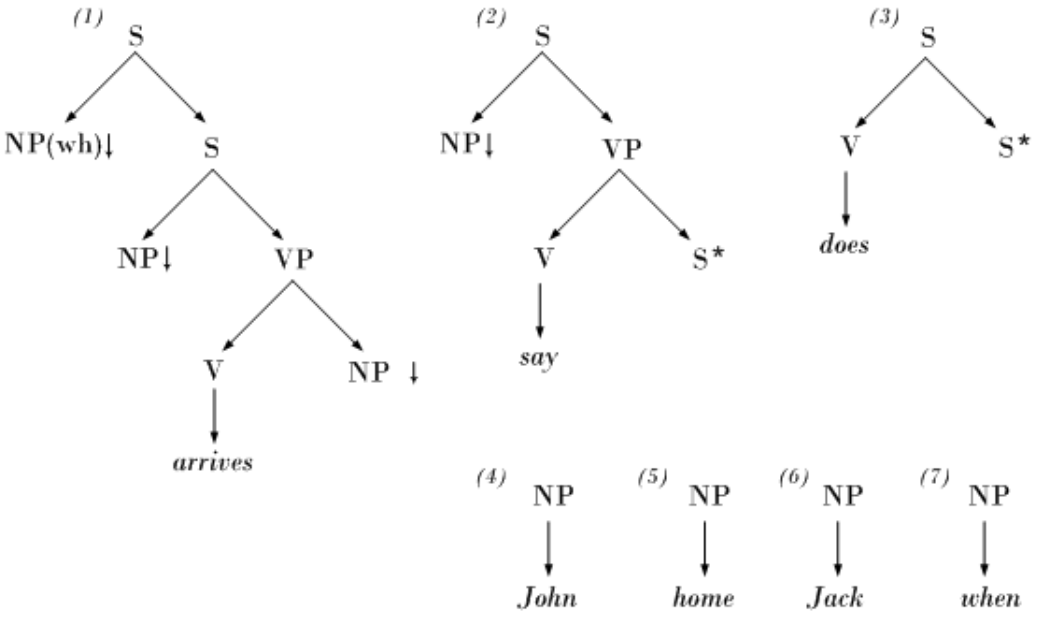

Figure 26: Tree objects after factoring out recursive information.

Appendix B

Anaphora

Following are some definitions regarding anaphora, for more information see [38].

Definition 1 (Anaphora) *Anaphora is the term used to describe the process in which one object is pointing back to another object. The pointing back is done to make a reference, or imply the first object.*

Definition 2 (Anaphor) *An anaphor is the object that is making the reference to the first object.*

Definition 3 (Antecedent) *An antecedent is the object that is being referenced by the anaphor.*

Example 9 *Consider the following sentences:*

Jack is buying ice-cream. He also loves chocolate.

In these sentences, anaphora is found between the word Jack*, and the word* He*. In this instance the word* He *is the anaphor, and the word* Jack *is the antecedent.*

Definition 4 (Anaphoric expression) *An anaphor together with its antecedent is called an anaphoric expression.*

Definition 5 (Intrasentential anaphors) *An anaphor is intrasentential, and called an intrasentential anaphor if its antecedent is found in the same sentence. In other words, if the anaphoric expression is found in one sentence.*

Definition 6 (Intersentential anaphors) *An anaphor is intersentential, and called an intersentential anaphor if its antecedent is found in a different sentence. In other words, if the anaphoric expression is found in more than one sentence.*

Example 10 Frank and Rose were running when he fell. They are brother and sister.

In these sentences two anaphoric expressions are found. The first one is the words Frank*, and* he*, and because they are found in the same sentence,* he *is an intrasentential anaphor. The second one is the words* Frank and Rose*, and* They*, and because they are found in different sentences, the word* they *is an intersentential anaphor.*

Definition 7 (Anaphora resolution) *Anaphora resolution is the process of determining antecedents from anaphors, or resolving an anaphoric expression. In other words, if an anaphor is found, to determine the correct antecedent for that anaphor.*

Example 11 John and Jill plays tennis. He won the last time they played.

In these sentences, anaphora resolution must be performed on the word He *to determine who won the previous match. By using a constraint such as gender agreement, implying that the gender of the antecedent and the anaphor must be the same, it is known that* He *is showing back to* John*, and not* Jill.

Definition 8 (Referent) *Both the anaphor and its antecedent are in reality pointing to the same entity. This shared entity is called the referent.*

Definition 9 (Coreferential) *Because of the fact that an anaphor and its antecedent share the same referent, together they are called coreferential.*

Example 12 James is walking to the shop. He does not own a car.

In these sentences, the words James*, and* He *are pointing to the same real world entity of the person James, who is the referent. Because the words* James*, and* He *share the same referent, they are coreferential.*

B.1 Types of anaphora

There are different types of anaphora. The most common occurrence of anaphora is *pronominal anaphora*. Pronominal anaphora is when the anaphor consists of a pronoun.

Example 13 Sam was very tired when he got back from school.

In this sentence pronominal anaphora is found were the anaphor he *is a pronoun pointing to the antecedent* Sam.

