UNIVERSITY OF GOTHENBURG

Towards a Wide-Coverage Grammar for Swedish Using GF

Master of Science Thesis in the Programme Computer Science

MALIN AHLBERG

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
Göteborg, Sweden, January 2012

Towards a Wide-Coverage Grammar for Swedish Using GF

Malin Ahlberg

Examiner: Aarne Ranta

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden January 2012

Abstract

This thesis describes work towards a wide-coverage grammar for parsing and generating Swedish text. We do this by using the dependently typed grammar formalism GF, a functional programming language specialized at describing grammars. The idea is to combine existing language resources with new techniques, with an aim to achieve a parser for unrestricted Swedish. To reach this goal, problems of computational as well as linguistic nature had to be solved. The work includes the development of the grammar – to identify and formalize grammatical constructions frequent in Swedish – as well as methods for importing a large-scale lexicon and for evaluating the parser. We present the methods and technologies used and discuss the advantages and problems of using GF for modeling large-scale grammars. We further discuss how our long-term goal can be reached by combining our rule-based grammar with statistical methods.

Our contribution is a wide-coverage GF lexicon, a translation of a Swedish treebank into the GF notation and an extended Swedish grammar implementation. The grammar is based on the multilingual abstract syntax given in the GF resource library, and now also covers constructions specific to Swedish. We further give an example of the advantage of using dependent types when describing grammar and syntax, in this case for dealing with reflexive pronouns.

Acknowledgments

Many people have helped me during this project and made this work possible. I would first of all like to thank Center of Language Technology, that has funded the project.

Further, thanks to my excellent supervisor Ramona Enache for all her help and guidance in every phase and all aspects of the work. Thanks to Elisabet Engdahl for sharing her seemingly unlimited knowledge of Swedish grammar. She has also has acted as a second supervisor, and given me very helpful comments and suggestions. Thanks to Aarne Ranta for all his great ideas and for letting me do this project.

I am also grateful to Krasimir Angelov, Markus Forsberg, Peter Ljunglöf, Lars Borin and many others who have contributed with ideas and inspiration and shown interest in this work.

Finally, I would like to thank my friends and family. Special thanks to Dan for all his support, advice and patience and – most importantly – for being such a good friend.

Contents

Chapter 1

Introduction

Grammatical Framework [Ranta, 2011] is a dependently typed grammar formalism. It is based on Martin-Löf type theory which allows reasoning within the programming language. GF has strong support for multilinguality and has so far been used successfully for controlled languages [Angelov and Ranta, 2009], while recent experiments have showed that it is also possible to use the framework for parsing free language [España-Bonet et al., 2011]. Parsing, that is the task of automatically identifying morphological and syntactical structures, is receiving increasing interest, especially considering the steadily growing amount of data available online. So far there is no freely available grammar-driven parser that gives a deep analysis for Swedish. The fact that a parser or grammar is not freely available does not only restrict its usage but also its possibilities of being further developed. Our goal is to create an open-source Swedish grammar from which we derive a parser accepting all sentences described by the given rules. As a freely available resource, it can be continuously enhanced and made use of in other projects.

To build the grammar from scratch would be not only time consuming but would also mean that already existing systems would have to be reimplemented. In order to not reinvent the wheel we proceed from a combination of well-tested sources. We start from a GF resource grammar, consisting of an abstract and a concrete syntax file defining a set of rules for morphology and syntax. This is what is meant by grammar in this thesis, as opposed to grammar in the traditional linguistic sense. From GF, we get a well-defined system for describing language, as well as a strong connection to and possibility of translation between the more than 20 other languages implemented in the framework. Further, we use the extensive lexicon Saldo and the treebank Talbanken.

1.1 Aims

The purpose of this project has been to prepare the GF grammar for parsing of unrestricted Swedish. This has meant to develop earlier techniques to fit for Swedish, create methods for achieving and keeping a large-scale lexicon and to adapt the existing resource grammar to model more language specific constrictions. The project was divided into three subsections, aiming at

- Extending the Swedish GF grammar
- Importing the lexicon Saldo
- Creating translation between Talbanken and GF

1.2 Outline

The thesis is divided into 6 chapters. We start by giving background information of areas relevant to the project. Grammatical Framework is presented in section 2.1 while a more profound description of the Swedish resource grammar is given in section 4.1. Section 2.4 gives an introduction to the Swedish language, and brief presentations of Saldo and Talbanken are found in section 2.3 and 2.2 respectively. A summary of related work can be found in section 2.5.

Chapters 3 - 5 present the methodology, implementation and some results. First we describe and evaluate the implementation of Saldo in chapter 3. The work on the GF grammar is described in chapter 4. In chapter 5 we account for an automatic mapping of Talbanken trees to GF.

Finally, the conclusion and evaluation are presented in chapter 6 together with some areas of future work.

Chapter 2

Background

The work described in this thesis is part of a bigger project which aims at using GF for parsing unrestricted Swedish. In previous work[1], a start was made to extend the Swedish GF grammar and a tool for lexical acquisition was developed. We now construct a bigger and more expressive grammar as well as a large scale GF lexicon. As all GF grammars, this one defines a parser, and we develop it by getting examples, ideas and test material from the treebank Talbanken. The project is hence heavily depending on three resources, which will be described in this section.

2.1 Grammatical Framework

The main component of the project is the Grammatical Framework (GF) [Ranta, 2011], a grammar formalism based on Martin-Löf type theory [Martin-Löf, 1984]. GF is designed for multilingual applications and represents a formalism stronger than mildly context-free grammars. The framework's expressiveness is hence stronger than Tree Adjoining Grammars [Joshi, 1975] and Head Grammars [Pollard, 1984], and shown equivalent to Parallel Multiple Context-Free Grammar [Seki et al., 1991] in [Ljunglöf, 2004].

GF is a strongly typed functional programming language, inspired by ML [Milner et al., 1997] and Haskell [Simon Thompson, 1999]. It is also a logical framework, and the built-in functionality for logic and reasoning is inspired by λProlog [Xiaochu Qi, 2009] and by Agda [Norell, 2008], with which GF also shares its type checking algorithm. The first version of the framework was implemented at Xerox Research Center in Grenoble and is now mainly developed in Gothenburg. One of the biggest achievements is a library covering the basic morphological and syntactic structures of more than 20 languages (see section 2.1.2).

A grammar written in GF can be used for both parsing and generation. The parsing algorithm is incremental and has polynomial time and space complexity [Angelov, 2011b]. The GF package also provides various tools for working with and using the grammars: a compiler, an interpreter and a runtime system. The grammars can be compiled into portable grammar format (PGF) [Angelov et al., 2010], supported by Java, Java script, C and Haskell libraries. The interpreter, the GF shell, allows the user to test grammars by commands for parsing, visualization of parse trees, random generation, word alignment, morphological quizzes etc. The shell can be tried out online together with an interactive

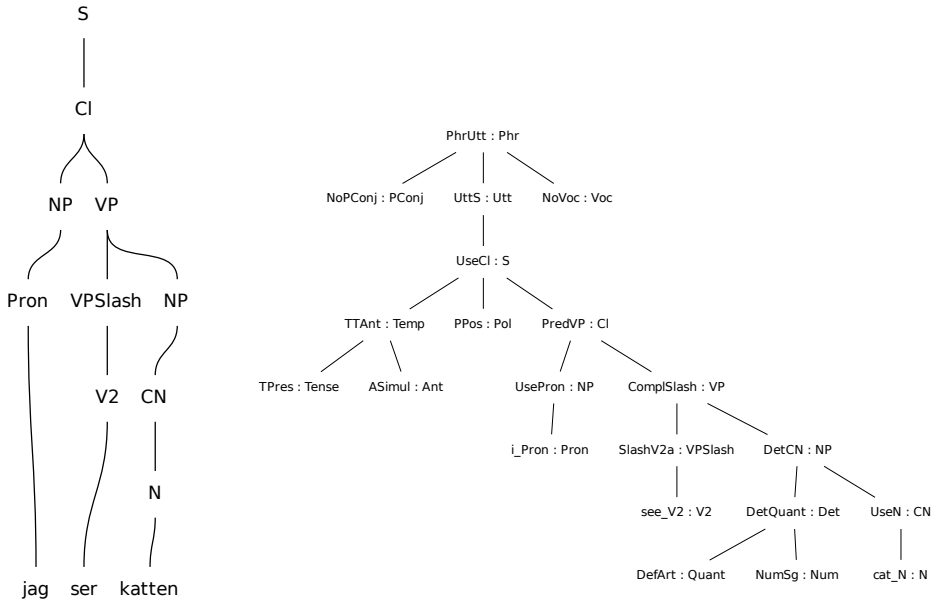
[1] web.student.chalmers.se/~mahlberg/SwedishGrammar.pdf

GF editor [2]. Figure 2.1 shows commands for parsing, where the results are shown as in

```
> parse "jag ser katten" | visualize_tree  -format=pdf -view=evince
> parse "jag ser katten" | visualize_parse -format=pdf -view=evince
```

Figure 2.1: Example of how the GF shell can be used.

figure 2.2. GF uses Graphviz[3] to visualize the trees and the given commands (fig. 2.1) specify that the output format should be PDF and that these files should be opened by the program Evince. The user can choose to see the *parse tree* (fig. 2.2a) or the *abstract tree* (fig. 2.2b). Abstract trees are more verbose and show all functions and types used for parsing the sentence.



(a) Swedish parse tree

(b) GF abstract tree

Figure 2.2: Abstract tree and parse tree for the sentence "Jag ser katten".

Parse trees on the other hand show only the types assigned to the words and phrases. Information about tense, polarity etc., which are explicitly given in the abstract tree, are not reproduced in the parse tree. Hence, parse trees do not give complete representations but model the parse results in a transparent manner. For our example, the definiteness and number of the noun '*katten*' is shown as `DetQuant DefArt NumSg` in the abstract tree while the parse tree only shows that the noun phrase consists of one noun. In the corresponding English parse tree, figure 2.3, the noun is explicitly quantified by the article *'the'*, and the determiner, the first argument to the function `DetCN`, is therefore shown in the parse tree.

[2]http://www.grammaticalframework.org/demos/gfse/
[3]http://www.graphviz.org/

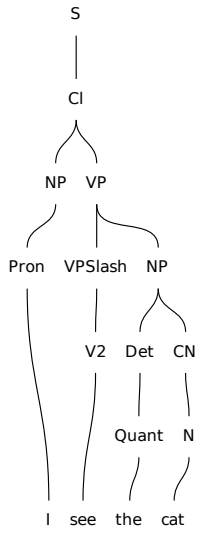


Figure 2.3: English parse tree

The scope of GF grammars has so far been controlled language, a subset of a natural language used for a restricted domain. By restricting the coverage, the number of ambiguities can be limited and it can be ensured that the semantics is preserved during translation. Inherited ambiguities as well as ambiguities arising from multilinguality will remain, but can be controlled more easily.

The use of controlled language thus gives the possibility of good and reliable translation and GF has successfully been used for this in a number of projects. WebAlt [Caprotti, 2006] aims to develop language independent material for mathematics problems by using GF grammars. The formal software verification tool KeY [Burke and Johannisson, 2005] used GF for translating formal specification to natural language. GF has further been used for describing dialogue grammars [Ljunglöf et al., 2005], and the framework is one of the main component in the European project MOLTO[4], which develop online translation between to 15 languages.

2.1.1 Writing a GF grammar

The key feature in GF is the distinction between *abstract* and *concrete* syntax. The abstract syntax represents the internal structure and models the semantics without concern for language specific features such as agreement or word order. An abstract grammar can be implemented by a set of concrete grammars, each representing one language. As a comparison, the abstract and concrete syntax may be thought of as f-structures and c-structures in Lexical Functional Grammar [Bresnan, 1982].

```
abstract TestGrammar = {
 cat N ; V ; S ;

 fun
   Pred : N -> V -> S ;
   cat_N : N ;
   sleep_V : V ;
}
```

Figure 2.4: A small example of an abstract syntax

[4] http://www.molto-project.eu/

The example in figure 2.4 shows an abstract grammar defining three categories, one for nouns, one for verbs and one for sentences. The abstract grammar also gives the function types. In this case we have `Pred`, which tells us that by taking a noun and a verb we can form a sentence. No information of how this is done is given at this stage. The grammar also defines two words, the noun `cat_N` and the verb `sleep_V`.

```
concrete TestGrammarSwe of TestGrammar = {
  lincat N, V, S = Str ;

  lin Pred n v   = n ++ v ;
      cat_N   = "katten"  ;
      sleep_V = "sover"  ;
}
```

Figure 2.5: A Swedish concrete grammar

Figure 2.5 shows how the abstract grammar can be implemented for Swedish. Nouns, verbs and sentences are all defined as strings, `Str`. The function `Pred` simply glues the two strings 'katten' and 'sover' together:
`Pred cat sleep = "katten sover"`.
We get a more complicated example if we allow the nouns to be used in both plural and singular. We add a category `N'` to the abstract, which represents a noun with a fixed number, and we introduce two functions for setting the number: `NSg : N -> N'` and `NPl : N -> N'`. Figure 2.6 introduces some new concepts: records, tables and parameters. In the concrete

```
abstract TestGrammar = {
 cat N ; N' ; V ; S ;

 fun
   Pred : N' -> V -> S ;
   NSg : N -> N' ;
   NPl : N -> N' ;
   cat_N : N ;
   sleep_V : V ;
}
```

```
concrete TestGrammarSwe of TestGrammar = {
  lincat V, S, N' = Str ;
         N = {s : Num => Str} ;
  lin
    Pred n v = n ++ v ;
    NPl n = n.s ! Pl ;
    NSg n = n.s ! Sg ;
    cat_N = {s = table {Sg => "katten" ;
                        Pl => "katterna"}};
    sleep_V = "sover"  ;
  param Num = Sg | Pl ;
  }
```

Figure 2.6: A modified grammar

syntax, `N` is defined to be a record consisting of the field `s`. The type of `s`, `Num => Str` shows that it is a table, which given a parameter of type `Num` returns a string. `Num` is defined to either have value `Sg` or `Pl`. The dot (`.`) is used for projection and the bang (`!`) as a selection operator. `n.s ! Sg` thus means that we use the branch for `Sg` in field `s` of `n`.

