KTH ROYAL INSTITUTE
OF TECHNOLOGY

Degree Project in Computer Science and Engineering

Second cycle, 30 credits

Generating Wikipedia Articles with Grammatical Framework

A Case Study

KEIVAN MATINZADEH

Stockholm, Sweden, 2023

Generating Wikipedia Articles with Grammatical Framework

A Case Study

KEIVAN MATINZADEH

Degree Programme in Computer Science and Engineering

Date: February 23, 2023

Supervisors: Johan Boye, Aarne Ranta

Examiner: Viggo Kann

School of Electrical Engineering and Computer Science

Swedish title: Generering av Wikipedia-artiklar med Grammatical Framework

Swedish subtitle: En fallstudie

Abstract

Natural language generation is a method used to produce understandable texts in human languages from data [1]. Grammatical Framework is a grammar formalism and a functional programming language using a non-statistical approach to build natural language applications. It separates the semantics and the syntax - achieving multilingualism by mapping the same semantic model to several syntaxes [2]. Grammatical Framework also has a large library called the Resource Grammar Library which serves the programmer pre-made functions in over 30 languages ready to be used to build words and sentences [3].

This report investigates if Grammatical Framework can be successfully used to perform natural language generation in order to create Wikipedia articles from data taken from Wikidata.

A grammar and a program has been built to generate articles in Swedish for urban areas in Sweden. The grammar has been built around the structure of the first three sentences in the Swedish article about the urban area Linköping. Furthermore, the grammar and program is extended in order to support generation of the same articles in English and French.

The results show that Grammatical Framework can be somewhat successfully used to generate small Wikipedia articles in different languages using data from Wikidata as input. While all texts were coherent, the Swedish texts were the ones having the least amount of grammatical mistakes. The biggest drawback is the rule of no pattern matching on run-time arguments, which severely limits the programmer since many functions in the resource grammar library use pattern matching internally.

Even though Grammatical Framework does not solve the whole problem, it serves as a powerful enough tool to be suitable for natural language generation, with the main advantage being that it relieves the programmer from needing to pay attention to tasks related to grammar such as inflection and gender agreement.

Keywords

Grammatical Framework, Computational Linguistics, Natural Language Generation, Computer Science

Sammanfattning

Textgeneration är en metod som används för att generera naturlig text från data. Grammatical Framework är en grammatikformalism och ett funktionellt programmeringsspråk som använder ett icke-statistiskt tillvägagångssätt för att skapa språkteknologiska applikationer. Grammatical Framework separerar semantik och syntax, och uppnår flerspråkighet genom att länka samma semantiska model till flera syntaxer. Grammatical Framework har också ett stort bibliotek, en resursgrammatik, kallad Resource Grammar Library, som tillhandahåller applikations-programmeraren färdiga funktioner i över 30 språk redo att användas för att skapa ord och meningar.

Syftet med den här rapporten är att undersöka om Grammatical Framework på ett framgångsrikt sätt kan användas för att generera Wikipedia-artiklar genom att använda data taget från Wikidata.

En grammatik och ett program har skapats för att generera artiklar på svenska för svenska tätorter. Grammatiken använder de tre första meningarna i den svenska artikeln om tätorten Linköping som text-struktur. Vidare utökas grammatiken och programmet till att kunna generera samma artiklar på engelska och franska.

Resultaten visar att Grammatical Framework är någorlunda framgångsrik när det kommer till att generera små Wikipedia-artiklar på olika språk. Fastän alla texter var läsbara, så hade de svenska texterna minst antal grammatiska fel. Den största nackdelen är den regel i Grammatical Framework som inte tillåter mönstermatchning med run-time argument, vilket begränsar programmeraren då många funktioner i resursgrammatiken använder möstermatching internt på sina argument.

Även om Grammatical Framework inte löser hela problemet så är det ett tillräckligt kraftfullt verktyg för att vara lämpat till att användas vid textgenerering, där den största fördelen är att den avlastar programmeraren från att behöva tänka på böjning och andra grammatiska aspekter.

Nyckelord

Grammatical Framework, Beräkningslingvistik, Textgenerering, Dataavetenskap

Acknowledgments

I would like to thank Johan Boye for his help and guidance throughout the work of this thesis, without him this would not have been possible. I also want to thank Aarne Ranta for his supervision and for his insights on all things related to Grammatical Framework.

Stockholm, February 2023
Keivan Matinzadeh

Contents

List of Figures

List of Tables

List of acronyms and abbreviations

AST	Abstract Syntax Tree
GF	Grammatical Framework
NLG	Natural Language Generation
RGL	Resource Grammar Library
UD	Universal Dependencies

Chapter 1

Introduction

As of November 21 2021, Wikipedia has 1.7 billion unique-device visitors monthly [4], making it the world's largest reference website. Of all 58 million articles, over 6 million are written in English. The European languages Swedish, French, German, and Dutch, have more than two but less than three million articles. While this is a substantial amount, there are over 100 languages that have less than 10 000 articles in their respective Wikipedia domains [5].

Abstract Wikipedia is an initiative from the Wikimedia Foundation, where the goal is to let more people share knowledge in more languages. Wikipedia articles can be created and maintained in a language-independent way. A language-independent article can then be translated into a particular language and used by a Wikipedia domain in that particular language [6].