The word *it* is in most cases not an anaphor. This is because the word *it* is not pointing back to anything. When this happens, the *it* is called *pleonastic*.

Example 14 It is raining.

In this sentence, it *is not pointing back to anything, and is therefore pleonastic.*

Another type of anaphora is *definite noun phrase anaphora*. This type of anaphora is found when the antecedent is referred to by an anaphor consisting of a definite noun phrase.

Example 15 Although Sam was tired, the eager student started with his homework.

In this sentence definite noun phrase anaphora is found were the anaphor the eager student *is a definite noun phrase pointing to the antecedent* Sam.

A third type of anaphora is *One-anaphora* where the word *one* is used to point back to an antecedent.

Example 16 *Look at the sentence:*

From all the colours, which one is your favorite?

In this sentence one-anaphora is found were the anaphor one *is referring to a colour.*

Appendix C

Prosody

Prosody for spoken languages is the non-lexical information contained within an utterance to convey meaning through the use of *sentence-stress*, or in other words, the accenting of words. In spoken language, this includes changes to the vocal pitch of words, and the rhythm in which the utterance is spoken.

C.1 Intonation

A sentence can be broken up into *prosodic phrases*, where boundaries are usually indicated by a change in rhythm. A sentence can be broken up into prosodic phrases in different ways. Every sentence contains at least one prosodic phrase.

Each prosodic phrase contains a *nuclear accent*. The nuclear accent is the last accented word in the prosodic phrase.

Every prosodic phrase has a *tune*. The tune is the rise and fall in vocal pitch as the utterance progress through the prosodic phrase. A tune consists of *pitch-accents* and *boundary tones*, which are discussed in the following section.

Intonation in spoken language is when prosodic phrases are spoken with a tune and rhythm. This means that in a prosodic phrase, there will be one pitch-accent associated with each accented word and one boundary tone associated with the end of each prosodic phrase.

C.2 Pitch-accents

A pitch-accent is associated with each accented word in a prosodic phrase. Three different pitch-accents are defined.

The H* pitch-accent has its pitch peaking on, or near the primary stressed vowel of the accented word. When such a pitch-accent is found, the preceding consonant will be voiced, for example in the word *lit*.

The L+H* pitch-accent is defined as similar to the H* pitch accent, however, there will be a longer rise to peak than in the usual case.

The L* pitch-accent has its pitch going through, on, or near the primary stressed vowel of the accented word.

C.3 Boundary tones

A boundary tone influence the pitch contour between the nuclear accented word and the prosodic phrase boundary to the right. Four different types of boundary tones are defined.

The L-L% boundary tone indicates that the tune of the prosodic phrase ends in a low pitch. If the nuclear accented word is accented with a high pitch, there will firstly be a fall in pitch whereafter the pitch will keep low until the end of the phrase. These boundary tones are found in neutral statements.

The H-H% boundary tone indicates that the tune of the prosodic phrase ends in a high pitch. These boundary tones are found in *yes-no* questions.

The L-H% boundary tone indicates that the tune of the prosodic phrase starts with a low pitch after the nuclear accented word, and then continually rise to the end of the phrase. These boundary tones are found in sentences with contradiction, if there is doubt or uncertainty, or if there is continuation of information into the next phrase.

The H-L% boundary tone indicates that the tune of the prosodic phrase starts with a high pitch after the nuclear accented word, and then continually fall to the end of the phrase. These boundary tones are found where a list is recited, but in a disinterested manner.

Appendix D

WordNet

WordNet [2] is an English semantic relational database where words are organized around semantic relationships. These relationships are based on the way in which humans understand and use these words. For example, humans know that a *table* is considered *furniture*, therefore a semantic relationship is placed between those words.

Wordnet consists of words from four part-of-speech categories: nouns, verbs, adjectives, and adverbs. Words are grouped in *synsets*, which is a subset of the dictionary, containing semantic relationships. For example, the word *dog*, and the word *canine* is in the same synset. If a word has more than one meaning, that word will belong to more that one synset, one for each meaning.

D.1 Semantic relationships

The following table shows which part-of-speech category is grouped according to which semantic relationships.

Noun	**Verb**	**Adjective**	**Adverb**
hypernym	hypernym		
hyponym	troponym		
holonym			
meronym			
	entails		
	causes		
antonym	antonym	antonym	antonym
		similar	
	see also	see also	
derivation	derivation		
domain/ classification	domain/ classification		
domain term/class	domain term/class		
other	attribute	attribute	derived from adjective

D.1.1 Holonym

A word is a *holonym* if another word is a part, a substance, or a member of that word. For example, a *tree* is a holonym because it contains parts such as a *leaf*, or a *branch*.

D.1.2 Meronym

A word is a *meronym* if it is a part, a substance, or a member of a holonym. For example, a *leaf* is a meronym because it is a part of a *tree*.

D.1.3 Hypernym

A word is a *hypernym* if it is a more general word than that of a word with a similar but more specific meaning. For example, a *tree* is a hypernym because an *oak* is a kind of tree.

D.1.4 Hyponym

A word is a *hyponym* if it is a more specific word than a word with similar meaning, but which is more general. For example, an *oak* is a hyponym because an oak is a kind of *tree*.