When implementing an English version of the grammar, we encounter another problem: the verb form depends on the number of the noun. We solve this by letting `N'` carry information about its number and letting `Pred` pass this on to the verb. Finally, the type of `V` is put into a table, showing the verbs forms for each number.

```
concrete TestGrammarEng of TestGrammar = {
  lincat S  = Str ;
         V  = {s : Num => Str} ;
         N  = {s : Num => Str} ;
         N' = {s : Str ; num : Num} ;
  lin
    Pred n v = n.s ++ v.s ! n.num ;
    NPl n = {s = n.s ! Pl ; num = Pl} ;
    NSg n = {s = n.s ! Sg ; num = Sg} ;
    cat_N = {s = table {Sg => "the cat" ;
                        Pl => "the cats"}};
    sleep_V = {s = table {Sg => "sleeps" ;
                          Pl => "sleep"}};
  param Num = Sg | Pl ;
  }
```

Figure 2.7: English implementation

We now have two implementations of the abstract, one for Swedish and one for English. The resulting GF grammar is able both to parse a string to an abstract tree and to go in the other direction; to produce a string of natural language given an abstract tree. This step is called linearization. Translation is a consequence of this, we can parse a Swedish string and then linearize the resulting abstract tree to English.

2.1.2 The resource library

The GF package provides an useful resource library [Ranta, 2009], covering the fundamental morphology and syntax of more than 20 languages. There is also a small test lexicon included, containing a few hundred common words. Since the languages all share the same abstract syntax translation between any given pair is possible, which is valuable when implementing multilingual applications.

The resource grammars describe how to construct phrases and sentences and how to decline words. The latter is done by smart paradigms: functions that analyse some given forms of a word to find out how the inflection table should look. For example, the declination of many Swedish nouns can be determined by looking at the singular form only.

1st declination	5th declination
flick**a**	hjärt**a**
flick**an**	hjärt**at**
flick**or**	hjärt**an**
flick**orna**	hjärt**ana**

This is the case for 'flicka', a noun belonging to the first declination. For others, like "hjärta", also the plural form "hjärtan" is needed. The worst case for nouns is four needed forms, both singular and plural in definite and indefinite form. Section 4.1 will give a more thorough description of the Swedish resource grammar.

2.1.3 Frontiers of Grammatical Framework

As an open source-project, GF is constantly being developed and improved. New languages are added, the compiler is being improved, ways of using it in more efficient and easy-going manners are added and the possibilities to use GF in different environments increased. There is research on how to make more use of the dependent types, for reasoning by using ontologies [Enache and Angelov, 2011] or generating natural language via Montague semantics [Ranta, 2004].

2.2 Talbanken

For testing and evaluation of the grammar and lexicon, we needed to be able to compare them against a reliable source. Talbanken [Einarsson, 1976] was perfect for our purpose, being a freely available, manually annotated, large-scale treebank. It is analyzed with the MAMBA annotation scheme (Teleman, 1974) and consists of four parts. Two of them are transcriptions of spoken language, one a collection of text written by high school students, and one, section P, consists of professionally written Swedish gathered from newspapers, brochures and textbooks.

Talbanken was also used to train the Swedish version of the Malt parser [Hall, 2007] and was then redistributed in an updated version, Talbanken05 [Nivre et al., 2006]. It is released in Malt[5] and Tiger[6] XML-formats where the trees have been made deeper and more detailed while still containing the lexical MAMBA layer. The Malt parser was trained on section P of Talbanken, and these more than 6000 sentences have been used our project. The treebank has served as an inspiration and an evaluation source throughout the project. An automatic mapping between its trees and the abstract trees from GF has been done, which will be explained in section 5.

2.3 Saldo

A good parser needs a good lexicon. We have used Saldo [Borin et al., 2008], a large electronic lexicon developed and maintained at Gothenburg University. It is built on Svenskt Associationslexikon and contains information about more than 120 000 modern Swedish words. For each word there is semantic, syntactical and a morphological information. The user can find examples of usage in corpora, graphs of semantically connected words and some suggestions for how to analyse compounds.

The semantic aspect of Saldo requires that words with multiple meanings are separated into different entries. The word 'uppskatta' for example, has two entries, with slightly different semantics: *enjoy*

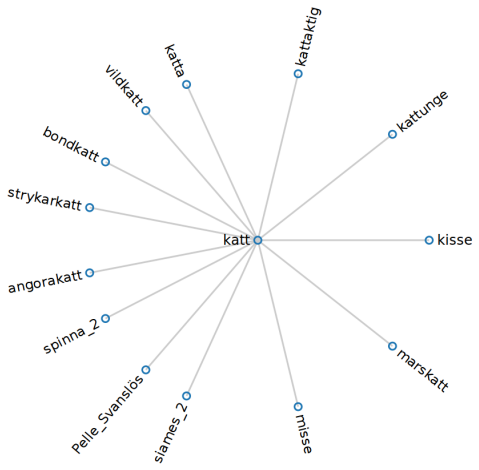
(1) Jag uppskattar teater.
"I enjoy theater."

[5] http://w3.msi.vxu.se/ nivre/research/MaltXML.html
[6] http://www.ims.uni-stuttgart.de/projekte/TIGER/TIGERSearch/doc/html/TigerXML.html

SALDO

grundform	katt
mönster	nn_3u_film
ordklass	nn
inherenta drag	u
böjningstabell	
sg indef nom	katt
sg indef gen	katts
sg def nom	katten
sg def gen	kattens
pl indef nom	katter
pl indef gen	katters
pl def nom	katterna
pl def gen	katternas
ci	katt/katt-
cm	katts/katt/katts-/katt-
sms	katt-

(a) Morphological info for 'katt'



(b) Graph showing the hyponyms of 'katt'

Figure 2.8: Information in Saldo

or *estimate*:

(2) Värdet uppskattades till 5 miljoner.
"The value was estimated to 5 million."

'uppskatta' is however declined the same way for both interpretations of the word. Hence *Saldo's morphological lexicon* only keeps one entry of 'uppskatta'. For the homonyms that do not share the whole inflection tables, different entries are kept also in the morphological lexicon.
For our purpose, only the morphological section, provided under LPGL in XML format, was needed. The data can be processed and translated to GF format as described in section 3.

2.4 Swedish

Swedish [Teleman et al., 1999, Inl. §3] is a North-Germanic language, closely related to Norwegian and Danish. The languages share most of their grammatical structures and are mutually intelligible. Swedish is also one of the official languages in Finland and altogether spoken by approximately 9 million people. Swedish syntax is often similar to English, but the morphology is richer and the word order slightly more intricate.

2.4.1 Post-nominal articles

Unlike most of the world's languages Swedish express not only the number but also the definiteness of a noun by suffixes. The endings are decided by the noun's gender, *neuter* or *non-neuter*.

	Indefinite	Definite	Gender
Singular	en katt	katten	*non-neuter*
	a cat	*the cat*	
Plural	katter	katterna	
	cats	*the cats*	
Singular	ett hus	huset	*neuter*
	a house	*the house*	
Plural	hus	husen	
	houses	*the houses*	

A definite article or a determiner is necessary when the noun is modified by adjectival phrase; sentence (3a) is not grammatically correct. As adjectives also marks definiteness, this is marked on three places in sentence (3b).

(3) a. *Gamla katten sov.
[+DEF] [+DEF]
"The old cat slept."

b. Den gamla katten sov.
[+DEF] [+DEF] [+DEF]
"The old cat slept."

However, for some determiners, ie. 'min' (*'my'*), the noun should be in indefinite form.

(4) min trötta katt
[+DEF] [+DEF] [-DEF]
"my tired cat"

2.4.2 Verb second

Swedish is a verb-second language [Josefsson, 2001, p.116]: the second constituent of a declarative main clause must consist of a verb. The normal word order is subject-verb-object, but any syntactic category can be fronted [Holmes and Hinchcliff, 1994, §1027]. This is called topicalisation and is very common, especially for temporal and locative adverbial phrases. The examples 5 - 7 all have the same propositional meaning, but vary in how the content is presented.

(5) Du ser inte mig.
you see not me
"You don't see me."

(6) Mig ser du inte.
me see you not

(7) Inte ser du mig.
not see you me

Inverted word order marks questions

(8) Såg du mig inte?
saw you me not
"Didn't you see me?"

The word order in subordinate clauses in also slightly modified,

(9) Main sentences:
Jag såg Anna men hon såg **inte** mig
I saw Anna but she saw ***not*** *me*
"I saw Anna but she didn't see me"

(10) Subordinate sentence:
Jag förstod att Anna **inte** såg mig.
I understood that Anna ***not*** *saw me*
"I understood that Anna didn't see me"

2.4.3 Passive voice

There are two ways of forming passive verb phrases in Swedish: the **periphrastic passive**, formed by using the modal auxiliary verb 'bli' (*'become'*).

(11) Tjuven blev tagen av polisen
the thief was taken by the police
"The thief was arrested by the police"

and the **s-passive** which is formed by adding an *s* to the verb:

(12)

Passive	Active
Tjuven togs av polisen	Polisen tog tjuven
the thief took+**s** *by the police*	*the police took the thief*
"The thief was arrested by the police"	"The police arrested the thief"

The s-passive is more commonly used than periphrastic passive, for both written and Swedish [Teleman et al., 1999, Pass. §1] and dominates especially when the subject is inanimate.

2.4.4 Impersonal constructions

Constructions with 'det är/var' (*'it is/was'*) are very common in Swedish [Holmes and Hinchcliff, 1994, §309d]:

(13) Det var roligt att höra.
it was nice to hear.
"I'm glad to hear that."

'Det' is also used as formal subject in presentational constructions where the real subject is put in the position of an object.

(14) Det står en älg på fältet.
it stands a moose on the field
"There is a moose in the field."

2.4.5 Reflexive pronouns

The Scandinavian language have special reflexive pronouns and reflexive possessive pronouns for the 3rd person [Holmes and Hinchcliff, 1994, §310 & 319], distinct from the normal 3rd person forms.

(15)
a. Han slog **sig**.
He hit him self.

b. Han såg **sitt** barn.
He saw his (own) child.

(16)
a. Han slog **honom**.
He hit him (another person).

b. Han såg **hans** barn.
He saw his (another person's) child.

The 1st and 2nd persons use the normal personal pronoun in object form as reflexive pronouns.

2.5 Related work

Many years of research have lead to many interesting language technology tools for Swedish. An example is the well-known data-driven Malt parser [Hall, 2007], which has been trained on Talbanken. There are also a number of grammar-based parsers, although none is freely available. The cascaded finite state parser CassSwe [Kokkinakis and Kokkinakis, 1999] and The Swedish Constraint Grammar [Birn, 1998] give syntactic analyses. Swedish FDG (Voultanien,2001) uses the Functional Dependency Grammar [Tapanainen and Järvinen, 1997], an extension of the Constraint Grammar formalism, and produces a dependency structure focusing on finding the nominal arguments.

The LinGO Grammar Matrix [Bender et al., 2002], is a starter-kit for building Head-Driven Phrase Structure Grammars [Pollard and Sag, 1994] (HPSG) providing compatibility with tools for parsing, evaluation, semantic representations etc. Translation is supported by using Minimal Recursion Semantics [Copestake et al., 1999] as an interlingua.
There is a collection of grammars implemented in this framework, giving broad-coverage descriptions of English, Japanese and German. The Scandinavian Grammar Matrix [Søgaard and Haugereid, 2005] covers common parts of Scandinavian, while Norsource (Hellan, 2003) describes Norwegian. A Swedish version was based upon this (SweCore, Ahrenberg) covering the morphology and some differences between Swedish and Norwegian. Further, there is the BiTSE grammar [Stymne, 2006], also implemented using the Lingo Matrix, which focuses on describing and translating verb frames.

The Swedish version of the Core Language Engine (CLE) [Gambäck, 1997] gives a full syntactic analysis as well as semantics represented in 'Quasi logical form'. A translation to English was implemented and the work was further developed in the spoken language translator [Rayner et al., 2000]. Unfortunately, it is no longer available. The coverage of the Swedish CLE is also reported to be very limited [Nivre, 2002, p. 134].