The long-term goal is that this will enable the creation of language-independent articles. This would mean that the number of articles available for small and medium-sized Wikis in their target language would drastically increase.

Natural language generation is a method for generating text from data, and it is itself a sub-domain of Natural Language Processing. In Natural language generation, there are two main approaches to machine translation; the symbolic approach and the statistical approach. While the statistical, data-driven approach is the most popular right now [1], the symbolic approach is still used and holds some advantages over the

statistical approach, such as not needing to be trained on a larger number of data [2] which can be useful when trying to generate text in languages with less available material.

One available symbolic approach is *Grammatical Framework*. Grammatical Framework is a grammar formalism based on type theory [7]. It enables the separation of what is called abstract and concrete syntaxes, where an abstract syntax can be seen as a model of the semantics of a domain, and a concrete syntax is a mapper that maps the semantics to a target language [8]. Multilingualism is achieved through its ability to parse and linearise between abstract and concrete languages. The abstract syntax can also be seen as an interlingua, a semantic expression of a sentence describing how the concepts expressed from each word in the sentence relate to each other.

This report aims to investigate if a Grammatical Framework grammar can be used to successfully generate Wikipedia articles using data from Wikidata.

1.1 Problem

Articles in Wikipedia are mainly written by humans. Since many people speak English, especially in the West, it makes sense that most articles on Wikipedia are in English. This also means that there are fewer articles for smaller languages. Using a purely statistical approach to ”fill the gap” would not necessarily produce well-written articles as statistical approaches need large amounts of data to produce coherent and fluent-sounding text. Using a grammar-based approach where there exists a number of predefined structures for generating texts by using the same data the articles use, could then help solve this problem.

For example, using the structure of the article about the city Linköping to generate a text about the city Örebro: The variability of the data needs to be taken into account; unlike Linköping, Örebro is not an episcopal see of a diocese*, and it is not in the same county nor in the same municipality. Furthermore, generating the article in a language with more complex gender inflection rules, such as a Romance language, will force the grammarian to handle two gender cases for each noun (masculine and feminine).

* An episcopal see is an area over which a bishop rules [9]

This thesis aims to explore these ideas in a case study, and to answer the following questions:

Is it possible to use the structure of a Wikipedia article about a topic, and generate articles of the same structure for different but related topics?

Is it possible to use Grammatical Framework to generate coherent and fluent translations of the generated texts into different languages?

1.2 Delimitations

The data is limited to the data available at Wikidata.com. The study itself is limited to generating texts about Swedish urban areas.

1.3 Structure of the thesis

The relevant background is supplied in chapter 2 and the method is presented in chapter 3. The method is evaluated by generating articles for 100 different urban areas, shown in chapter 4. Chapter 5 contains a discussion of the results as well as general thoughts about using Grammatical Framework for natural language generation. The conclusion and recommendations for future work are presented in chapter 6.

Chapter 2

Background

2.1 Linguistics

2.1.1 Constituency

The idea that grouped up words can behave as units, or constituents, is called syntactic constituency [10]. Two common constituents are briefly presented below:

Noun Phrase (NP) A phrase where the head word is a noun, a pronoun, or a proper name [11]:

```
a boat
the old man's car
me and my friends
Stockholm
```

Verb Phrase (VP) A phrase consisting of a verb followed by other things, such as a noun phrase [10]:

```
discussing with a person
ate food
```

2.1.2 Context-free Grammars

A context-free grammar (CFG) is a formal grammar describing a formal language. It is a language-generating rewriting system made up of

terminal symbols, non-terminal symbols, and grammatical rules in the form of *productions* [12]. Terminal symbols correspond to words in the language, and non-terminal symbols express abstractions over the terminal symbols [10]. A production rule is made out of a non-terminal symbol followed by a right-pointing arrow, and then a set of terminal and/or non-terminal symbols, such as $S \rightarrow A\ B$.

A CFG generates sentences, which can be seen in the example below of a simple grammar:

```
S          →  NP VP
NP         →  Det Noun_subject
VP         →  V Noun_object
Det        →  "the" | "a"
Noun_subject  →  "boy"
Noun_object  →  "Japanese"
V          →  "speaks" | "talks"
```

The set of all sentences generated by a grammar represents the language [12], where all sentences in the set need to be derivable by a start symbol S. A sentence that can be generated by a grammar is seen as grammatical, and a sentence that cannot be generated by said grammar is seen as ungrammatical [10]. In the example above, only four different sentences are seen as grammatical:

1. the boy speaks Japanese
2. the boy talks Japanese
3. a boy speaks Japanese
4. a boy talks Japanese

Changing *e.g.,* Noun_object to include every language in the world (over 7000) would result in the grammar being able to generate over 28 000 different grammatical sentences.

2.1.3 Syntax Trees

Syntax trees, or parse trees as they are also called, are trees representing the generation of a sentence by some grammar.