D.1.5 Troponym

A word is a *troponym* if it describes a way in which something is being done. For example, *amble* is a troponym because to amble is to *walk* in some manner.

D.1.6 Antonym

A word is an *antonym* of another word, if it means the opposite of the other word. For example, the word *slow* means the opposite of the word *fast*.

D.1.7 Synonym

If two words can be swapped in a sentence without changing the meaning of the sentence, then those words are *synonyms*. For example, the words *arrange*, and *order* are synonyms in certain context.

D.1.8 Entailment

One verb *entails* another verb, if the first one cannot be done without doing the other. For example, a person cannot *give* anything if that person does not *have*

anything to give.

D.1.9 Cause

One verb *causes* another verb, if the first verb can cause the second one to happen. For example, if a person *has* something, then that person can *give* something.

D.1.10 Attribute

A noun is an *attribute* if it gives a value for an adjective. For example, the noun *weight* is an attribute, that gives a value to the adjective *heavy*.

D.1.11 Pertainym

An adjective is a *pertainym* if it has to do with something else, and does not have an antonym. For example, the word *dental* is a pertainym, and refers to something that has to do with teeth.

Appendix E

English-to-Sign Language machine translation

In this appendix methods are considered for translating English into Sign Language, and discuss three different approaches that have been put forward to achieve this goal. Problem areas concerning *machine translation* (MT) in general is also discussed.

E.1 Introduction

If a sentence is spoken in one language, then it might be important to know how it could be translated into another language.

From this point onward when MT is discussed, the term *source language* will refer to the input language and *target language* will refer to the output language.

MT is the process of translation, by a machine, from a source language to a target language. The different approaches to MT is summarized by the MT pyramid [16] depicted in Figure 27 on page 103. At the bottom of the pyramid the approach of direct translation is found. In this approach there exists for each grammar rule of the source language, an equivalent grammar rule for the target language. A sentence in the source language is analyzed and classified, and then the equivalent sentence in the target language is generated while the words from

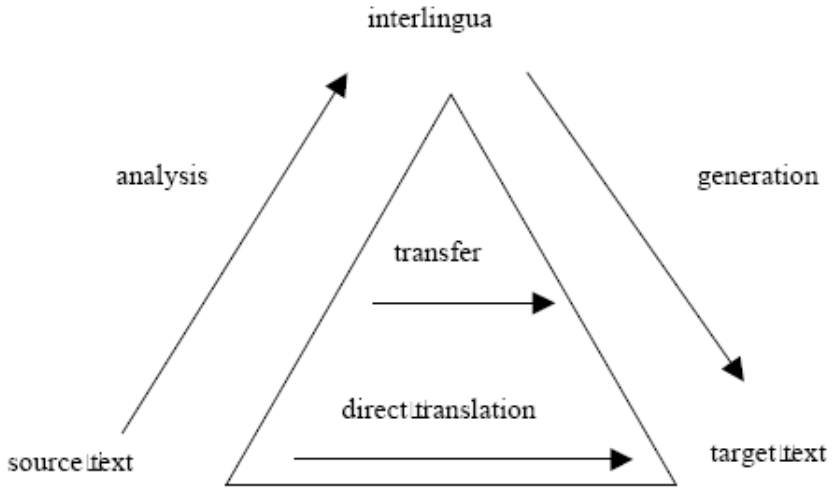

Figure 27: Translation pyramid

the source language is replaced by words from the target language. This approach focus mainly on syntax.

Example 17 *Assume a source language A exists with grammar rule $S \rightarrow \alpha\beta$, and a target language B with the grammar rule $S \rightarrow \gamma\delta$. Using direct translation, if a sentence $\alpha\beta$ was found in the source language, then the sentence can be mapped to the target language as $\gamma\delta$.*

At the top of the pyramid, an approach is found where the source text is analyzed, and from the analysis an intermediate representation is produced. This *interlingua* will be a structure capturing the essence of the meaning of the sentence, but has no relation to the original syntax of the input sentence. Finally the output text will be generated from the intermediate representation. This approach focus on semantics.

Example 18 *Assume a source language A exists with grammar rule $S \rightarrow \alpha\beta$, and a target language B with the grammar rule $S \rightarrow \gamma\delta$. Using semantic analysis of the sentence $\alpha\beta$, the meaning of the sentence is identified as say ϵ. The meaning ϵ can now be used to generate the target language sentence as $\gamma\delta$.*

In between these two methods approaches are found that use a combination of analysis, intermediate representation, and the use of syntax to do MT. There is a *transfer* from the one language to the other.

Example 19 *Assume a source language A exists with grammar rule $S \rightarrow \alpha\beta$, and a target language B with the grammar rule $S \rightarrow \gamma\delta$. Using a transfer method, the sentence $\alpha\beta$ can be transformed into an intermediate representation of $\Lambda\Theta$. This intermediate representation of the target language is used to create an intermediate representation of the source language $\Gamma\Delta$. Finally, the target language sentence as $\gamma\delta$ is created from the target language intermediate representation.*

E.2 Related work

In this section an overview is given of other projects concerned with the MT of a spoken language in written form to a Sign Language.