In the TAG formalism [Joshi, 1975], there are projects on getting open source, wide-coverage grammars for English and Korean, but, to our knowledge, not for Swedish.

The ParGram [Butt et al., 2002] project aims at making wide coverage grammars using the Lexical Functional Grammar approach [Bresnan, 1982]. The grammars are implemented in parallel in order to coordinate the analyses of different languages and there are now grammars for English, German, Japanese and Norwegian.

Chapter 3

Importing Saldo

The lexicon provided with the GF resources is far too small for open-domain parsing. Experiments have been made to use an interactive tool for lexical acquisition, but this should be used for complementing rather than creating the lexicon. This section describes the process of importing Saldo, which is compatible with GF, and easily translated to GF format. As Saldo is continuously updated, the importing process has been designed to be fast and stable enough to be redone at any time.

3.1 Implementation

The basic algorithm for importing Saldo was implemented by Angelov (2008) and produces code for a GF lexicon. For each word in Saldo, it decides which forms should be used as input to the GF smart paradigms. For a verb, this will in most cases mean giving the present tense form, see figure 3.1.

```
mkV "knyter" ;
```

Figure 3.1: First code produced for the verb 'knyta' (*'tie'*)

All assumed paradigms are printed to a temporary lexicon, which will produce an inflection table for every entry when compiled. The tables are compared to the information given in Saldo and if the tables are equal the code for the word is saved. If the table is erroneous, another try is made by giving more forms to the smart paradigm. For example 3.1, the smart paradigm will fail to calculate the correct inflection table. In the next try both the present and the past tense are given:

```
mkV "knyter" "knöt" ;
```

Figure 3.2: Second output for the verb 'knyta'

The program is run iteratively until the GF table matches the one given in Saldo, or until there are no more ways of using the smart paradigm. The verb 'knyta' will need three forms:

```
mkV "knyter" "knöt" "knutit"
```

Figure 3.3: Final output for the verb 'knyta'

Figure 3.4 shows the information given by Saldo and by GF respectively.
The tables does not overlap completely, there are some more forms in Saldo's table (e.g.

grundform	växa
mönster	vb_va_växa
ordklass	vb
böjningstabell	
pres ind aktiv	växer
pres konj aktiv	växe
pret ind aktiv	växte
pret konj aktiv	vuxe
imper	växa
inf aktiv	växa
sup aktiv	växt/vuxit
pres_part nom	växande
pres_part gen	växandes
pret_part indef sg u nom	vuxen
pret_part indef sg u gen	vuxens
pret_part indef sg n nom	vuxet
pret_part indef sg n gen	vuxets
pret_part indef pl nom	vuxna
pret_part indef pl gen	vuxnas
pret_part def sg no_masc nom	vuxna
pret_part def sg no_masc gen	vuxnas
pret_part def sg masc nom	vuxne
pret_part def sg masc gen	vuxnes
pret_part def pl nom	vuxna
pret_part def pl gen	vuxnas
c	väx-/väx
sms	väx-

(a) Saldo

```
s (VF (VPres Act)) : växer
s (VF (VPres Pass)) : växas
s (VF (VPret Act)) : växte
s (VF (VPret Pass)) : växtes
s (VF (VImper Act)) : växa
s (VF (VImper Pass)) : växas
s (VI (VInfin Act)) : växa
s (VI (VInfin Pass)) : växas
s (VI (VSupin Act)) : vuxit
s (VI (VSupin Pass)) : vuxits
s (VI (VPtPret (Strong (GSg Utr)) Nom)) : vuxen
s (VI (VPtPret (Strong (GSg Utr)) Gen)) : vuxens
s (VI (VPtPret (Strong (GSg Neutr)) Nom)) : vuxet
s (VI (VPtPret (Strong (GSg Neutr)) Gen)) : vuxets
s (VI (VPtPret (Strong GPl) Nom)) : vuxna
s (VI (VPtPret (Strong GPl) Gen)) : vuxnas
s (VI (VPtPret (Weak Sg) Nom)) : vuxna
s (VI (VPtPret (Weak Sg) Gen)) : vuxnas
s (VI (VPtPret (Weak Pl) Nom)) : vuxna
s (VI (VPtPret (Weak Pl) Gen)) : vuxnas
```

(b) GF

Figure 3.4: Inflection table for the verb 'växa' (grow)

the compounding form 'väx-') while the one generated in GF contains some that are not in Saldo (e.g. 'vuxits'). As GF concerns about syntax only, and not semantics, and as the GF tables are automatically generated, they always contain all word forms, although some forms may never be used in the natural language. Saldo may also contain variants: 'växt' and 'vuxit' are both supine form. As far as possible, the program makes up for this by only comparing the overlapping forms and only requiring that GF generates one variant whenever alternatives are given.

During this project, the program has been made more robust than the previous version. It also prints log files providing information about the process of importing each word: which paradigms that have been tried, the results from the comparisons of the inflection tables and finally listings of the words that could not be imported.

Each entry in Saldo has an identifier, e.g. `äta..vb`, which is used as constant names in the GF lexicon. However, the identifier may need some renaming since there are special characters in the Saldo identifiers that should be avoided in GF function names. The importation therefore needed some renaming. The Swedish letters *å,ä,ö* are translated into *aa,ae,oe* respectively.

```
äta..vb  -> aeta_V
```

This translation may cause two or more lemmas to share the same GF identifier, and to avoid name collision a number is added to the name whenever a translation have been done:

kältisk	→	kaeltisk_1
kaeltisk	→	kaeltisk
entrecôte	→	entrecoote_1

3.2 Results

The resulting dictionary contains more than 100 000 entries, approximately 80 % of the total size of Saldo. There are a number of reasons why some words were not imported, the most obvious one is that we do not want all categories from Saldo in the GF lexicon. Prepositions, numerals, personal pronouns etc. are assumed to be present in the resource grammars and should not be added again. Saldo contains many pronouns which are not analysed the same way in GF (see section 4.1.1). Before adding them to our lexicon, we need to do more analysing to find their correct GF-category. Some experiments on finding the category have been done using Talbanken, see section 68.

Categories involving multiple words are usually handled as idioms and should be given in a separate lexicon. In total six types of words were considered for the extraction:

	Saldo tag	GF category	Example
Adverb	**ab**	**Adv**	ofta (*often*)
Adjective	**av**	**A**	gul (*yellow*)
Noun	**nn**	**N**	hus (*house*)
Verb	**vb**	**V**	springa (*run*)
Reflexive verbs	**vbm**	**V**	raka sig (*shave*)
Verbs with particles	**vbm**	**V**	piggna till (*perk up*)

Figure 3.5

Most but not all words of these categories have been imported. One reason why the importing phase would fail is that Saldo, unlike GF, only contains the actually used word forms. For technical reasons, the smart paradigm might need forms never used. Consider for example the plural tantum noun *'glasögon'* (*'glasses'*). The smart paradigm requires a singular form, and since the program could not find this in Saldo, there was no way of adding the lemma to the lexicon. When the program failed to import a noun, this was often the explanation. Words of this type may be added manually, for *'glasögon'* we could use the ostensibly correct singular form 'glasöga', although this has another meaning ('glass-eye'). The same problem occurred for the irregular s-verbs, (*'synas' ('show')* or *umgås ('socialize')*) which made up 61.5 % of the failing verbs of type `vb`.

In a few cases the smart paradigms could not generate the correct declination.

When testing the coverage of Talbanken, we found that there are around 2500 word forms still missing, excluding the ones tagged as names and numbers. This number may seem very high, but 4/5 of the word forms are compounds and when performing the intended parsing, an additional analysis identifying compounds should be preformed before looking-up the words in the lexicon. We should also take into consideration that we cannot automatically find out how many actually stem from the same word, or how many abbreviations that are present. Talbanken also contains a small number of spelling errors, which probably are enumerated among our missing words. The majority of the missing words are only used once.

Missing words	$\sim$ 2500 word-forms
Missing words, ignoring compounds	$\sim$ 500 word-forms
Missing words used more than once	$\sim$ 500 word forms
Missing words used more than once, ignoring compounds	$\sim$ 150 word-forms

Figure 3.6

A list of words that were given different labels in GF than in Talbanken has been composed, consisting of about 1600 entries. Many of those are acceptable and reflects the difference made in the analyses, such as the examples in table 3.7. Others are examples of words that are still missing from the lexicon.

Word	Talbanken tag	GF category
måste	MVPS	VV
allting	POTP	N
få	POZP	Det

Figure 3.7

Valency information, which is crucial for GF, is not given in Saldo and hence not in the imported lexicon. It remains as future work to find methods to extract this information from Talbanken and to automatically build it into the lexicon.

Chapter 4

The grammar

An important part of this project has been to develop the Swedish GF grammar and to adapt it to cover constructions used in Talbanken. As a grammar implementation can never be expected to give full coverage of a language, we aim for a grammar fragment which gives a deep analysis of the most important Swedish constructions. The starting point has been the GF resource grammar and the new implementation is still compatible with this. Before describing the actual implementation in section 4.2, we will give an introduction to the resource grammars in general and to the Swedish implementation in particular.

4.1 The Swedish resource grammar

The GF resource grammars gives a fundamental description of Swedish, covering the morphology, word order, agreement, tense, basic conjunction etc. Due to the syntactic similarities of the Scandinavian languages, much of the implementation is shared with Norwegian and Danish. The modules that concern the lexical aspects are separate, while 85 % of the syntax description is shared. There are about 80 functions, which describe the rules for building phrases and clauses.

```
PredVP       : NP -> VP -> Cl ; -- Predication
ComplVPSlash : VPSlash  -> VP ; -- Complementation
```

The analysis preformed by GF is driven by the parts of speech, which are combined into parts of sentences. Figure 4.1 shows the different categories, or types, used in the resource grammars. Words from the open word-classes are shown in rectangular boxes in the picture. Each lexical entry is assigned a type describing its word-class.

The concrete grammar gives a *linearization type* for each category, usually a record containing a table with the forms the word or phrase can be used in. They may also carry information about inherent features that matter to other parts of the sentence.

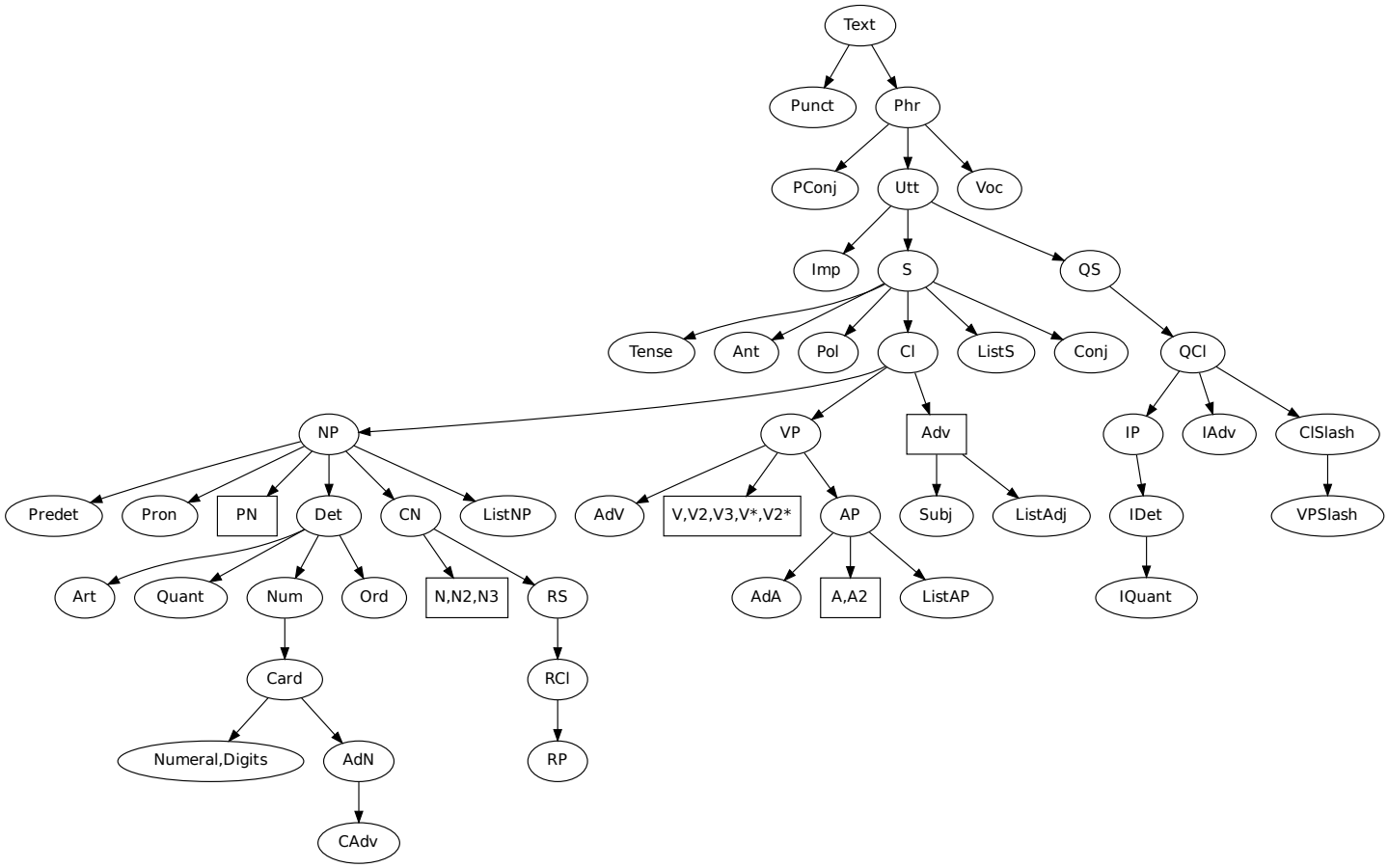


Figure 4.1: The types of the GF resource grammars.