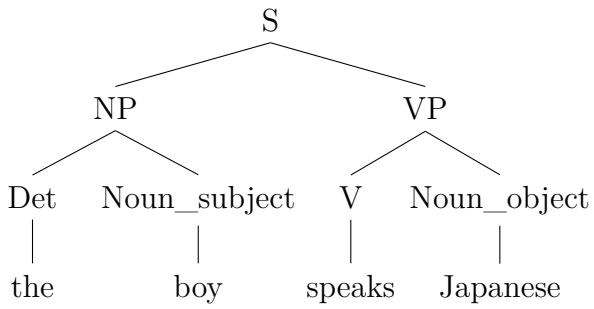


Figure 2.1: Syntax tree of one of the sentences according to the grammar described in 2.1.2

2.1.4 Dependency Grammars

The main idea behind Dependency Grammar (DG) is that in a sentence, all words but one depend on other words. The word that is not dependent on any other word is called the *root* [13]. The words have a binary grammatical relation between each other, where one word is the *head* and the other word, the one that is immediately dependent on the head, becomes the dependent. The root node can then be seen as the head of the whole structure. Dependency grammar also allows for the labelling of the grammatical relationships between head-dependent pairs, such as *subject*, *direct object* and *indirect object* [14].

An advantage that dependency grammars hold over context-free grammars is the ability to abstract away from the word order in a language, meaning that different placement-orders in a sentence of a word in relation to its head, do not affect the way the dependency tree would be parsed [14].

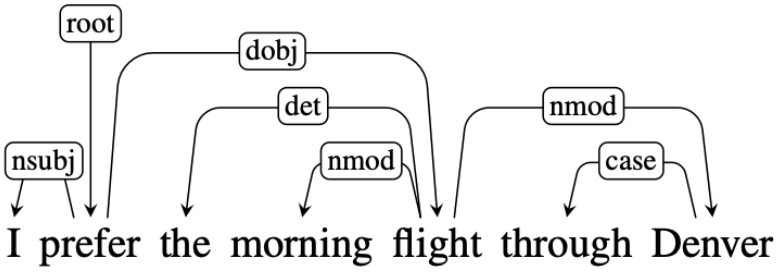


Figure 2.2: Dependency relations between words in a sentence as seen in [14].

2.2 Formal Language Theory

2.2.1 Abstract Syntax Trees

An Abstract Syntax Tree (AST) abstracts away from the semantically insignificant parts of an expression. The parse tree contains the lexical items of an expression in the order they have been parsed, while an AST will look the same even if the expression has *e.g.,* a different word order.

While parse trees are defined by context-free grammars, ASTs are defined by constructor type signatures and it is the trees that are returned by the parser and manipulated by the type checker [15]. The parse trees for the expressions $3 * 4$ and $4 * 3$ can be seen in 2.3.

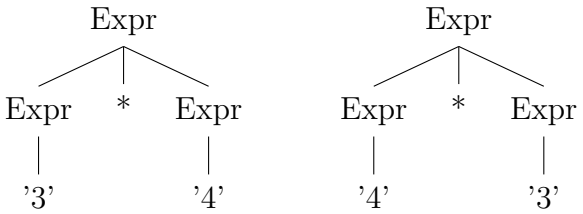


Figure 2.3: Parse trees for the expressions $3 * 4$ and $4 * 3$.

Both parse trees in 2.3 can be represented by the AST shown in 2.4.

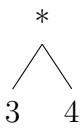


Figure 2.4: The same AST representing both parse trees in 2.3.

While the two expressions are syntactically different, they are semantically identical and so they have the same AST.

2.3 Natural Language Processing

Natural Language Processing is an area of research and application in the intersection of linguistics and computer science that explores how

to manipulate text and speech. Computational techniques are employed in order to learn, understand, and produce natural language content [16].

2.3.1 Natural Language Generation

A subfield of natural language processing, Natural Language Generation (NLG) is the area concerned with producing understandable texts in human languages from underlying non-linguistic representations of information such as numerical data or structured knowledge bases. Though the input can be linguistic as well, the main challenges of NLG center around non-linguistic input [1].

According to [1], in order to address the challenges of converting data into text, the problem is split into different sub-problems with the following sub-problems presented below being some of the most frequently found in NLG systems[1]:

Content Determination

The information that needs to be conveyed in the text is typically less than the information contained in the data, or not needing to be as detailed as the data. Thus, a decision has to be made on which information to include in the generated text.

When choosing the information to include, the data is filtered and abstracted into semantic representations of information such as graph structures or logical or database languages. These formal representations can among other things express relations between domain entities [1].

Text Structuring

When the information to be conveyed has been chosen, the order of presentation of the messages needs to be decided. The result of this stage is a structured and ordered representation of messages: a text or document plan. According to Gatt, *et al.,* the order preference is constrained by the application domain [1].

Lexicalisation

During the lexicalisation, the decision is made regarding which words or phrases to use to express the meaning of the message. This can be a complicated task as there are often several ways to express a single event in natural language.

The system also usually tries to perform lexicalisation in a way that fits the context of the domain. *e.g.,* varying the language in a football report might be more appreciated than varying the language in a weather report [1].

Linguistic Realisation

When the words and phrases to be used have been decided upon, constituents of a sentence are ordered and the right morphological forms are generated in order to create a well-formed sentence. Three approaches are discussed in [1], the two relevant for this thesis being:

Human-crafted templates Small application domains with minimal expected variation can have their outputs specified using templates. A template can for example be a handwritten sentence where some of the words have been exchanged for variables that are then filled in to complete the sentence. Templates allow for full control of the quality of input since most of the sentence is already predetermined, reducing the risk of generating ungrammatical structures. Handwritten templates can be labour-intensive, though templates can also be learned automatically from corpus data[1].