E.2.1 Written Polish to Polish Sign Language

At the Silesian University of Technology, a prototype of a system was developed to translate a written Polish document into its *Polish Sign Language* (PSL) equivalent. The system is named TGT-1, which stands for Text-into-Gesture Translator [39].

An overview of their system is as follows: A sentence from the source language is analyzed to identify what is called *syntactic groups*. After analysis, the sentence is then represented by a *syntactic graph*, where a node in the graph is a syntactic group. The arcs between nodes are viewed as syntactic relationships between syntactic groups. The words in each syntactic group could be found anywhere in the original sentence, and the syntactic graph therefore bears little or no resemblance to the original syntactic structure.

After the syntactic graph is completed, the system constructs a *predicative graph*. This predicative graph is an extension of the syntactic graph with the structure of the syntactic graph preserved, but were predefined roles are assigned

to every node. In the original model, sixteen roles where defined, but only three are implemented in TGT-1. The roles are *Action*, *Agent*, and *Object*.

Once the predicative graph is completed and the roles are assigned, a mapping is done from the predicative graph to the target language. The *Action* becomes the predicate, the *Agent* becomes the subject, and the *Object* becomes an adjunct of the target language.

In this specific case, it is claimed that the system yields good results for a definite class of sentences and compound sentences. The authors indicate that, to improve their system, the number of roles must be increased in the predicative stage.

Because written Polish and Polish Sign Language share a significant number of similarities, and because Polish has a free syntax, this is a good approach for such an MT system. On the other hand, the syntax of English and SASL are more strict, and therefore such a radical approach is not needed to attempt MT from English to SASL. Furthermore, a *syntactic group grammar* needs to be constructed to find the syntactic groups in the initial sentences, which can be a linguistically time consuming task.

E.2.2 English to Sign Language using a Tree-adjoining grammar

At the University of Pennsylvania [45], a system was developed to translate English into *American Sign Language* (ASL) using a *Synchronous Tree-adjoining grammar* (STAG) [32].

TAGs were used to represent the source language, and for every elementary tree of the English TAG, an equivalent elementary tree for the ASL TAG was defined. As the English TAG trees are parsed, the ASL TAG trees are constructed using synchronous TAGs. To finish the translation, an ASL gloss replaced every English word in the ASL TAG trees by using an English-to-ASL dictionary.

Sign Languages consist of two types of signs, *manual signs* (MS) and *non-manual signs* (NMS). An example of a MS would be the signing of TREE, while facial expressions are examples of NMS.

The MT system of the University of Pennsylvania incorporated some NMS components, such as intonational breaks, into the ASL TAGs. Other NMS were also incorporated into the translation model, for example in the sentence, *John opened the door slowly*, the translation would be *JOHN OPEN(0.0, -2.0, 0.0) DOOR*. This means that when the sign OPEN is signed, the movement in the y-direction will be slower than that of the pre-defined normal speed for that sign. In the translation, hints were also added for facial expressions such as frowns for questions, and the shaking of the head for yes-no questions. The system also incorporated spatial element of Sign Language into its model, including *directional verbs*.

The success of this system was attributed by the authors, to the fact that almost all ASL sentences are constructed as *subject*, *verb*, *object*.

E.2.3 English to Sign Language using Lexical Functional Grammar correspondence architecture

d'Armond Speers [16] suggests a model for translating English into ASL using *Lexical Functional Grammar* [24] correspondence architecture.

In this system, English text is analyzed and a *constituent structure* (c-structure), and a *functional structure* (f-structure) is created for each sentence. The c-structure is a phrase-structure tree that looks similar to a TAG. The f-structure contains grammatical information about the sentence such as its subject and its object, but still has a strong connection with the syntax of the original sentence.

Once the c-structure and f-structure for an English sentence is created, an ASL f-structure is created from the English f-structure. This new f-structure is then used to create the corresponding c-structure for ASL. Lastly, a *phonetical structure* (p-structure) is created from the ASL c-structure.

Speers constructed a program implementing this model called *ASL Workbench*. In this program, the process of translation is not fully automated, but rather human-aided. Issues such as *pronoun resolution* is left up to the user to resolve.

It should be noted that there are great similarities between this MT system and the one developed by the University of Pennsylvania using an STAG. In each case, there is a mapping from one structure in the source language to a similar kind of structure in the target language. In the STAG example, it is from one elementary tree to another, and in the model Speers suggests, it is from one f-structure to another. The difference however, is that using an STAG is a direct translation method, while the method described by Speers is a transfer method.

In the following section an overview is given of the problems that Speers discuss in his PhD thesis concerning problem areas in MT.

E.3 Problem areas concerning machine translation

Speers [16] mentions the following four pitfalls that may be found during MT:

The first pitfall for direct translation or transfer methods, is *syntactic ambiguity*. To illustrate this, the example as given in Speers is used. It should be left up to the syntax analyzer to resolve such ambiguities.

Example 20 *"One morning I shot an elephant in my pajamas. How he got in my pajamas, I don't know." -Julius "Groucho" Marx.*

Lexical ambiguity is found when a word in the source language could have more that one meaning, and in the target language have at least two of those meanings represented by different lexical items.