4.1.1 Noun phrases

Section 2.1.1 contained an easy example with nouns. We used the categories `N` and `N'` to distinguish nouns with respectively without number information. The category `N'` was a simplification and is not used in the resource grammar. The full analysis includes a distinction between the categories `N` – simple nouns – and `CN` – common noun phrases – and `NP` – noun phrases. The category `N` is implemented as an inflection table, generated by the smart paradigm, and a field keeping information about the gender, see figure 4.2.
Nouns may turned into common noun phrases `CN`, which may be modified by adjectival phrases or conjoined.

```
AdjCN : AP -> CN -> CN ;
```

The function `DetCN`, determination, creates a `NP` by setting the number and definiteness of a `CN`, shown in figure 4.3.

```
flicka_N = {s = {Sg Indef Nom =>  flicka
                 Sg Indef Gen =>  flickas
                 Sg Def Nom   => flickan
                 Sg Def Gen   => flickans
                 Pl Indef Nom => flickor
                 Pl Indef Gen => flickors
                 Pl Def Nom   => flickorna
                 Pl Def Gen   => flickornas ;
            g = utr }
```

Figure 4.2: Representation of the noun 'flicka' (*'girl'*) in GF

```
DetCN : Det                    -> CN     -> NP ;
        indefinite, singular -> flicka -> "en flicka"
        definitive, singular -> flicka -> "flickan"
```

Figure 4.3

Some rules for determination were described in section 2.4.1; if we have a common noun phrase consisting of the parts *'liten'* (*'small'*) and *'katt'* (*'cat'*), there are three ways they can be combined, as shown in figure 4.4. The noun may be used in definite or indefinite form and the adjective in its weak or strong form [Josefsson, 2001, p. 31].

Determiner	**Adjective**	**Noun**	**DetSpecies**
en	liten [-DEF]	katt [-DET]	**DIndef**
min	lilla [+DEF]	katt [-DET]	**DDef Indef**
den	lilla [+DEF]	katten [+DET]	**DDef Def**

Figure 4.4

Hence, all determiners in our grammar must keep information about which definiteness they require; the `DetSpecies` parameter is stored as an inherent feature of the determiner. The resource grammar distinguishes between quantifiers (`Quant`), determiners (`Det`) and predeterminers (`Predet`). Predeterminers modify `NP`s while the other to modifies `CN`s. The differences are further shown in table 4.5.

The definite article is considered to be a quantifier, which has the forms *en* and *ett* for singular. In plural it is the either *'de'* or nothing, cf. sentence (17a) and 17b.

(17) a. Katten sov. b. **Den** gamla katten sov.
cat+DEF *slept* *the* *old* *cat*+DEF *slept*

In order to get the correct plural form we need to know if the common noun phrase has been modified, that is, whether the function `AdjCN` has been used. However, once a category has been formed, there is no longer any information available about how it was put together. This is a result of the functional approach of GF. Therefore, this information has to be passed on by an inherent feature of the `CN`, a flag set to tell if the `AdjCN` was applied.

	Has number	Has definiteness	Example
Predeterminers	–	–	alla katter, all maten
Quantifiers	–	✓	min katt, mina katter, *min katten
Determiners	✓	✓	varje katt, *varje katten, *varje katter

Figure 4.5

```
DetNP : Det              -> CN        -> NP ;
        definite, plural +  katt      =  katter
        definite, plural +  stor katt =  de stora katterna
```

Figure 4.6

Due to the syntax oriented analysis in GF, the GF category for pronouns `PN` only contains personal pronouns. Many words, like 'somliga' (*'some'*), are considered to be pronouns in other analyses, such as The Swedish Academy Grammar [Teleman et al., 1999], but are classified differently in GF, usually as determiners or quantifiers as they may determine noun phrases (18).

(18) Somliga studenter jobbar bara på nätterna
"Some student only work at night time"

4.1.2 Verb phrases

In figure 2.2 we saw the GF representation of the sentence "Jag ser katten". The verb 'ser' (*'see'*) takes one object: 'katten'. In GF, this can be seen on the type of the verb, valency information is encoded into the lexicon. Transitive verbs have the type `V2` and ditransitive `V3`. There are types for verb which takes sentences (`VS`) and adjectival (`VA`) complements. The lexicon is therefore a very important part of the grammar.

```
see_V2 ;
SlashV2a   : V2 -> VPSlash ;
ComplSlash : VPSlash -> NP -> VP ;
```

The function `SlashV2a` lets you create a `VPSlash`. The name `SlashV2a` is inspired by categorical grammar and is a short hand for `VP \ NP` – a verb phrase missing a noun phrase.. When we combine the `SlashVP` with the object we get a complete verb phrase. Both the verb and its complements are stored in the verb phrase category, `VP`. The `VP` category resembles the Didrichsen's field model. There are fields for negation, different types of adverbs, objects, the finite verb and an infinite verb. The fields may in turn be tables showing the different forms of the component.

	VP							
VP field		FINIT	NEG	ADV	INF	COMP	OBJ	ADV
	(han)	har	inte	alltid	tänkt	på	henne	så

The verb phrase fields are not put in their correct order until the tense and type of clause is determined, ie. when a sentence is created.

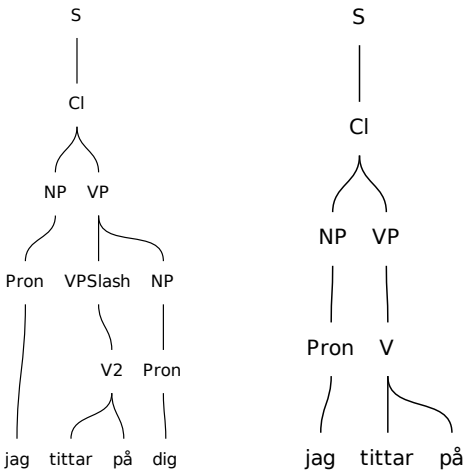
Swedish verbs may take up to five arguments [Stymne, 2006, p. 53]. These may be prepositions, particles, reflexive object, indirect objects and direct objects.

(19) Jag tar med mig den till honom
I take with me it to him
"I bring it to him"

The verb 'ta' in sentence (19) takes one particle (*'med'*), one preposition (*'till'*) and two objects: *'den'* and *'honom'*. In a GF lexicon this verb is given the category `V3`, a *three-place verb*, taking two objects. The notion `V3` is motivated by the formal translation: `bring(I,it,him)`.
Particles are given in the lexicon, as well as the prepositions that are chosen by the verb, since these may not vary. The entry for '*ta med*', as used in sentence (19), is described as follows:

```
ta_med_V3 = dirV3 (reflV (partV (take_V "med"))) (mkPrep "till") ;
```

The function `dirV3` creates a three-place verb, where the first object is direct and the second is to be used with the preposition given as the last argument: '*till*'. `reflV` shows that the verb always is used with a reflexive pronoun and `partV` gives the particle *'med'*. The fact that the chosen prepositions is attached to the verb in the lexicon causes the parse tree visualization algorithm to group them together. This is also the case for particles, cf. parse tree 4.7b and 4.7a.



(a) Verb with a chosen preposition

(b) Verb with particle

Figure 4.7: The visualized parse trees do not show the internal difference of chosen prepositions and particles

As already stated, the visualized parse trees is not a complete representation, even if the verb phrases in the visualizations look the same, the two cases are treated and represented differently internally. The fronting of the preposition, as in of sentence (20a)., is accepted but fronting of particles, as in 20b., is not.

(20) a På pojken tittar du.
on the boy look you
"You look at the boy."

b *På när du springer tittar jag
on when you run watch I

4.1.3 Clauses

The category clause, `Cl`, represents a pre-sentence, that does not yet have any tense, polarity or word order set, see figure 4.8.

Tense	**Polarity**	**Word order**	**Cl**	**S**
present	negative	main	du ser mig	'Du ser inte mig'
perfect	positive	inverted	du ser mig	'Har du sett mig'
perfect	negative	subordinate	du ser mig	'Du har sett mig'

Figure 4.8

Like verb phrases, a clause may also be missing an object and then has the type `ClSlash`. The `ClSlash` is formed by a `VPSlash` which is given a subject. This is a convenient way to form questions, relative clauses and topicalized clauses (see figure 4.9), as introduced in [Gazdar, 1981].

Cl	Johan + tittar på	
Wh-Questions	Interrogative pronoun	'Vad tittar Johan på?'
Relative clauses	Object + Relative pronoun	'Katten som Johan tittar på'
Topicalized clauses	Object	'Henne tittar Johan på'

Figure 4.9

4.1.4 Overview

The original resource grammar could express complex sentences as the one in figure 4.10. Even though the verb phrase *"har inte ätit de gula äpplena idag"* is discontinuous, the whole phrase is still treated as one constituent in GF. The parts are connected in the tree, and the subject *'han'* is put between the finite verb and the rest of the phrase. At the code level, this is done using the record type for `VP`, which consists of fields that can be put in different order.

```
table { Inv => verb.fin ++ subj ++ verb.neg ++ verb.inf ++ verb.compl ; ...
```

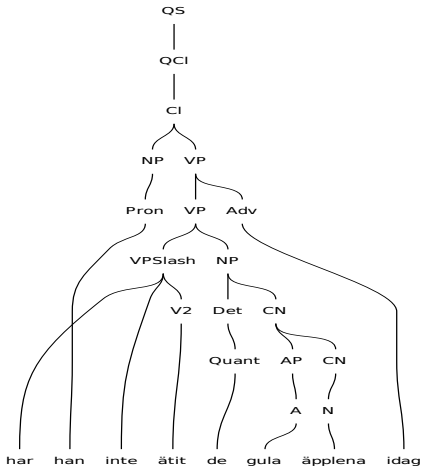


Figure 4.10: Parse tree for "Har han inte ätit de gula äpplena idag?"

The resource grammar also covered relative clauses:

(21) a Hon ser pojken som sover
she sees the boy that sleeps

b Han ser katten han tycker om
he sees the cat he likes

In addition to the core resource grammars, which is shared with the other languages implemented in the library, there is also an extra module, simple called the `Extra` module, where language specific constructions may bee added The functions given here do not have to be translatable to all other language, but are meant to cover language specific constructions. Among those were functions for topicalisation (22),

(22) Det där äpplet vill jag inte ha
that apple want I not have
"I don't want that apple"

and for preposition stranding:

(23) a. Stranded preposition
Vem måste jag akta mig **för**?
who must I watch out me ***for***?

b. cf.
För vem måste jag akta mig?
for *who must I watch out me*?
'Who do I need to watch out **for**?' '**For** whom do I need to watch out?'

4.2 Development of the grammar

It has earlier been hard to identify missing constructions of the Swedish implementation, since there was no large resource available to evaluate it on.

Our evaluations are based on Talbanken, and when first conducting tests, we found much room for improvement. From the topics listed in section 2.4, the post-nominal articles and the verb-second property were covered by the resource grammar, as well as the periphrastic passive and a limited form of topicalisation. The other constructions have been added during this project.

4.2.1 The s-passive

Passive voice is often used in Swedish, especially the *s-passive*.

(24) Uppsatsen skrev**s** av en student.
*the essay wrote+**s** by a student*
"The essay was written by a student."

Some studies suggest that the s-passive is used in more than 80 % of the times [Laanemets, 2009]. It is however not as common in the other Scandinavian languages, where not all words have passive forms for all tenses. The Norwegian translation of sentence (24) is:

(25) Oppgaven ble skrevet av en student [NO]
uppsatsen blev skriven av en student [SE]

The corresponding Swedish sentence is acceptable, but not as natural sounding as sentence (24). The resource grammar for Scandinavian therefore implemented the function for passive, `PassV2`, by using auxiliary verb.

```
PassV2 : V2  -> VP ;
         ta -> blev tagen
```

The function allows two-place verbs to be used in passive by using *bli* (*become*), and thereby turned into complete verb phrases; they no longer need an object.

During this project, the s-passive was added although the periphrastic passive is still allowed. The grammar further allows not only V2, but all verb phrases that misses an object, to form passives:

```
PassVP : VPSlash -> VP ;
         ta      -> togs
         erbjöd  -> erbjöds
```

A `V3` like 'give' in sentence (26) hence gives rise to two passives, (28) and (27).