Hand-Coded Grammar-Based Systems These general-purpose domain-independent systems make all or the majority of the decisions based on the grammar of the language under consideration [1]. These systems are mostly hand-crafted. When there are multiple options to choose from, it can be difficult to pick the right one since hand-crafting rules that are sensitive to context and input is hard for these kinds of general-purpose systems.

2.3.2 Universal Dependencies

Universal Dependencies (UD) is a project with the goal to facilitate multilingual NLP systems and parser research (among other things)

by developing cross-linguistically consistent treebank annotations for many languages [17]. Emphasis is put on parallelism between similar constructions across languages, regardless of differences in word order, morphology and function words. It is based on dependency grammar and the ideas are expressed with three fundamental linguistic unit: *nominals*, *clauses*, and *modifiers*. Nominals typically refer to entities while clauses primarily refer to events, while a modifier is described as an attributive modifier of other units [18].

A word in UD has its morphological specification represented in three different levels [19]:

1. The base form of a word represented as a lemma.
2. The grammatical category of the word represented as a part-of-speech (POS) tag.
3. The lexical and grammatical properties associated with the particular word form represented as a set of features.

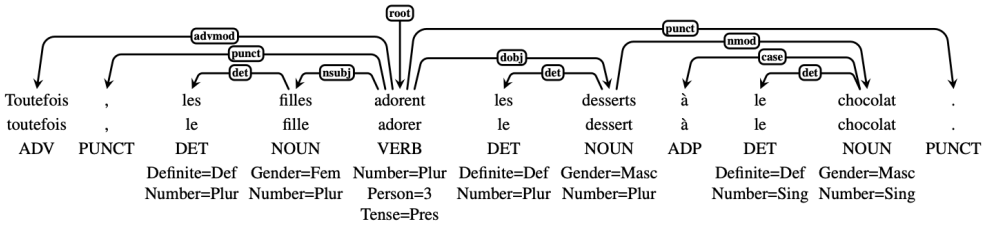


Figure 2.5: UD annotation for a sentence as seen in [20]

.

2.4 Grammatical Framework

Grammatical Framework (GF) is a grammar formalism, and also a functional programming language. The goal of GF is to supply linguists with a high-level grammar formalism, and also supply programmers with an efficient tool for building natural language applications [21].

This section is based on the book *Grammatical Framework: Programming with Multilingual Grammars* [2]. This section aims to provide the reader with a sufficient understanding of GF and its main features.

2.4.1 Structure

A simple GF program can be structured in two ways: either it is defined by a modified BNF grammar, or by an abstract syntax and one or more concrete syntaxes. One of the goals of GF has been multilingualism and which is achieved by using abstract and concrete syntaxes, and so the focus of this section will only be on those methods, and not on BNF grammars.

2.4.1.1 Abstract & Concrete Syntaxes

The abstract syntax describes how the AST should be defined while the concrete syntax describes how the tree should be linearised. This means that all languages in the model will share the same abstract syntax while having their own independent concrete syntaxes. In practice, this means that the abstract syntax defines the type signatures of all the functions in the grammar, while the concrete syntax contains the language-specific implementation of said functions.

A multilingual grammar based on the context-free grammar shown in 2.1.2 can be constructed as:

```
abstract AbstractSyntax = {
 flags
   startcat = S ;
 cat
   S ; NP ; VP ; Det ; Noun_subject ; Noun_object ; V ;
 fun
   Sentence : NP -> VP -> S ;
   Det_Noun_subject_to_NP : Det -> Noun_subject -> NP ;
   V_Noun_object_to_VP : V -> Noun_object -> VP ;
   the_Det, a_Det : Det ;
   boy_Noun : Noun_subject ;
   japanese_Noun : Noun_object ;
   speaks_V, talks_V : V ;
}
```

In the syntaxes, different kinds of rules are known as *judgements*. The abstract syntax above has three kinds of judgements [2]:

- flags - What type of flags there are.
- cat - What type of categories there are, meaning what type of trees can be built.
- fun - the function declarations.

GF denotes a functions type signature in the same style seen in typed functional programming languages such as Haskell, where the argument types and value type are separated by an arrow. The type signatures in the abstract syntax presented above can be compared to the CF grammar in 2.1.2 where the non-terminal symbols on the left side correspond to the return values described by the type signatures in the abstract syntax. It can also be noted that there are no terminal symbols present; GF separates the non-terminal symbols and terminal symbols into an abstract syntax and one or more concrete syntaxes, respectively.