Example 21 *The word* book *in English, could mean* BOOK*, or* RESERVE*, in SASL.*

A problem that goes beyond syntax, or lexical entries, is *semantic ambiguity*. Again the example of Speers is used to explain this pitfall.

Example 22 *While driving, John hit a tree.*

The truth is, John didn't hit the tree, his car did. Because Sign Language is a *visual-spatial* language, if this sentence was literally translated, it would seem strange to an observer watching the signing of it.

Lastly the pitfall of *contextual ambiguity* exists. One example of contextual ambiguity is called pronoun resolution, and as mentioned before will be addressed in the next chapter. Pronoun resolution entails that if a pronoun is found in a sentence, then it is necessary to find out to whom the pronoun refers.

Appendix F

Test data

This appendix contains the modified text that was used during analysis by the preprocessor.

The following article is from

Media: Baby murder suspect [44] *The 24-year-old suspect in the murder of six-month-old Jordan-Leigh is to remain in custody. Police are hunting for four men she suspectedly hired to kill the infant.*

The four tricked their way into the Rondebosch home of the Nortons where they fatally stabbed the baby girl.

Dina, from Wynberg, was arrested on Friday following the murder of the baby a week ago.

She made her first court appearance at the Wynberg Court yesterday.

She was not asked to plead, and the case against her was postponed to June 27.

Dina did not apply for bail, which the state would have opposed. She will be held at Cape police station until her next appearance. The courtroom was packed, but the Norton family stayed away. Jordan-Leigh's grandfather said the family's main concern was to give the baby a dignified burial.

A funeral service will be held today at twelve thirty, followed by a private cremation.

Norton said he did not know Dina. His daughter Natasha had known her as the ex-girlfriend of Jordan-Leigh's father.

Norton said the support that the family received after the tragedy had been outstanding.

Media: India's Sharapova [30] *Teenager Mirza became the first Indian woman to reach the second round of Wimbledon. She battled to victory over Japan's Morigami yesterday.*

Mirza arrived at the club having already achieved an impressive list of firsts during 2005. She boosted her growing reputation by bludgeoning her opponent into submission.

Mirza's victory set up an intriguing show down with fifth seed Kuznetsova. Kuznetsova will now be aiming to avenge her surprise defeat by the Indian in Dubai.

The 18-year-old Mirza has turned into an overnight celebrity in her homeland. She became the first Indian woman to reach the third round of a grand slam at the Australian Open.

Expectations for her to have a good run at Wimbledon, have been high.

In five months, she has become a sporting idol and a fashion icon. And like Sharapova, Mirza needs her own entourage of security guards when stepping out.

The attention surprises her everywhere. It is a tough life. It is very busy. It is very different.

Mirza thinks in India people are very excited. Suddenly, when she goes out, she needs security.

It is really nice that people get inspired. She hopes they will have many more tennis players from India playing at this level.

Academic: Fant [17] *ASL is the Sign Language most deaf people use when they are communicating among themselves. ASL has its own grammar. The grammar is different from English grammar. You must approach ASL in the same manner you would approach any foreign language. Do not expect ASL to be like English or to conform to rules of English grammar. Do not ask why ASL, or any language, has a certain structure. Ask only how it works. It does not help to ask Spanish-speaking people why they put adjectives after nouns. Spanish people just do, and you must accept that. Some of the constructions in ASL may look strange at first. This is because we say things differently from English. After a while they will look as natural as English.*

It is a common misconception that ASL is only the finger spelling of English words. Finger spelling, using the alphabet to spell entire words letter by letter, is occasionally incorporated into ASL. However, the vocabulary of ASL consists of signs.

The format of this book is not that of a traditional foreign language textbook.

Academic: TAGs [23] *We first give a short introduction to TAGs. Next we briefly state the results concerning both the properties of the string sets and tree sets. We will also describe the notion of lexicalization of grammars. We investigate the relationship of lexicalization to context-free grammars and TAGs and then summarize the issues on lexicalization. We then describe a model that greatly corresponds to TAGs. As we have said earlier TAGs were motivated by some important linguistic considerations. Time aspects of these considerations are mathematically important. Hence we have presented a brief discussion of these issues together with some simple examples. We also present in section 9 some variants of TAGs that are currently under investigation. We then present a bottom up predictive parser for TAGs. This is both theoretically and practically important. Next we offer some concluding remarks.*

The motivations for the study of TAGs are of linguistic and formal nature. Using structured objects as the elementary objects, it is possible to construct formalisms directly related to the strong generative capacity.