(26) **Active use of two-place verb**
Vi erbjöd henne jobbet
we offered her the job
"We offered her the job"

(27) **First place in two-place verbs**
Hon erbjöds jobbet
*she offered+**s** the job*
"She was offered the job"

(28) **Second place in two-place verbs**
Jobbet erbjöds henne
*the job offered+**s** her*
"The job was offered to her"

(29) **V2A verb**
Huset målades rött
*the house painted+**s** red*
"The house was painted red"

4.2.2 Impersonal constructions

Formal subjects [Teleman et al., 1999, §19] are often used in Swedish.

(30) Det sitter en fågel på taket
it sits a bird on the roof
"There is a bird sitting on the roof"

'Det' has the position of the subject, and the real subject, *'en fågel'* the one of an object.
Transitive verbs may not be used like this

(31) *Det äter en fågel frön på taket
it eats a bird seeds on the roof

unless their in passive form

(32) Det dricks mycket öl nuförtiden
It drinks+s much beer nowadays
"A lot of beer is being drunk these days"

A very common example of this is sentences with the verbs *finnas*, (*exist*) *saknas* (*miss*) and *fattas*(*lack*).

(33) a. Det finns kaffe b. Det saknas kaffe c. Det fattas kaffe
it exist coffee *it misses coffee* *it lacks coffee*
"There is coffee" "There is no coffee" "There is no coffee"

To implement this construction, we needed a special GF category, `SimpleVP`, to exclude other verb phrases like the one in sentence (31). Any intransitive verb can form a `SimpleVP`, as can transitive verbs without their objects. The `SimpleVP` may further be modified by adverbs.

There are also restrictions on the real subject, which is not allowed to be in definite form.

(34) *Det sitter **den fågeln** på taket.
it sits the bird on the roof

Neither is sentence (35) correct.

(35) *Det sitter **min fågel** på taket
it sits my bird on the roof

In order to implement this, we utilized the different types of determiners shown in figure 4.4. For postverbal subjects, the determiner must be of type `DIndef`, that requires both the noun and the adjective to be indefinite. To form a clause with formal subject, we hence need to inspect the determiner.

```
FormalSub : SimpleVP -> Det -> CN -> Cl ;
```

The function combines a verb phrase, a determiner and a noun phrase to a clause. If the determiner does not fulfill the requirements stated above, the clause is put to NONEXIST. This works well for parsing, but leads to problems if the grammar is used for random generation. The solution is thus not ideal, but since the definiteness of noun phrases or determiners cannot be seen on the type level, it is not known until runtime whether the determiner is accepted in the subject.

As a future direction it would be interesting to examine the consequences of letting the noun phrases have more information on the type level. In the implementation for reflexive objects (see section 4.2.3), dependent types are used for showing if a noun phrase needs an antecedent. We would also like to differentiate between the `NP`s in sentence (36a,b), where '*av*' should be used only when the noun has an explicit article or determiner.

(36) a. De flesta hästarna b. De flesta **av** de där hästarna.
"Most of the horses" b. "Most of those horses."

4.2.3 Formalizing the rules for reflexive pronouns by using dependent types

An important area in a Swedish grammar is the treatment of the reflexive pronouns and the reflexive possessive pronouns as described in section 2.4.5 The reflexives require an antecedent with which they agree in number and person. Our grammar should accept the following sentences:

(37) Han såg sina barns skor.
He saw SELF'S children's shoes.

(38) Sina vantar hade han glömt på tåget.
SELF'S gloves, he had forgotten on the train.

(39) Hon ber alla sina kompisar att gå
She asks all SELF'S friends to leave

(40) Jag vill att han rakar sig.
I want him to shave SELF.

(41) a. Han är längre än sin kompis. b. Han är här oftare än sin kompis.
He is taller than SELF'S friend. He is here more often than SELF'S friend.

(42) a. Hon tyckte om skolan och alla sina elever.
She liked the school and all SELF'S students.

b. Han såg sina få böcker och sin penna.
He saw SELF'S few books and SELF'S pencil.

Reflexive pronouns can not be used in subject noun phrases of finite sentences, as shown by the ungrammatical examples in sentence (46) and (43). The third person reflexives ('sig','sin') requires a third person antecedent (see 45). Furthermore, the antecedent must be within the same finite sentence as the reflexive pronoun, see (46). The grammar should not accept any of these sentences:

(43) *Sina vantar var kvar på tåget.
SELF'S gloves were left on the train.

(44) *Han och sin kompis läser en bok.
He and SELF'S friend are reading a book.

(45) *Jag ger sina pengar till sina syskon.
I give SELF'S money to SELF'S siblings.

(46) *Han vill att jag ser på sig.
He wants me to look at SELF.

Apart from these restrictions, noun phrases containing reflexive pronouns may be used as any other `NP`. They may be conjoined (42 a,b) and used with other determiners (39). In the standard GF analysis, which is preformed bottom-up starting from the POS-tags, information about semantic roles are given by the functions, not by the categories. That is, we know that the first argument of the function `PredVP` acts as the subject, but the noun phrase itself does not carry information about its semantic role. Until it is given as an argument to a clause level function, no difference is made between subject and object noun phrases. For this reason, the formalization of reflexive pronouns required the use of a different analysis. In short, what is wanted can be summarized as follows:

1. Construct a noun phrase from a common noun phrase
katt → sin katt
2. Modify it like a `NP`
sina katter → alla sina katter
3. Use it only as object
sin katt → han såg sin katt

If we ignore the second requirement, we might come up with a solution where special functions for using common nouns together with reflexive possessive pronouns are introduced:

```
ReflVP: CN -> VPSlash -> VP ;
        mat + äta     -> äta sin mat
```

This way, using the phrases as subject is excluded, but none of sentence (42 a,b) or (39) are allowed.
The simplest way to fix this is to treat the phrases as normal NPs. We can add a rule that allows using the possessives with common noun phrases:

```
ReflNP: CN -> NP ;
        mat -> sin mat
```

They may now be used with predeterminers:

```
PredetNP : Predet -> NP       -> NP ;
           alla   + sina barn -> alla sina barn
```

But by keeping them as NPs we lose information, and there is no way of keeping them from being used as subjects. We have ignored the third restriction and the grammar would generate and accept sentence (43).

We get closer to a satisfying solution by introducing new object type, Obj, which is identical to normal NP except that it depends on the antecedent. All functions where objects are modified in the same way as other NPs are duplicated, one version is dealing with objects and one with NP.

```
ReflVP: Obj     -> VPSlash -> VP ;
     -- sin mat +  äta     -> äta sin mat
NPtoObj : NP  -> Obj ;
PredVP  : NP -> VP -> Cl ;  -- NP used as subject
```

Any noun phrase may also be used as an object, whereas subjects only can be made up by NPs. The drawback of this solution is the duplication of functions.

```
PredetNP  : Det -> NP  -> NP ;
PredetObj : Det -> Obj -> Obj ;
```

The dependence on the antecedent needs to be applied to adverbial phrases and adjective phrases as well. The adjective phrase in (41a) and the adverbial phrase in (41b) are examples of this; if the subject was changed to 2nd person, they would need to change too.

(47) a. **Du** är längre än **din** kompis.
"You are taller than your friend."

b. *__Du__ är här oftare än **sin** kompis.
"You are here more often than SELF'S friend."

Adverbs containing reflexive possessive pronouns may further not be used to construct subject noun phrases.

(48) *Taket på sitt hus läcker
The roof on SELF'S house leaks
(NP +(Adv (NP Obj)))

The dependency spreads through the grammar and in order to avoid all code duplication we use another approach. When looking at the type of the functions, we notice that they can be generalized:

```
PredetNP  : Det -> NP x  -> NP x;
```

The solution chosen in this project is to make use of this generalization and introduce the use of dependent types in a resource grammar. Following the idea given above, we make a difference between subjects and objects, but not by giving them entirely different types, but by letting the type `NP` *depend* on an argument, which may either be `Subject` or `Object`.

```
cat NP NPType ;
PredVP     : NP Obj  -> VP     -> Cl ;
ComplSlash : VPSlash -> NP Obj -> VP ;
PredetNP   : (a : NPType) -> Det -> NP a -> NP a ;
```

The types for adverbial and adjectival phrases are also turned into dependent types.

(49) taket på sitt hus
NP a + (Adv Obj) ⇒ NP Obj
the roof of SELF'S *house*

(50) sitt hus på berget
NP Obj + Adv a ⇒ NP Obj
SELF'S *house on the mountain*

We hence combine the-part-of-speech driven analyse normally preformed by GF with a part-of-sentence analysis, where the dependent types gives the information we were missing.

The use of dependent types separates our grammar from the other resource grammars. Such separation is not desirable in itself, but there are at least two reasons why we believe it is appropriate in this case:

- The new grammar can be made compatible with the old implementation.
- The possibility for noun phrases to agree with an antecedent will be useful in other languages. For example, the Slavic and Romance languages have similar reflexive pronouns.

Moreover, the goal of the project is to make a language specific grammar, whereas the common abstract is developed to be generalizable enough to describe any language. It thus would be surprising if we could cover all syntactical aspects of Swedish while leaving the abstract grammar unchanged.

Dependent types are also interesting for other problems. The rules for reciprocal pronouns could be formalized using the same idea. As reflexive pronouns, reciprocals (sentence 51) must not be used as subjects and furthermore they may only be used when the number of the antecedent is plural.

(51) De ser varandra
they see each other
"They see each other"

4.2.4 A second future tense: "Kommer att"

Swedish and Norwegian use two auxiliary verbs for future tense [Holmes and Hinchcliff, 1994, p. 246]:

(52) a. Det **kommer att** bli mörkt snart
"It will get dark soon"
b. Jag **ska** gå och lägga mig nu
"I'm going to bed now"

(53) Jeg kommer til å savne deg [NO]
"I will miss you"

The modal verb *'ska'* signals an intention of committing the action, either from the subject or from the speaker. Cf.

(54) a. Du ska tycka om den
"You *shall* like it"
b. Du kommer tycka om den
"You will like it."

The verb *'kommer'* (*'come'*), normally used with the infinite marker *att*, does not signal any intention, but that the speaker has belief that it will actually come true.
The resource grammar included *'ska'*, which was implemented as the standard way of forming future tense, and hence represents the translation of *'will'*. The new grammar also supports *"kommer att"*, expressed as an alternative future tense.

```
(UseCl (TTAnt TFutKommer ASimul) PPos
       (ImpersCl (AdvVP (ComplVA become_VA (PositA dark_A)) soon_Adv))
```

Figure 4.11: Result of parsing "Det kommer att bli mörkt snart". The future tense is marked by the constant TFutKommer

Since two out of the three Scandinavian languages share this tense, it has been added to the Scandinavian `Extra` module. Table 4.12 shows how the grammar expresses new tense in different types of sentences.

```
s SFutKommer Simul Pos Main : jag kommer att se henne
s SFutKommer Simul Pos Inv  : kommer jag att se henne
s SFutKommer Simul Pos Sub  : jag kommer att se henne
s SFutKommer Simul Neg Main : jag kommer inte att se henne
s SFutKommer Simul Neg Inv  : kommer jag inte att se henne
s SFutKommer Simul Neg Sub  : jag inte kommer att se henne
s SFutKommer Anter Pos Main : jag kommer att ha sett henne
s SFutKommer Anter Pos Inv  : kommer jag att ha sett henne
s SFutKommer Anter Pos Sub  : jag kommer att ha sett henne
s SFutKommer Anter Neg Main : jag kommer inte att ha sett henne
s SFutKommer Anter Neg Inv  : kommer jag inte att ha sett henne
s SFutKommer Anter Neg Sub  : jag inte kommer att ha sett henne
```

Figure 4.12: The covered and accepted usage of the future tense with 'komma'

4.2.5 Modifying verb phrases

Focusing adverbs

The GF analysis distinguishes between two categories of adverbs: `Adv` and `AdV`. The `AdV` – e.g. 'aldrig' (*'never'*) and 'inte' (*'not'*) – attaches directly to the verb.

(55) Cf.
a. Jag äter **aldrig** fisk
I eat never fish
" I never eat fish"
b. Jag äter fisk **nu**
I eat fish now
"I eat fish now"

The difference is implemented by having separate fields in the `VP` table for the two categories.

```
Main clause : subj ++ verb.fin ++ verb.adV ++ verb.inf ++ verb.adv
              han     har         aldrig      varit       här
```

The adverb 'bara' may be used as an `AdV` but also before the finite verb, when emphasizing the verb itself. This is an example of a *focusing adverb*, others examples are 'inte ens' (*'not even'*) and 'till och med' (*'even'*).

(56) a. Han bara log
he only smiled
" He just smiled"
b. Hon till och med visslar
she even whistles
"She even whistles"

Focusing adverbs are accepted by the new grammar implementation, where they have their own field in the `VP`table.

```
Main clause : subj ++ verb.focAdv ++ verb.fin ++ verb.adV ++ verb.inf ++ verb.adv
              han     bara           sover        -             -           -
```

They are also allowed as `AdV`...