The implementations of the functions are defined in a concrete syntax. Each language has its own concrete syntax. For this grammar, an English concrete syntax is added along with a French concrete syntax:

```
concrete EnglishSyntax of AbstractSyntax = {
 lincat
   S, NP, VP, Det, Noun_subject, Noun_object, V = Str ;
 lin
   Sentence NP VP = NP ++ VP ;
   Det_Noun_subject_to_NP det Noun_subject = det ++ Noun_subject ;
   V_Noun_object_to_VP v Noun_object = v ++ Noun_object ;
   the_Det = "the" ;
   a_Det = "a" ;
   boy_Noun = "boy" ;
   japanese_Noun = "Japanese" ;
   speaks_V = "speaks" ;
   talks_V = "talks" ;
}
```

```
concrete FrenchSyntax of AbstractSyntax = {
 lincat
   S, NP, VP, Det, Noun_subject, Noun_object, V = Str ;
 lin
   Sentence NP VP = NP ++ VP ;
   Det_Noun_subject_to_NP det Noun_subject = det ++ Noun_subject ;
   V_Noun_object_to_VP v Noun_object = v ++ Noun_object ;
   the_Det = "le" ;
   a_Det = "un" ;
   boy_Noun = "garçon" ;
   japanese_Noun = "japonais" ;
   speaks_V = "parle" ;
   talks_V = "parle" ;
}
```

A concrete syntax has two kinds of judgements:

- lincat - What type of linearisation object each respective category tree produces.
- lin - the rules describing how each tree is linearised

If there exists a lincat for each cat, and a lin for each fun, the concrete syntax is complete with respect to its abstract syntax [2].

Though all types are linearised into strings, the linearisation functions respect the type constraints put on them in the abstract syntax. For example, `boy_Noun` and `japanese_Noun` are both of types that linearise into strings, but since the types are different they can not be used as arguments in the same functions and so the original constraint set in the context-free grammar is also present in the GF grammar.

Except for parsing and generation of sentences, the grammar allows for translation from a sentence in English to a sentence in French and vice versa. Since the languages share the same abstract syntax they use the same abstract syntax trees to build words or sentences. When translating the English sentence ”the boy speaks Japanese” into French, the sentence is first parsed with the English concrete syntax into an abstract syntax tree:

```
Sentence (Det_Noun_subject_to_NP the_Det boy_Noun)
```

```
(V_Noun_object_to_VP speaks_V japanese_Noun)
```

The AST is then linearised into French using the French concrete syntax, resulting in the sentence

```
le garçon parle japonais
```

2.4.2 Features

GF offers several data structures to facilitate and make possible the creation of more complex grammars.

Parameters

A parameter is an object of some type. One can define the type `Number` to contain the values `Sg` and `Pl` to signify whether a word is in its singular or plural form [2]:

```
param Number = Sg | Pl
```

Tables

A table is a structure used to formalise inflection tables. A table is defined with the table keyword, and inside its body each element is a key-value pair in the format `key => value`.

An inflection table for the French word for meat, *viande*:

number	**form**
singular	*viande*
plural	*viandes*

This can be expressed in GF with a table:

```
table {Sg => "viande" ; Pl => "viandes"}
```

More complex inflections such as for words inflecting on both number and gender can also be handled by tables:

gender	**singular**	**plural**
masculine	*grand*	*grands*
feminine	*grande*	*grandes*

And as a table in GF:

```
table {
    Masc => table {Sg => "beau" ; Pl => "beaux"}
    Fem => table {Sg => "belle" ; Pl => "belles"}
}
```

Records

A record is a structure that can group together objects of possibly different types. All information belonging to the concrete syntax of an expression is stored in records.

```
{s = table {Sg => "viande" ; Pl => "viandes"} ;
 g = Fem}
```

The record above contains the inflection table for *viande*, but also has another variable `g` which denotes the inherent gender of the word.

A record value is accessed with the dot operator:

```
{s = table {Sg => "viande" ; Pl => "viandes"} ;
 g = Fem}.s
```

which evaluates to

```
table {Sg => "viande" ; Pl => "viandes"}
```

Operations

Functions in the concrete syntax are called operations. These can be used to perform various tasks that would be too tedious to do by hand. An example given in [2] is the `regNoun` operator, which produces a record containing an inflection table for a regular noun in English:

```
oper regNoun : Str -> {s : number => Str} =
\word -> {s = table {Sg => word ; Pl => word + "s"}} ;
```

The expression `regNoun "lion"` will then return the record

```
{s = table {Sg => "lion" ; Pl => "lions"}}
```

2.4.3 Resource Grammar Library

The Resource Grammar Library (RGL) provide the main grammatical rules of a language, such as low-level details of syntax and morphology [2]. This alleviates the programmer of the task of modelling the features

of the language and instead focus can be put on the application grammar, which handles the high-level details. The resource grammar can then be used by the application programmer when needed.

Common Abstract Syntax

The GF Resource Grammar Library's common abstract syntax consists of a set of categories, functions, function words such as pronouns and determiners, and a number of content words in a test lexicon.

Below is a sample of some of the types and functions available in the RGL, along with examples.

Type	**Description**	**Example**
Det	determiner	*a*
N	noun	*airplane*
NP	noun phrase	*an airplane*
PN	proper name	*Stockholm*
VP	verb phrase	*to eat*
Conj	conjunction	*and*
Cl	clause	*he is old and tired*
Utt	utterance	*he took the bus*
PConj	phrase-beginning conjunction	*but*
Phr	phrase	*but he took the bus*
Adv	verb-phrase-modifying adverb	*in the building*