General: Birthday of the world [20] *I live in the oldest city in the world. Long before there were kings in Karhide, Rer was a city. It was the marketplace and meeting ground for all the land. The House was a center of learning. It was a refuge and a judgment seat fifteen thousand years ago. Karhide became a nation here, under the kings, who ruled for a thousand years. In the thousandth year, the Unking, cast the crown into the river from the palace towers. He proclaimed an end to dominion. The time they call the Flowering, the Summer began then. It ended when the Harge took power and moved their capital across the mountains to Erhenrang. The old palace has been empty for centuries. But it stands. Nothing in Rer falls down. The Arre floods through the street tunnels every year in the thaw. Winter blizzards could bring thirty feet of snow , but the city stands. Nobody knows how old the houses are, because they have not been rebuilt forever. Each house sits in its gardens without respect to the position of any of the others. The houses are so random and ancient as hills. The roofed streets and canals run about among them. Rer is all corners. We say Harge left, because they were afraid of what could be around the corner.*

General: Five days in Paris [36] *The road to Faviere was boring and long. Peter drove faster than he thought. It took him exactly ten hours. He drove slowly into town at six o'clock, just as the sun came up. The apple was long gone, and the Evian bottle was almost empty on the seat beside him. He drove with all the windows down. Now that he had reached his destination, he was truly exhausted. He had been awake all night, for the second time in two days. Peter's excitement was there, and the adrenaline spurred him on. It was too early to look for her. Except for the fishermen beginning to arrive at the dock, everyone in Laviere was still sleeping. Peter pulled off to the side of the road , and climbed into the backseat. It was cramped, but it was exactly what he needed.*

It was nine o'clock when he woke, and he heard children playing near the car. Their voices were raised as they ran by, and Peter could hear seagulls overhead. There were a variety of sounds and noises as he sat up. He was feeling as though he had died. It had been a long night, and a long drive. But if he found her, it would be worth it. As he sat up and stretched, he caught a glimpse of himself in the rearview mirror and laughed.