(57) Han har bara sovit
he has only slept

...or as predeterminers:

(58) Det är bara barn som får göra så
it is only children that may do so
"Only children are allowed to do that"

Two more rules are consequently needed

```
FocAdvAdV : FocAdv -> AdV ;
PredetAdvF : AdvFoc -> Predet ;
```

The focusing adverbs are usually not combined with modal verbs, the copula or temporal auxiliaries.

(59) a ? Han bara är dum
he only is stupid

b ? Han bara har sovit
he only has slept

We have however chosen to allow this, since it is grammatically correct although semantically dubious, and since the alternative linearization *"Han har bara sovit"* is covered by the rules using focusing adverbs as `AdV`.

4.2.6 Miscellaneous

This section covers some minor changes of the grammar. They are described as documentation of the implementation and to illustrate some problems and solution of more general nature.

Relative clauses

The resource grammar already gave a good coverage of relative clauses and embedded sentences. All constructions used in examples 60a-c were accepted.

(60) a. Pojken, som är blyg, tystnar
"The boy, who is shy, falls silent"

b. Han såg kunden som tyckte om sallad
"He saw the customer who liked salad"

c. Jag tänkte på huset i vilket hon bodde
"I thought about the house in which she lived"

The grammar has been extended to accept a definite article for nouns in indefinite form, whenever the noun phrase is followed by a restrictive relative clause [Holmes and Hinchcliff, 1994, §329]. Talbanken contained several examples where the modified noun is in the indefinite form as in (61).

(61) de uppfattningar som förs fram ...
*the opinions that put+*PASSIVE *forward ...*
"the opinions, that are presented ..."

When no relative clause is present, the definite form with the postnominal definite article must be used, cf. sentence (62).

(62) a. de uppfattningar**na** förs fram
"the opinions are presented"

b. *de uppfattningar förs fram

Apart from this, only corrections have been done, exemplified in this sentence.

(63) Hon sover **som** är bra → Hon sover, **vilket** är bra
"She sleeps, which is good"

As a side note, some complications regarding the function `RelCl` can be pointed out. The English implementation of this construction is *'such that'*, and the Swedish version *'sådan att'* sounds awkward, except when used in of logic and mathematics books.

(64) a. *From the resource grammar*
Jag vill ha en katt sådan att den inte fäller
"I want a cat such that it does not shed"

b. *An alternative formulation*
Jag vill ha en sådan katt som inte fäller
"I want such a cat that it does not shed"

In GF, this relative clause *"en katt sådan att den inte fäller"* is constructed by two constituents: *'katt'* of type `CN` and *"den inte fäller"* of type `RCl` (relative clause), glued together by *"sådan att"*. As a subject *'den'* is of type `NP` and does not depend on any agreement. However, it should agree with its antecedent, *'katt'*.

(65) *Jag vill ha en katt sådan att **det** inte fäller.

The dependent types introduced in section 4.2.3 could probably be useful to solve this problem. However a carefully prepared analysis and motivation is needed, and sentence (65) is still accepted by the grammar.

An example of overgeneration and how it can be solved

The `Extra` module contains a function for constructing genitives.

```
GenNP : NP -> Quant ;
```

It selects the possessive form given in the `NP` and changes its type to `Quantifier`. The `GenNP` function overgenerated when this project started, even though its implementation was not erroneous in itself. The cause of the overgeneration was a combination of two things:

- `Quantifiers` do not have any possessive forms, so when a quantifier is turned into a `NP` (by the functions `DetQuant` and `DetNP`), its genitive field gets the same form as its nominative field.
- `GenNP` introduced a cycle: a `Quant` may be turned into a `NP`, and now a `NP` can be turned into a `Quant`.
  `GenNP : NP -> Quant, DetQuant : Quant -> Det; DetNP : Det -> NP`

We had that a noun phrase, eg *'katten'* could be turned into a quantifier, by using its genitive form, *"kattens'*. This could be used as a determiner,

(66) Det är kattens mat.
"It is the cat's food."

and thus as a noun phrase:

(67) Det är kattens.
"It is the cat's."

The possessive form of this `NP` is now the same as its nominative: *'kattens'*. Hence, the function `GenNP` could be applied any number of times without any change in the output.

The problem can be solved by changing the implementation of the function `DetNP` which is responsible for creating the possessive form of quantifiers and determiners. We let it append an extra *s* to the word, so that applying `GenNP` to *'kattens'* results in *'katten**ss**'*. Even though this generates a form that does not exist, it is perhaps the best approximation to what it would look like, given that it was indeed used in Swedish. Most important is however that this stopped the `GenNP` from introducing unnecessary ambiguities. The number of parse trees for the phrase *"Kattens bils hus"* was reduced from over one hundred to six. Besides, the grammar will allow the actually existing genitive forms of quantifiers:

(68) Somligas skor står kvar i hallen
some's shoes stand left in the hall
"Some peoples shoes are still in the hall"

Using Talbanken to assign correct categorisations

One of the biggest differences between the GF analysis and the one made in Talbanken is related to pronouns. As noted in section 4.1.1 our GF grammar only classifies personal pronouns as pronouns. Many pronouns could not be imported from Saldo, since the correct category could not be identified automatically. Therefore, some experiments were conducted on how to extract this information from Talbanken. We were interested in words which can be used the same ways as 'sådana' (*'such'*) that acts either as a determiner (69a), or as an adjective agreeing with the noun (69b-c).

(69) a. Sådana katter vill jag ha
such cats want I have
"I want such cats'

b. Han är sådan
he is such
"He's like that"

c. Huset är sådan**t**
the house is such
"The house is like that"

By starting from a list containing all pronouns in Saldo and all forms they may occur in, Talbanken was harvested for the ones that both occurred with the determiner tag `DT` and with the present participle tag `SP`. This way we are left with a limited number of words that could easily be analysed and added to the lexicon.

This method is both faster and more reliable than going through the list by hand and categorising them manually by looking up each word or by using introspection.

4.3 Testing

Throughout the project regression testing has been used. Every time the grammar was updated it was tested against a treebank consisting of 155 sentences, and the number of parse trees were compared to a standard. The purpose was first of all to make sure that the coverage was not decreased, but also to make sure that we would notice if new ambiguities or overgenerating functions were introduced. If so, the ambiguities could be removed when possible, and when not, we were at least made aware of them and could decide whether the increase of coverage were worth the increase in ambiguities.

New functions were also tested by random generating and by creating tables showing all forms, such as table 4.12.

Chapter 5

Extracting a GF treebank

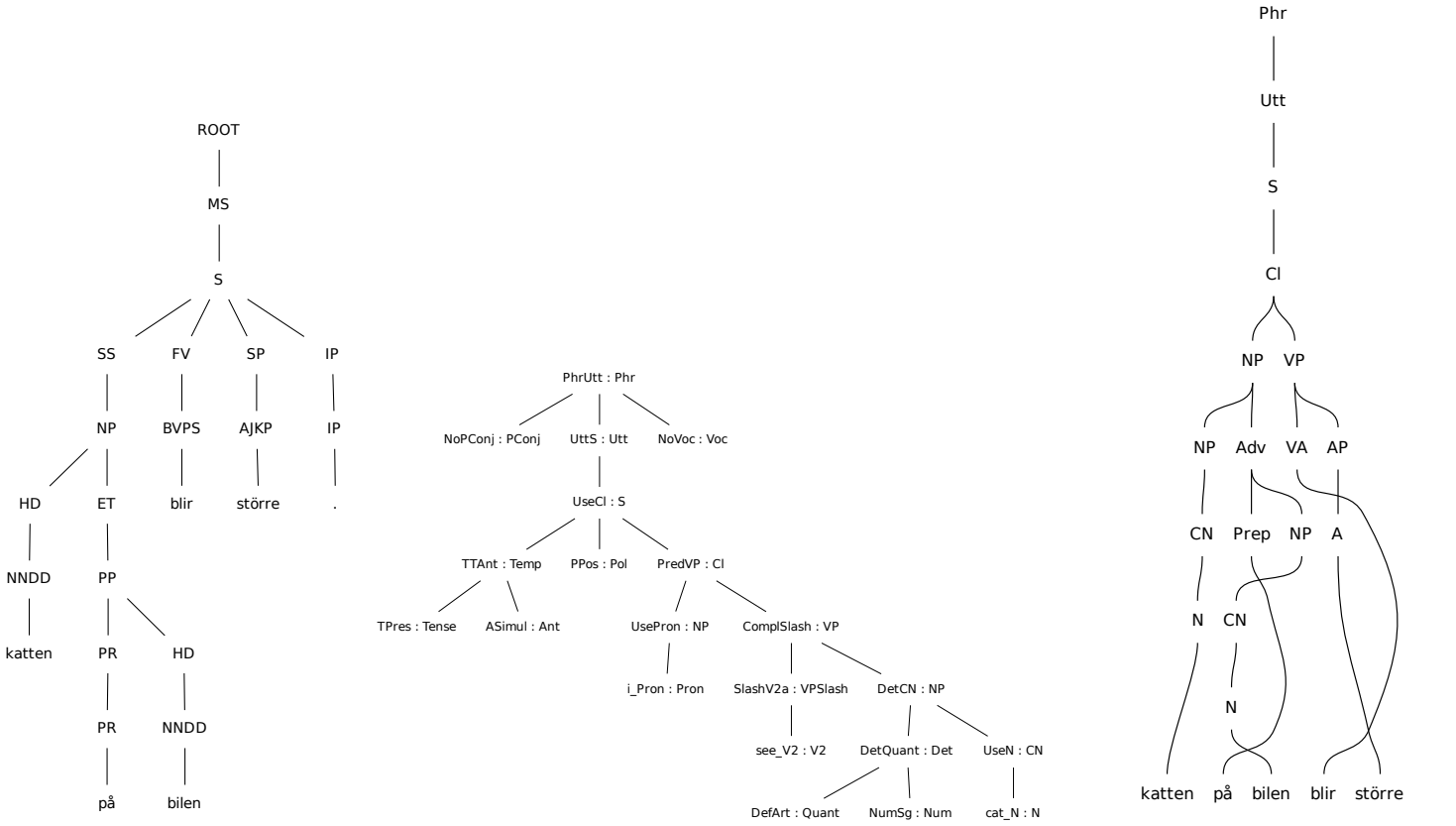
Talbanken contains much valuable information about phrase structures and word usage. It is analyzed with the MAMBA annotation scheme (Telemann, 1974). One part of this project has focused on translating trees from Talbanken to GF by constructing a mapping, which automatically transforms trees in the Talbanken format to GF abstract syntax trees. We hence get a comparison between the two annotations and at the same time we get methods for extracting and translating to GF notation. Figure 5.1 shows an example of a visualized Talbanken05 tree of the sentence "Katten på bilen blir större" (*"The cat on the car gets bigger"*) and the result of translating it to GF. Both the POS tags as well as the syntactic information are needed during the translation, so that all the information shown in the abstract tree (5.1b) can be extracted. We need to know that *katten* is a noun used in definite form (`DefArt`), singular (`NumSg`). Even though the notation is translated, the original analyse from Talbanken is preserved. *på bilen* should still form a subtree attached to *katten*.

The mapping is based on a translation of the English Penn Treebank [Angelov, 2011a]. By modifying this program, we now have a translation that works for the Tiger XML-format of Talbanken. Adaption was required for the differences in annotation as well as for the syntactic differences between Swedish and English.

The translation gives us means to evaluate our parser. By both parsing a Talbanken sentence and transforming its annotated tree, we can easily inspect if the results are equal. Additionally, the mapping shows which grammatical constructions that are still missing from the GF grammar and shows how the GF analysis differs from the one made in Talbanken. If there are words missing from our dictionary, the rich POS-tags may help us to automatically find the correct declination and add it to the lexicon. Further, our parser will need probabilities (see section 6.2) of how often a function is used. The GF treebank we achieve from the translation is a good source for this information.

5.1 The Talbanken annotation

Talbanken05 uses three kinds of tags: categories, edge labels and POS-tags. While the POS-tags are reserved for words, the categories give grammatical information, such as `S`, `NP` and `VP`. The edge labels give the part of sentence: `SS` for subject, `OO` for object etc. Nearly 400 different tags are used, of which more than 300 different are POS-tags. The high number is



(a) Talbanken tree (b) GF abstract tree (c) The corresponding GF parse tree

Figure 5.1: Trees for the sentence “Katten på bilen blir större”.

due to the level of detail, there are for example almost 50 tags for nouns, excluding proper names. The tags show definiteness, case and whether the word is a compound. Some words also have their own tags.

Tag	**Example**	**Explanation**
SV	ska	The verb *shall*
WVIV	vilja	The verb *want* in infinitive
KVSP	kommit att	The future auxiliary verb *kommer* in supine
GVPT PA	gjordes	The verb *do, make*, present, passive
NNDDHSGG	familjemedlemmarnas	Noun, definite, compound (person), genitive
PU	* 1) a)	List item

5.2 The translation process

The implementation consists of a set of rules that translate nodes of a Talbanken tree to GF functions, possibly by describing what the nodes subtree should look like.

```
translate S  = do
             np <- find NP
             vp <- find VP
             return (PredVP np vp)
```

This rule tells that we may map the category `S` to the function `PredVP`, given that we find a noun phrase and a verb phrase in the subtree of `S`.