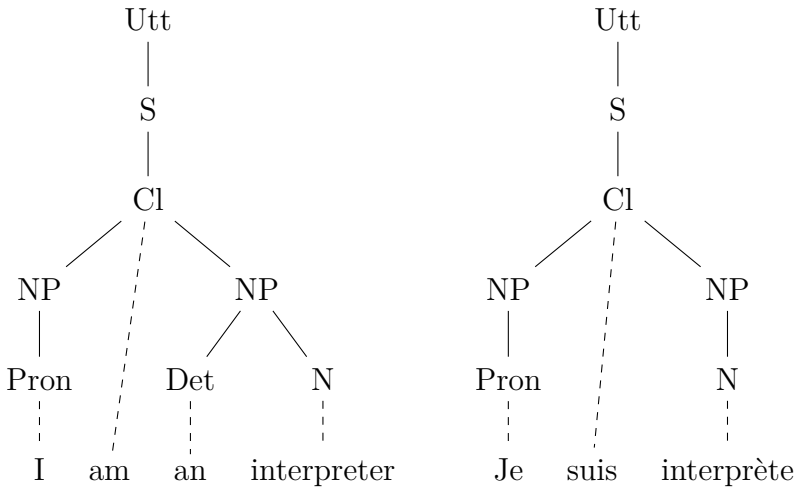
Table 2.1: A sample of the types available in the RGL

Function	**Type**	**Example**
mkNP	Det → N → NP	*the seven rabbits*
mkAdv	Prep → NP → Adv	*with a ball*
mkCl	NP → V2 → NP → Cl	*he likes food*
mkCl	NP → NP → Cl	*Stockholm is a city in Sweden*
mkCN	AP → N → CN	*very big boat*

Table 2.2: A sample of the functions available in the RGL

The abstract syntax is technically an interlingua which implements a level of linguistic theory of cross-lingual syntax, though it is not seen as a semantic interlingua since it cannot guarantee meaning-preserving

translation [22]. This can be demonstrated in the sentence *I am an interpreter*, which translates to *je suis interprète* in French. There is no indefinite article in the French sentence, and so the phrase structure trees will look slightly different:



The two trees show that in order to keep the same semantic meaning between the two languages, the syntax needs to be altered.

The English sentence can be created with the line

```
mkUtt (mkS (mkCl
                (mkNP i_Pron)
                (mkNP a_Det (mkN "interpreter"))))
```

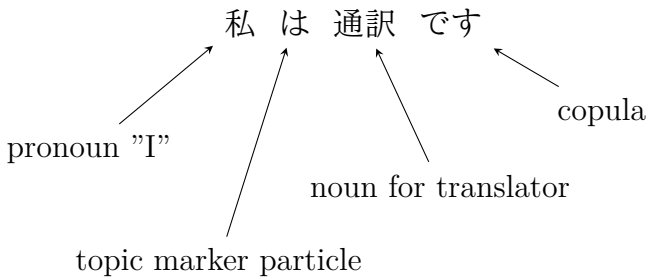
and the French with

```
mkUtt (mkS (mkCl
                (mkNP i_Pron)
                (mkNP (mkN "interpreter"))))
```

The `mkCl` function can take a lot of different arguments, and in the example above it is used with the `NP` → `NP` → `Cl` type signature. In both the French and English implementation of `mkCl : NP → NP → Cl`, the copula is inserted between the two noun phrases in order to connect them. In both cases, the copula is the ”to be” verb.

Looking at how `mkCl : NP → NP → Cl` is implemented in Japanese further demonstrates some of GF's goals, such as preserving meaning between languages and coverage of grammatical structures within individual languages.

The same sentence in Japanese 私は通訳です (Watashi ha tsūyaku desu)



will then produce not only one sentence but a table with the different forms taking into account the topic marker particle は and the subject marker particle が, as well as the plain form and polite form of the copula which in Japanese are the sentence-ending suffixes だ and です, respectively [23]:

```
table
   ResJpn.Particle
   [table
      ResJpn.Style
      ["私" ++ "は" ++ "通訳" ++ "だ";
       "私" ++ "は" ++ "通訳" ++ "です"];
    table
      ResJpn.Style
      ["私" ++ "が" ++ "通訳" ++ "だ";
       "私" ++ "が" ++ "通訳" ++ "です"]];
```

The sentence is generated with the same line of code as the sentence in French, except that the supplied noun is different:

```
mkUtt (mkS (mkCl
                (mkNP i_Pron)
                (mkNP (mkN "通訳"))))
```

In order to preserve meaning, the Japanese implementation of the `mkCl` function adds the necessary particles as well as the copula.

Lexical Paradigms

Each language in the RGL has its own uniquely implemented set of paradigms - functions needed for handling language-specific morphology

and building a lexicon. In addition to inflectional paradigms, a paradigm library also includes names of inherent features and functions for words that need arguments [24].

Function	Type	Explanation
Gender	Type	-
utrum	Gender	*the "en" gender*
neutrum	Gender	*the "ett" gender*
mkN	(apa : Str) → N	*predictable nouns: apa-apor, rike-riken, or bil-bilar*
mkN	(nyckel,nycklar : Str) → N	*singular and plural suffice for most nouns*
mkN2	N → Prep → N2	*e.g., syster - till*
mkN3	N → Prep → Prep → N3	*e.g., flyg - från - till*
mkPN	Str → PN	*default gender utrum*
mkV2	V → Prep → V2	*preposition for complement*
mkV3	V → Prep → V3	*preposition for last argument*
mkV3	V → Prep → Prep → V3	*prepositions for both complements*

Table 2.3: A sample of the functions available in the Swedish paradigm library as seen in [25].