Appendix G

Preprocessor schema XML

```
?xml version="1.0" encoding="UTF-8"?> <xs:schema
xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
    <xs:element name="ppDocument">
        <xs:annotation>
            <xs:documentation>Root element</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="paragraph" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="paragraph">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="sentence" maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="id" type="xs:ID" use="required"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="sentence">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="sentence_str" type="xs:string" minOccurs="0"/>
                <xs:element ref="expressiveness" minOccurs="0"/>
                <xs:choice maxOccurs="unbounded">
                    <xs:element ref="node"/>
```

```
                <xs:element ref="phrase"/>
            </xs:choice>
        </xs:sequence>
        <xs:attribute name="id" type="xs:ID" use="required"/>
    </xs:complexType>
</xs:element>
<xs:element name="phrase">
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="phraseType">
                <xs:attribute name="id" type="xs:ID" use="required"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<xs:element name="node">
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="nodeType">
                <xs:attribute name="id" type="xs:ID" use="required"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<xs:element name="morphoSyntacticInfo" type="morphoSyntacticInfoType"/>
<xs:element name="location" type="locationType"/>
<xs:element name="expressiveness" type="expressivenessType"/>
<xs:complexType name="phraseType">
    <xs:sequence>
        <xs:element ref="morphoSyntacticInfo" minOccurs="0"/>
        <xs:element ref="location" minOccurs="0"/>
        <xs:choice maxOccurs="unbounded">
            <xs:element ref="node"/>
            <xs:element ref="phrase"/>
        </xs:choice>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="nodeType">
    <xs:sequence>
        <xs:element name="part_of_speech" type="xs:string"/>
        <xs:element name="word_string" type="xs:string"/>
        <xs:element name="word_root" type="xs:string" minOccurs="0"/>
        <xs:element ref="morphoSyntacticInfo" minOccurs="0"/>
```

```
            <xs:element ref="location" minOccurs="0"/>
            <xs:element ref="expressiveness" minOccurs="0"/>
            <xs:element name="boundary" type="xs:string" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="morphoSyntacticInfoType">
        <xs:sequence>
            <xs:element name="person" type="xs:string"/>
            <xs:element name="number" type="xs:string"/>
            <xs:element name="gender" type="xs:string"/>
            <xs:element name="pseudo_name" type="xs:string" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="locationType">
        <xs:sequence>
            <xs:element name="abs_pos" type="xs:string"/>
            <xs:element name="start_loc" type="xs:string"/>
            <xs:element name="num_loc" type="xs:int"/>
            <xs:element name="action" type="xs:string"/>
            <xs:element name="direction" type="directionType" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="directionType">
        <xs:sequence>
            <xs:element name="x"/>
            <xs:element name="y"/>
            <xs:element name="z"/>
            <xs:element name="v"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="expressivenessType">
        <xs:sequence>
            <xs:element name="type" type="xs:string" minOccurs="0"/>
            <xs:element name="class" type="xs:string" minOccurs="0"/>
            <xs:element name="form" type="xs:string" minOccurs="0"/>
            <xs:element name="accent" type="xs:string" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

Appendix H

Preprocessor schema

This appendix describes an XML schema that defines the XML documents that the preprocessor generates as output (see Appendix G for the schema XML).

H.1 Introduction

The results of preprocessing a text document by the preprocessor is stored in an XML file. A schema is used to define for some XML document, what elements the document contains, the order in which they appear, the element content, and the attributes of elements if they are defined.

An XML document is used to store the information generated by the preprocessor because it is a flexible and universal format. XML makes it possible to keep the grammatical structure and part-of-speech information intact, while adding the metadata created by the algorithms implemented in the preprocessor.