At every leaf a word is given. It is looked-up in the GF lexicon and the result compared to the information of the POS-tag. If the word is not found in the lexicon a *meta variable* will be put in its place, denoted '`?`'. The meta variable is also used as a function, needed when we have found two translatable trees but no rule that allows them to be combined.

In figure 5.1a the verb *'bli'* is followed by a predicative adjective modifying the subject tagged with `SP`. Our GF grammar admits *'bli'* to have type `VA`, a verb with an adjective as complement, and the translator can hence apply the GF function `ComplVA` to combine the verb with its object.

The translation covers complex noun phrases which may consist of pronouns, reflexive objects or common nouns and contain predeterminers, determiners, quantifiers and adjectival modifiers. Special rules were needed for each verb category in order to find the right number of arguments. The mapping also covers different word order, such as questions and topicalisation.

5.2.1 Differences in the notation

Some structural differences between GF and Talbanken became obvious during the translation. In Talbanken the valency information is given implicitly by the complements of a word. If a verb is followed by an object, `OO`, which contains an `S`, we can conclude that this is a sentence-complement verb. In GF, the valency information is given for each entry in the lexicon. A sentence-complement verb has the type `VS` and from this we know that the function `ComplVS` should be used to combine the verb and with a sentence.

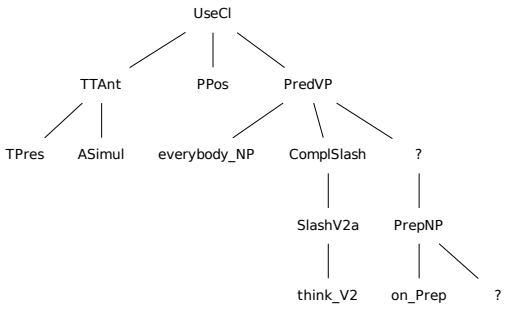


Figure 5.2: Abstract tree with meta variable

A similar issue arises from the use of verb particles. A verb with a particle is treated as two independent units in Talbanken but in GF they are considered to be one constituent. The particle has therefore no effect on the abstract tree, its presence is announced by the verb itself.

When constructing the abstract trees, we must of course follow and use the GF analysis. However, the mapping cannot depend on the valencies given in GF only,

since the distinction between different types of verbs is not always clear in Swedish; most transitive verbs can also be used as intransitive. Both sentences in example (70) are grammatically correct but only (70b) is accepted by the GF grammar: the grammar rules are simply too strict for accepting many Talbanken sentences.

(70) a. Han sitter och läser. b. Han sitter och läser en bok.
he sits and reads *he sits and reads a book*

Another annotational difference that turned out to be problematic for the translation can be illustrated by sentence (71), containing the generic pronoun *'man'*.

(71) Man uppskattar dem
one appreciates them
"They are appreciated"

In Talbanken, *'man'* is simply a personal pronoun, in GF it is represented by using the function `GenericCl`, see figure 5.3. `GenericCl` turns a `VP` into a clause and has no subject visible in the abstract tree. When translating a sentence like this, it is thus not possible to

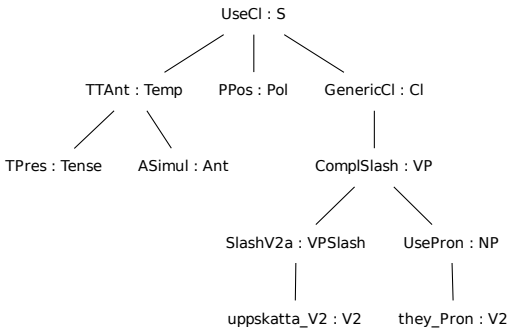


Figure 5.3: Abstract tree for "Man uppskattar dem".

simply glue the subject to the verb with `PredVP`. For each similar GF construction, an extra rule would be needed to get full coverage.

5.3 Results and evaluation

The development was mostly example-driven, and at least one rule for the translation of every Talbanken-tag has been implemented. Shorter sentences have been prioritized in order to get a coverage of the most fundamental constructions. Regression testing was used to ensure that a new rule would not violate the old ones or allow trees to be translated erroneously.

The flat version of Talbanken05 was used for developing the mapping, but the program works for both the flat and the deep Tiger-XML versions. The deeper contains six more tags, although they are not always useful for our purpose. Figure 5.4 shows the difference

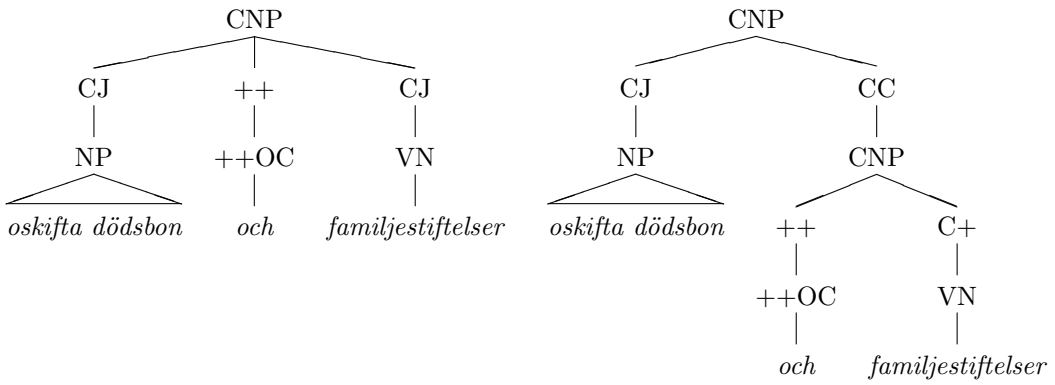


Figure 5.4: An example of the difference in the two analyses

for a conjunction. The focus on the mapping has been simple sentences, in order to cover the most fundamental structures. If the program is extended and adapted to cover complex conjunction, a deeper evaluation should be preformed to see which version that suits our needs the best. The deeper implementation also gives some more information about verb phrases, by the tag `VG` (*verb group*), which groups an infinitival verb and its object together into a `VP`. This information can however be extracted from the flat implementation, and the results get slightly better when using this version.

When evaluating the mapping, the results strongly depend on which restrictions we put on the input. One of the reasons why a node cannot be translated, is the use of the tags show in figure 5.5. The **PU** tag is used for graphic listings, and not for fluent text. In our grammar there is naturally no corresponding function; the listings are meant for making the text look nice in folders etc and outside the scope for the grammar itself. The tags **XX** and **NAC** are often used since Talbanken makes a difference between subject and object noun phrases. The analysis of elliptical expression in (72)

(72) För stora krav.
"Too high demands."

contains the tags **XX** and **NAC**, since it is not obvious whether the noun phrase is used as subject or an object. The tags shown in figure 5.5 occur quite frequently in the treebank and are always translated to metas, which lowers our result.

NAC	Not a constituent
XP	Other (non-coordinated) phrase
DB	Doubled function
PU	List item
XX	Unclassifiable part-of-speech

Figure 5.5

The main goal has been to be able to translate shorter sentences, with no idioms or conjunction. If we assure that the lexicon contains the correct word class for all lemmas

involved, we can restore more than 85 % of the nodes in the original tree. If we lift all the restrictions excluding the `PU`, we get 65 % coverage. If we test randomly collected sentences that do not contain any of the tags listed in figure 5.5, 72 % can be restored (see figure 5.6)

No list items	65 %
No special punctuation or bad tags	72 %
Short sentences with known words	85 %

Figure 5.6

A mapping between GF and the Wall Street Journal of Penn Treebank has earlier been conducted [Angelov, 2011b]. The percentage of restored nodes from Peen Treebank is higher than our results. The reason for this may be the fact that English is syntactically less complicated than Swedish. Furthermore, the text in Talbanken are from various brochures, newspapers and text books, where idiomatic expressions are more likely the be appear and the language presumably less strict than in Wall Street Journal[1]. Also, the Penn Treebank contains a lower number of tags, 82 compared to more than 300 in Talbanken. Even if the tags describing the same word class as another tag are excluded, Talbanken still leaves us with more than 130 tags. With more tags, we get more information, but as the number increase, so does the amount of work of finding the correct translation for each combination of tags and writing rules that cover all constructions.

We believe that the results could be enhanced by simply adding more rules and in this way get a wider coverage. There are many special cases that require special rules. Since we are not aware of any formal specification of how the tags may be combined in Talbanken, the only way of finding all possibilities are to manually look for different patterns. Another option would be to make the mapping more robust, but the robustness must not interfere with the correctness.

[1] www.wsj.com

Chapter 6

Discussion

6.1 Evaluation

The project has resulted in

- a large-scale GF lexicon and a program to redo the importation when needed
- an extended grammar covering an important part of Swedish
- a comparison between GF and another annotation
- a deeper testing of the Swedish resource grammar and an estimation of how well GF can be used to describe larger parts of a language
- a study of how dependent types can be used in the resource grammars

Besides being capable of reimporting Saldo, the lexicon extraction program could also be modified for importing other lexical resources. The only requirement is that the resource provides inflection tables.

The grammar has been extended and enhanced, and its current status is a specialized extension of the resource grammar. Besides parsing, the grammar may well be used for language generation, which works fast even when using an extensive lexicon. Although it is not been formally verified, we believe that the majority of the sentences generated are grammatically correct in a syntactical point of view. Without any semantic knowledge, nonsense phrases cannot be avoided in random generation. However, given that the abstract tree has a meaningful interpretation, the linearization should be correct. There are some cases when the correctness of the output has been put aside in order to increase the expressivity, such as `ComplBareVV`, which use a verb-complement verb without the infinitive mark, as in sentence (73b).

(73) a. Jag börjar **att** bli hungrig
*I begin **to** become hungry*
"I'm getting hungry"

b. Jag börjar bli hungrig
I begin become hungry
"I'm getting hungry"

Since there is no information in the grammar or lexicon about which verbs that allows the infinitive mark to be left out, it will also allow the more questionable sentence (74b).

(74) a. Jag låter bli **att** titta
*I let be **to** watch*
"I refrain from watching"

b. *Jag låter bli titta
I leave be watch

As these functions serve to provide stylistic flexibility when parsing, they can be left out when generating language, and the grammar then generates text of good grammatical quality.

When it comes to parsing, we do not get far without robustness. The grammar in itself is by no means robust, and just one unexpected punctuation mark, unknown word or ellipsis will cause the parsing of the whole sentence to fail.

Parsing bigger parts of Talbanken would hence give very low results at this stage, and a comparison of the results would not be of much value as there would not be enough material to do be able to do any interesting analysis. An estimation of the improvement can be given by looking at the results from running the test suite used for the grammar development. The sentences given in the test suite are short, four to 10 words, and the words are in most cases included in the lexicon, but there are also constructions that have not been implemented during this project. The first grammar could parse less than half of the sentences, the result for the final grammar was 66 %. It is thus not yet interesting to talk about coverage, but about the quality and the ability to scale up, which has so far proved to be good. I further believe that the presence of a expert in Swedish, professor Elisabet Engdahl[1], has increased the standard substantially.

By the renewed import of Saldo, we have doubled the size of the lexicon and thereby added many of the commonly used words that were missing from the older version. This is of course a big improvement. However, the lexical part still requires much work before it can be made use of. We need valency information to make a good analysis. The lexicon is also too big to use with the current techniques. Its size causes the incremental parsing algorithm to use more heap memory than normal computers allow. To solve this, we need to use the lexicon data more cleverly.

[1] http://svenska.gu.se/om-oss/personal/elisabet-engdahl

6.2 Future Work

At the end of the current project, we are left with many interesting future directions. The future work described in this section is divided into two categories: the ones aiming at making the parser robust and the ones that can be seen as extensions of the work done so far.

6.2.1 Enhancements and improvements

Grammar

The grammar should cover the most prominent Swedish features. Even as some work must be left for the robustness, there are specific constructs that are desirable and interesting to implement. A few examples are listed here:

- *Pronominal object shift* is common in Swedish and obligatory in Danish and Norwegian.

  (75) a. Jag ser honom **inte** b. Jag ser **inte** honom
  I see him not *I see not him*

  Personal pronouns are allowed to precede negation and some other sentence adverbials in main clauses without auxiliary verbs.

  (76) a. *Vi har honom inte sett b. * Jag ser huset inte
  we have him not seen b. *I see the house not*

  Although object shifts are frequently used, they are hardly found in Talbanken's part P, which has been the inspiration for this project. Therefore, this implementation has so far not been prioritized.

- *Bare indefinites* are at this point always parsed as mass nouns.

  (77) Jag dricker vatten
  I drink water

  The parse tree of sentence (77) looks as follows:

  ```
  PredVP (UsePron i_Pron) (ComplSlash (SlashV2a drink_V2)
                                      (MassNP (UseN water_N)))
  ```

  This however is not the correct analysis for 'hund' (*'dog'*) in sentence (78).

  (78) Vi ska köpa hund
  we will buy dog
  "We are going to buy a dog"

- The implementation of reflexive pronouns can be improved. It should for example be possible to differentiate between object and subject control verbs.