For example, the Swedish paradigm library supplies the user with, among other functions, the *utrum* and *neutrum* functions. These functions can be used as arguments to the noun-creating functions to manually set the gender of the noun. These are used when the paradigm cannot figure out the proper gender just by supplying the different noun forms as strings.

Adding arguments to words is also possible with a variety of functions. One of them is `mkN2 : N -> Prep -> N2` which binds a preposition to the noun and returns a new noun of the type `N2`. The sentence *bror till min mamma* (in English *brother of my mother*) can then be constructed by using the `mkN2` function:

```
mkUtt (mkCN (mkN2 (mkN "bror" "bröder")
                  to_Prep)
            (mkNP (mkQuant i_Pron)
                  (mkN "mamma" "mammor")))
```

While many paradigm libraries have similar function names and type signatures, the internal workings do not necessarily need to be the same: while both the Swedish and the French paradigm libraries have the function `mkPN : Str -> PN` for creating a proper name out of a string, the French paradigm also pattern matches the supplied string argument to determine whether the gender of the proper name should be set to feminine or masculine. In contrast, the Swedish version of the paradigm sets the gender of the proper name to utrum by default.

Revisiting the grammar created at the beginning of the section, it can now be modified to include RGL functions and types, making it easier to extend the grammar itself but also to add any new grammars:

```
concrete SwedishSyntax of AbstractSyntax =

open
    SyntaxSwe,
    (S=SyntaxSwe),
    ParadigmsSwe
in

{
lincat
    S = S.Utt ;
    NP = S.NP ;
    VP = S.VP ;
    Det = S.Det ;
    Noun_subject = S.N ;
    Noun_object = S.N ;
    V = S.V ;
lin
    Sentence np vp = mkUtt (mkCl np vp) ;
    Det_Noun_subject_to_NP det Noun_subject =
        mkNP det Noun_subject ;
    V_Noun_object_to_VP v Noun_object =
        mkVP (mkV2 v) (mkNP Noun_object) ;
    the_Det = S.the_Det;
    a_Det = S.a_Det ;
    boy_Noun = mkN "pojke" "pojkar" ;
    japanese_Noun = mkN "japanska" ;
```

```
    speaks_V = mkV "prata" ;
    talks_V = mkV "tala" ;
}
```

Swedish is added as a language. Not using the RGL for Swedish would have resulted in more work. The biggest issue is that definite articles are not used in Swedish, instead, the noun itself is inflicted to reflect definiteness. This would not have been doable in the previous non-RGL version without adding a new function that only takes a noun as an argument and returns a definite noun phrase, effectively needing to change the abstract syntax, meaning that the other concrete syntaxes would need to change as well. The English and French concrete syntaxes can be seen in the appendix.

2.4.4 PGF

The package PGF, which is the runtime format of GF, makes it possible to use GF within a Python program[26, 27].

A PGF file is created by compiling the GF source code into PGF, resulting in the file `Grammar.pgf`:

```
gf -make GrammarSwe.gf GrammarEng.gf
```

A grammar can be loaded with the method, which returns an object:

```
gr = pgf.readPGF(Grammar.pgf)
```

The languages (concrete implementations) can be queried with the `languages` property, which maps a language name to an object:

```
swe = gr.languages["GrammarSwe"]
```

A new tree can be read from a string with the method `pgf.readExpr`:

```
tree = pgf.readExpr("StringObject 'Stockholm'")
```

In order for this to work, a function `StringObject : String -> Str` is defined in the abstract syntax, and an implementation of that function is defined in the concrete syntax: `StringbObject str = str.s`.

A new tree can be created from a GF function by supplying the function name and a list of arguments to the method `pgf.Expr`:

```
tree = pgf.Expr('Func', [Arg1, Arg2])
```

where `Func` can be defined as a function with the type signature

```
Func : TypeA -> TypeB -> TypeC
```

2.5 Related work area

2.5.1 LsjBot

A similar and successful project has been done in the past; a bot called *Lsjbot* created by the Swedish physicist Sverker Johansson which creates Wikipedia articles automatically [28]. The generated articles are stubs - pages that contain only basic bits of information.

In 2014, 2 700 000 articles written in Swedish and Filipino had been created by Lsjbot, and in 2013 it created nearly half of the one million articles on the Swedish Wikipedia.

According to [29], it seems that Sverker Johansson himself chooses a semantic domain, finds a machine-readable database covering said domain, then writes templates suitable to describing the chosen domain and its sub-domains. The bot is then programmed to read from the database, and to insert the data into the templates and generate text that is then uploaded to Wikipedia.

Since 2017, due to a lack of adequate documentation or because of other reasons, at least 900 000 articles written by Lsjbot have been deleted [28].

2.5.2 Abstract Syntax as Interlingua

Ranta *et al.,* explore the concept of using an abstract syntax as an interlingua in natural language generation [22]. It is concluded that even though the abstract syntax of the resource grammar technically is an interlingua, it cannot guarantee meaning-preserving translation as a semantic interlingua.

Furthermore, the use of layered interlinguas is discussed. The idea is to back up interlinguas with other interlinguas, *i.e.,* backing up what is called the controlled natural language abstract syntax with the RGL abstract syntax.