In the rest of this appendix an overview of the preprocessor XML schema is given by looking at the content model for each element and complex type. An XML schema is defined by using *XML element syntax*, and is therefore in itself, a well formed XML document.

H.2 Preprocessor XML schema overview

The root element of an XML document created by the preprocessor is the *ppDocument* element. This element could contain either no, or between one and an unbounded number of *paragraph* elements.

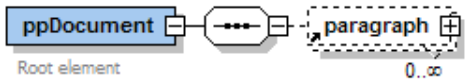

Figure 28: XSD content model of the root element *ppDocument*.

The *paragraph* element has an *id* attribute, and contains between one and an unbounded number of *sentence* elements.

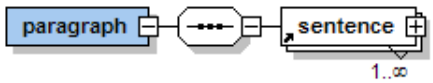

Figure 29: XSD content model of the element *paragraph*.

The *sentence* element has an *id* attribute, and has a sequence of child elements. The first child element is the *sentence string* element, it contains the sentence string from the original text input, and is optional. The next element which is also optional, is called the *expressiveness* element and is of the complex type *expressivenessType*. After the two optional elements, there will be between one and an unbounded number of elements chosen either as a *node* element of the complex type *nodeType*, or as a *phrase* element of the complex type *phraseType*.

The complex type *phraseType* is defined as an element that contains a sequence of child elements. The first element is the *morphoSyntacticInfo* element of complex type *morphoSyntacticInfoType*, and is optional. The next element is the *location* element of complex type *locationType*, and is also optional. After the two optional elements, there will be between one and an unbounded number of

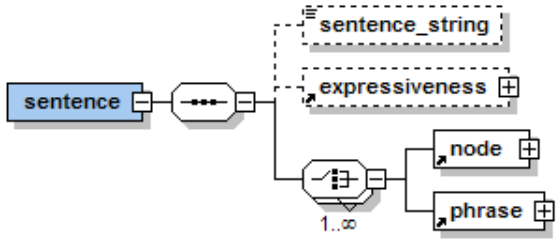

Figure 30: XSD content model of the element *sentence*.

elements chosen either as a *node* element of the complex type *nodeType*, or as a *phrase* element of the complex type *phraseType*.

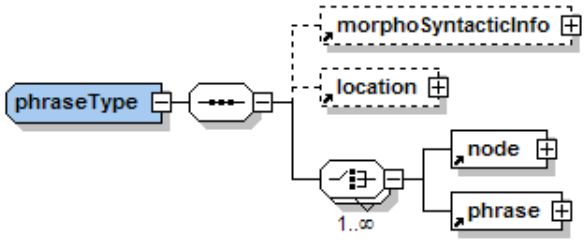

Figure 31: XSD content model of the complex type *phraseType*.

The complex type *nodeType* is defined as an element that contains an *id* attribute, and a sequence of child elements. The first child element is the *part of speech* element. This is followed by the *word string* element. The third element is the *word root* element, it is optional, and will usually only be used when the word root is different from the word string found in the sentence. The next element is the *morphoSyntacticInfo* element of complex type *morphoSyntacticInfoType*, and is optional. The following element is the *location* element of complex type *locationType*, and is also optional. Next is the *expressiveness* element of complex type *expressivenessType*, which is optional. Finally the *boundary* element is found, which is optional.

The complex type *morpoSyntacticInfoType* is defined as an element that contains a sequence of child elements. The four child elements in sequence are the

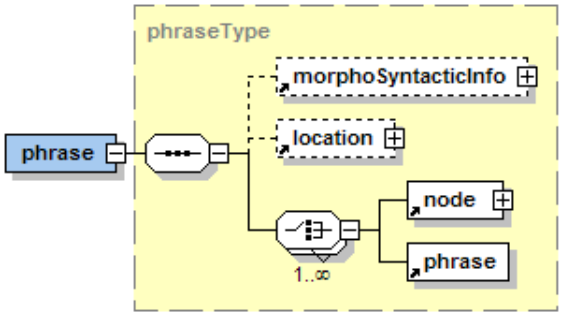

Figure 32: XSD content model of the element *phrase*.

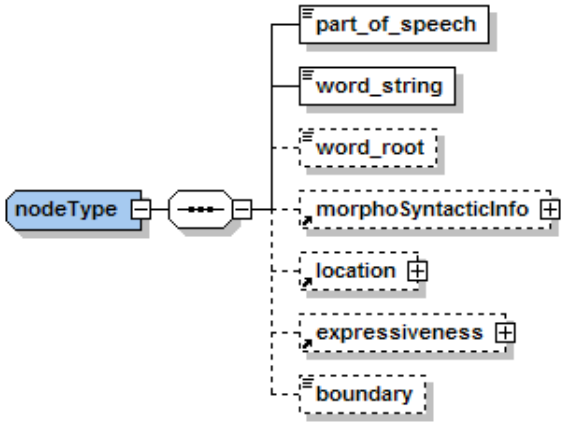

Figure 33: XSD content model of the complex type *nodeType*.

person, *number*, *gender*, and the optional *pseudo name* elements.

he complex type *locationType* is defined as an element that contains a sequence of child elements. The five child elements in sequence are the *abs pos*, *start loc*, *num loc*, *action* and the optional *direction* elements. The *direction* element is of complex type *directionType*. The complex type *directionType* is an element that contains four child elements in sequence. These elements are the x, y, z and v elements.

The complex type *expressivenessType* is defined as an element that contains a sequence of child elements. The four child elements in sequence, and all of them

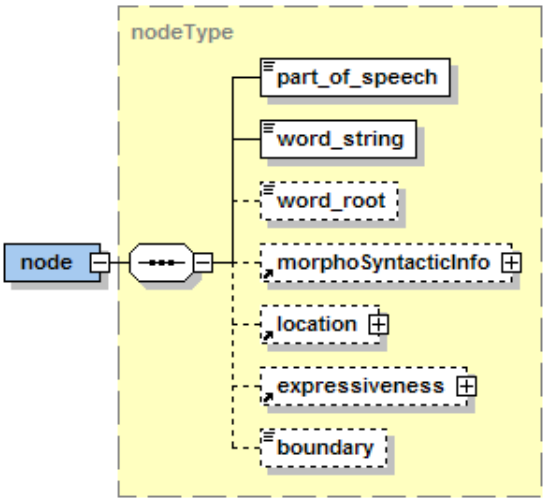

Figure 34: XSD content model of element *node*.

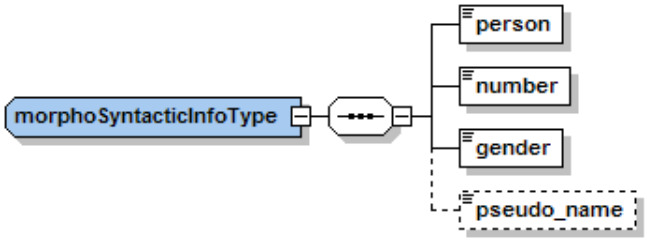

Figure 35: XSD content model of the complex type *morphoSyntacticInfoType*.

optional, are the *type*, *class*, *form*, and *accent* elements.

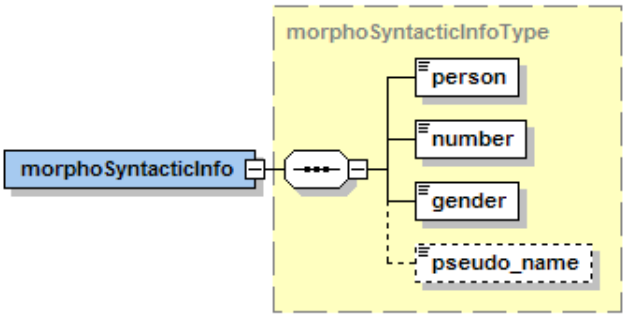

Figure 36: XSD content model of the element *morphoSyntacticInfo*.

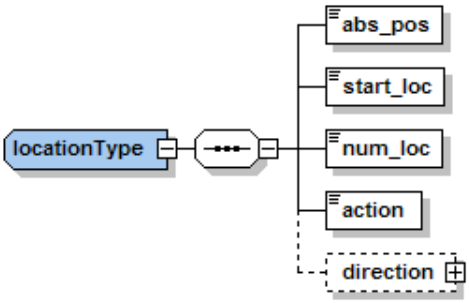

Figure 37: XSD content model of complex type *locationType*.

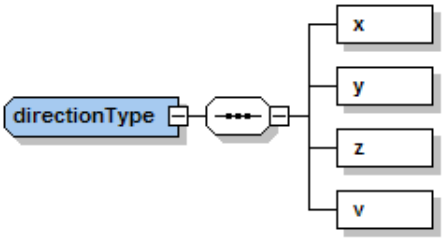

Figure 38: XSD content model of the complex type *directionType*.

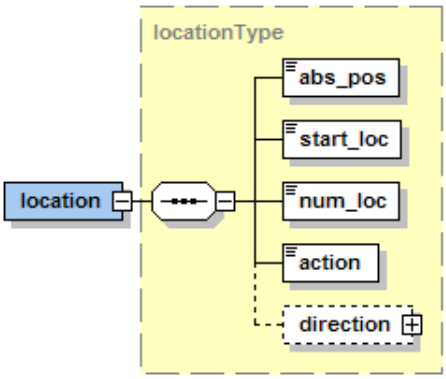

Figure 39: XSD content model of the element *location*.

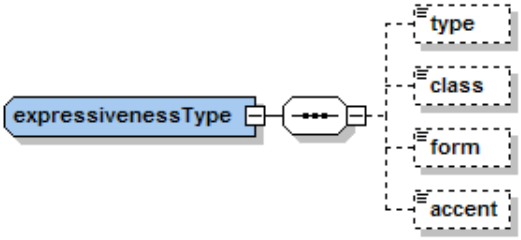

Figure 40: XSD content model of the complex type *expressivenessType*.

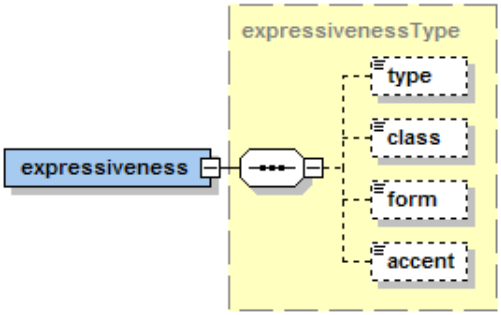

Figure 41: XSD content model of element *expressiveness*.