(79) a Han ber henne att städa sitt rum
he begs her to clean SELF'S *room*
"He asks her to clean her room"

b Han lovar henne att städa sitt rum
he promises her to clean SELF'S *room*
"He promises her to clean his room"

Lexicon

Our lexicon still lacks some parts that can be imported from Saldo. The multiple-word verbs (`vbm`) and multiple-word nouns (`nm`) should be imported to an idiom lexicon, which can be used as an extension to the main lexicon. For the words that we tried but failed to import, another tool for lexicon acquisition could be used. The tool developed in the previous part of this project[2] would be suitable. All in all, it should be ensured that we use as much information as possible from our resources.

6.2.2 Making the parser robust

To achieve our goal of parsing unrestricted text, we need to use statistical methods. The parser needs to be able to handle ellipses and long and complex sentences, names and numbers, compound words and other unknown constructions.

Parallel parsing/Chunk parsing

Instead of parsing a whole sentence in one go, we intend parse parts of it in parallel and then try to combine the pieces into a parse tree for the whole sentence. This method should be faster then normal parsing and would at the same time give robustness; components that cannot be combined by any grammar rule are combined by a meta function. When an unknown construction is encountered, the parser will not fail but give a partial parse tree with the analysis for the rest of the sentence [Angelov, 2011b]. The output given to the user will be more valuable, since partial parse trees are more interesting the no parse trees.

Lexicon

No matter how much we increase the size of our lexicon, we will never cover all compounds. We therefore need to be able to do compounding analysis. Saldo has tools for this, which might be usable in this project as well [Forsberg, 2007]. As noted, we additionally need to come up with better methods for using the lexicon when parsing, since its time and memory consumption is very high. One possibility is of course to do deeper refinements of the parsing source code, an extensive work which is far outside the scope for this project. Other solutions are to either adapt the lexicon automatically for its domain and hence making it smaller and faster, or to adapt the input sentences to a smaller lexicon by preprocessing.

Further, the lexicon needs valencies since this information lays the foundation for the GF analysis. During the mapping, described in section 5, we analysed how Talbanken annotates

[2] web.student.chalmers.se/~mahlberg/SwedishGrammar.pdf

valency information. It should be possible to extract data from Talbanken, showing how words are normally used. Lexin[3] provides list of valencies information, from which lexical information could be extracted, given that the data is freely available. This is also a source of information helping us separate between verbs that can be used without an infinitive marker from the ones that cannot, cf. (74a-b). However, Lexin is relatively small and does not contain more than 10 000 entries.

Many Swedish verbs can be used with different number of arguments. Having one lexical entry for every possible usage of a verb does not seem to be a good idea considering the ambiguities it will lead to and the already high time usage. The task should instead be left to the robust layer of the parser, possibly implemented by using external resources.

Probabilities

The grammar will always contain ambiguities and before returning the result to the user, some analysis should be done to find the most probable tree. When the size of the lexicon increases, so will the ambiguities as the number of word forms overlapping each other gets higher. Our new grammar also have many more rules which contribute to increasing the number of interpretations of each sentence.

GF already allows probabilities, and by using a large GF treebank, we can get more reliable statistics for our functions. By implementing dependency probabilities, our results would be even better.

Named entity recognition

The parser should be able to identify names of persons, companies, countries etc. This can be done by the Haskell program executing the parsing [España-Bonet et al., 2011].

6.3 Conclusion

We have developed the main components for a deep Swedish parser; an extended grammar and lexicon and material for evaluation and disambiguation. By starting from the GF resource grammar, we got a well-defined system for describing language.

Our final goal is to be able to parse unrestricted text, but considering that no syntactic theory is yet capable of wholly covering a natural language, we are satisfied with the grammar implementation of an important fragment of Swedish and have focused on constructions that are frequent in Talbanken. Being a natural language processing-project, it will probably never be entirely complete, but by combining the rule-based parser with statistical techniques, as described in section 6.2, we still believe that it is possible to achieve our goal.

All parts of the project are open-source and may thus be used in other applications. The grammar and the lexicon may be beneficial also when working with controlled languages, as it increases the coverage of the Swedish resource grammar.

The constructions that we have focused on have all been possible to implement, with varying amounts of work. Many of them could be done by utilizing and extending the resource library but in some cases we needed to part from the multilingual abstract and use other grammatical theories in order to arrive at a good analysis.

[3] http://spraakbanken.gu.se/lexin/

Bibliography

[Angelov, 2011a] Angelov, K. (2011a). A Treebank for Grammatical Framework.

[Angelov, 2011b] Angelov, K. (2011b). *The Mechanics of the Grammatical Framework*. PhD thesis, Chalmers University of Thechnology and University of Gothenburg.

[Angelov et al., 2010] Angelov, K., Bringert, B., and Ranta, A. (2010). PGF: A Portable Run-time Format for Type-theoretical Grammars. *Journal of Logic, Language and Information*, 19:201–228.

[Angelov and Ranta, 2009] Angelov, K. and Ranta, A. (2009). Implementing Controlled Languages in GF. In *CNL*, pages 82–101.

[Bender et al., 2002] Bender, E. M., Flickinger, D., and Oepen, S. (2002). The Grammar Matrix: An Open-Source Starter-Kit for the Rapid Development of Cross-Linguistically Consistent Broad-Coverage Precision Grammars. In Carroll, J., Oostdijk, N., and Sutcliffe, R., editors, *Proceedings of the Workshop on Grammar Engineering and Evaluation at the 19th International Conference on Computational Linguistics*, pages 8–14, Taipei, Taiwan.

[Birn, 1998] Birn, J. (1998). Swedish Constraint Grammar. Available from: http://www2.lingsoft.fi/doc/swecg/intro/.

[Borin et al., 2008] Borin, L., Forsberg, M., and Lönngren, L. (2008). SALDO 1.0 (Svenskt associationslexikon version 2). Available from: http://spraakbanken.gu.se/personal/markus/publications/saldo_1.0.pdf.

[Bresnan, 1982] Bresnan, J. (1982). *The Mental Representation of Grammatical Relations*. MIT Press.

[Burke and Johannisson, 2005] Burke, D. A. and Johannisson, K. (2005). Translating Formal Software Specifications to Natural Language/A Grammar Based Approach. In *In Proceedings of Logical Aspects of Computational Linguistics (LACL'05)*, pages 52–66. Springer-Verlag.

[Butt et al., 2002] Butt, M., Dyvik, H., King, T. H., Masuichi, H., and Rohrer, C. (2002). The Parallel Grammar project. In *COLING-02 on Grammar Engineering and Evaluation*, pages 1–7, Morristown, NJ, USA. Association for Computational Linguistics. Available from: http://dx.doi.org/10.3115/1118783.1118786.

[Caprotti, 2006] Caprotti, O. (2006). WebALT! Deliver Mathematics Everywhere.

[Copestake et al., 1999] Copestake, A., Flickinger, D., Sag, I. A., and Pollard, C. (1999). Minimal Recursion Semantics: An Introduction. Draft. Available from: http://www-csli.stanford.edu/~sag/sag.html.

[Einarsson, 1976] Einarsson, J. (1976). Talbankens skriftspråkskonkordans.

[Enache and Angelov, 2011] Enache, R. and Angelov, K. (2011). Typeful Ontologies with Direct Multilingual Verbalization. *LNCS Post-Proceedings of the Controlled Natural Languages Workshop (CNL 2010) , Marettimo, Italy.* Available from: http://publications.lib.chalmers.se/cpl/record/index.xsql?pubid=150487.

[España-Bonet et al., 2011] España-Bonet, C., Enache, R., Slaski, A., Ranta, A., Marquez, L., and Gonzalez, M. (2011). Patent translation within the MOLTO project. Available from: http://www.molto-project.eu/sites/default/files/patentsMOLTO4.pdf.

[Forsberg, 2007] Forsberg, M. (2007). The Functional Morphology Library. Available from: http://www.cs.chalmers.se/~markus/FM_Tech_Report.pdf.

[Gambäck, 1997] Gambäck, B. (1997). *Processing Swedish Sentences: A Unification-Based Grammar and Some Applications.*

[Gazdar, 1981] Gazdar, G. (1981). Unbounded Dependencies and Coordinate Structure. *Linguistic Inquiry*, 12:155–184.

[Hall, 2007] Hall, J. (2007). A Hybrid Constituency-Dependency Parser for Swedish.

[Holmes and Hinchcliff, 1994] Holmes, P. and Hinchcliff, I. (1994). *Swedish - A Comprehensive Grammar.* Routledge, London, 2nd edition.

[Josefsson, 2001] Josefsson, G. (2001). *Svensk universitetsgrammtik för nybörjare.* Studentlitteratur.

[Joshi, 1975] Joshi, A. K. (1975). Tree adjunct grammars. *Journal of Computer and System Sciences archive.*

[Kokkinakis and Kokkinakis, 1999] Kokkinakis, D. and Kokkinakis, S. J. (1999). A Cascaded Finite-State Parser for Syntactic Analysis of Swedish. In *In Proceedings of the 9th EACL*, pages 245–248.

[Laanemets, 2009] Laanemets, A. (2009). The passive voice in written and spoken Scandinavian. Available from: http://gagl.eldoc.ub.rug.nl/root/2009-49/2009-49-07/?pLanguage=en&pFullItemRecord=ON.

[Ljunglöf, 2004] Ljunglöf, P. (2004). *Expressivity and Complexity of the Grammatical Framework.* PhD thesis. Available from: http://www.ling.gu.se/~peb/pubs/Ljunglof-2004a.pdf.

[Ljunglöf et al., 2005] Ljunglöf, P., Bringert, B., Cooper, R., Forslund, A.-C., Hjelm, D., Jonson, R., and Ranta, A. (2005). The TALK Grammar Library: An Integration of GF with TrindiKit. Available from: http://www.talk-project.org/fileadmin/talk/publications_public/deliverables_public/TK_D1-1.pdf.

[Martin-Löf, 1984] Martin-Löf, P. (1984). Intuitionistic type theory. *Notes by Giovanni Sambin of a series of lectures given in Padua, June 1980.*

[Milner et al., 1997] Milner, R., Tofte, M., and Macqueen, D. (1997). *The Definition of Standard ML.* MIT Press, Cambridge, MA, USA.

[Nivre, 2002] Nivre, J. (2002). What Kinds of Trees Grow in Swedish Soil? A comparison of four annotation schemes for Swedish.

[Nivre et al., 2006] Nivre, J., Nilsson, J., and Hall, J. (2006). Talbanken05: A Swedish treebank with phrase structure and dependency annotation. In *In Proceedings of the fifth international conference on Language Resources and Evaluation (LREC2006)*, pages 24–26.

[Norell, 2008] Norell, U. (2008). Dependently typed programming in Agda. In *In Lecture Notes from the Summer School in Advanced Functional Programming*.

[Pollard, 1984] Pollard, C. (1984). *Generalized phrase structure grammars, head grammars and natural language*. PhD thesis.

[Pollard and Sag, 1994] Pollard, C. and Sag, I. (1994). *Head-Driven Phrase Structure Grammar*. University of Chicago Press.

[Ranta, 2004] Ranta, A. (2004). Computational Semantics in Type Theory. *Mathematics and Social Sciences*, 165:31–57.

[Ranta, 2009] Ranta, A. (2009). The GF Resource Grammar Library. *Linguistic Issues in Language Technology*.

[Ranta, 2011] Ranta, A. (2011). *Grammatical Framework: Programming with Multilingual Grammars*. CSLI Publications, Stanford.

[Rayner et al., 2000] Rayner, M., Carter, D., Bouillon, P., Digalakis, V., and Wirén, M. (2000). *The spoken language translator*. Cambridge Univerisy press.

[Seki et al., 1991] Seki, H., Matsumura, T., Fujii, M., and Kasami, T. (1991). On multiple context-free grammars. *Theor. Comput. Sci.*, 88:191–229. Available from: http://dx.doi.org/10.1016/0304-3975(91)90374-B.

[Simon Thompson, 1999] Simon Thompson (1999). *Haskell: The Craft of Functional Programming*. Addison-Wesley, 2nd edition.

[Stymne, 2006] Stymne, S. (2006). Swedish-English Verb Frame Divergences in a Bilingual Head-driven Phrase Structure Grammar for Machine Translation. Master's thesis, Linköping University.

[Søgaard and Haugereid, 2005] Søgaard, A. and Haugereid, P. (2005). A brief documentation of a computational HPSG grammar specifying (most of) the common subset of linguistic types for Danish, Norwegian and Swedish. *Nordisk Sprogteknologi 2004*, pages 247–56.

[Tapanainen and Järvinen, 1997] Tapanainen, P. and Järvinen, T. (1997). A non-projective dependency parser. In *In Proceedings of the 5th Conference on Applied Natural Language Processing*, pages 64–71.

[Teleman et al., 1999] Teleman, U., Hellberg, S., and Andersson, E. (1999). *Svenska Akademiens grammatik*. Svenska Akademien, Nordstedts.

[Xiaochu Qi, 2009] Xiaochu Qi (2009). An Implementation of the Language Lambda Prolog Organized around Higher-Order Pattern Unification. *CoRR*.