Chapter 3

Methods

This chapter presents the construction of the GF grammar, the methods for collecting and pre-processing the data, as well as a section investigating the use of UD trees as formal representations for performing natural language generation with GF.

3.1 Goal

In order to answer the first research question stated in 1, Swedish urban areas are chosen as the topic and the article of whose structure is to be used for the natural language generation is chosen to be the article about the Swedish urban area Linköping.

> Linköping är en tätort i Östergötland samt centralort i Linköpings kommun, residensstad i Östergötlands län och stiftsstad för Linköpings stift. Linköping är också en universitetsstad med Linköpings universitet grundat 1975. Linköping är Sveriges åttonde största tätort med över 115 000 invånare.

Which in English is:

> Linköping is an urban area in Östergötland as well as the seat of Linköping Municipality, the capital of Östergötland County and the episcopal see of the Diocese of Linköping. Linköping is also a university city with Linköping university founded in 1975. Linköping is Sweden's eighth largest urban area with more than 115,000 inhabitants.

The second sentence also contained the sub-sentence *belägen på Östgötaslätten, strax söder om Stångåns utlopp i Roxen mitt i landskapet och länet* which was edited out as the only locational data relevant would be the coordinates, and producing such as sentence only with the coordinate location would not be crucial to the results. Therefore, it was decided to leave that part out.

What makes the structure of these sentences interesting is that the information they contain is available on Wikidata (except the part edited out). And keeping Wikidata in mind, another change is done to the sentences: The first part of the first sentence *Linköping är en tätort i Östergötland*, describes the location of Linköping in relation to the province it is located in. The information of which province an urban area belongs to is not available for most urban areas in Wikidata and so in order to use Wikidata to its fullest, the structure of the first sentence is slightly changed so that it denotes the county instead of the province, thus becoming *Linköping är en tätort i Östergötlands län* or *Linköping is an urban area in Östergötland county.*

Using Linköping's structure to generate an article about Örebro should then result in

> *Örebro är en tätort i Östergötlands län samt centralort i Örebro kommun och residensstad i Örebro län. Örebro är också en universitetsstad med Örebro universitet grundat 1977. Linköping är Sveriges sjunde största tätort med över 125 000 invånare.*

The difference in structure from the original article about Linköping is that there is no sentence like *stiftsstad för Linköpings stift.* This is because Örebro is not an episcopal see for a diocese, unlike Linköping.

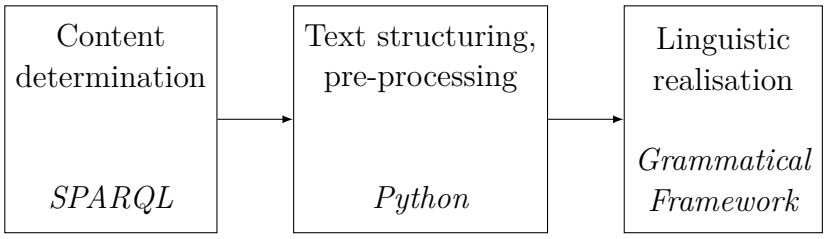
Generating a text for the small urban area Båtskärsnäs should yield a result with even fewer components since it is neither a seat of a municipality, nor a capital of a county, nor an episcopal see of a diocese:

> *Båtskärsnäs är en tätort i Norrbottens län. Det finns över 205 invånare i Båtskärsnäs.*

Thus, the goal is to be able to generate articles about other urban areas using the structure of Linköping's Wikipedia article.

3.2 Construction of the Program

This section describes the steps outlined in the pipeline presented below:



3.2.1 Content Determination - Data gathering

The data was gathered from Wikidata with the database-language *SPARQL*[30]. This was done directly inside the Python script.

The process of gathering the data is straightforward:

1. Determine what content is needed. *e.g.,* municipalities, counties, or universities in Sweden.
2. Go to the site of what would be one of the results and look at its properties listed.
3. Determine what property or properties would be present among all results returned by the query.
4. Write a query based on those properties.

For example, querying for all municipalities in Sweden only requires one property. All entries of municipalities in Sweden are instances of `municipality of Sweden`. The query can then be written as:

```
SELECT ?municipalityLabel WHERE {
    SERVICE wikibase:label {
    bd:serviceParam wikibase:language
        "[AUTO_LANGUAGE],sv". }
    ?municipality wdt:P31 wd:Q127448. }
```

A different example is querying for all universities of Sweden. There is no property defined as ”university of Sweden”. Instead, what can be seen on the Wikidata page for Uppsala University is that it is an instance of `university`. This is not enough to only narrow it down to only Swedish

universities. The entry does have another property; `Country`. The value of the Country property in this case is Sweden. So when writing the query the goal is to ask the database for all objects that are universities and belong to the country Sweden.

The return value from a SPARQL query called from Python is a nested dictionary structure. As the result contains other information as well, the relevant information is extracted and put into a new dictionary for easier access.

In the Python script, each query gets a dedicated method which returns a dictionary or array containing the results.

3.2.2 Text Structuring and Pre-processing

3.2.2.1 Pre-Processing Proper Names

When getting information such as the municipality or county name, the name had to be trimmed down to only the proper name form in order to be able to be usable when linearising into other languages. This is because the same AST is used for all languages and so the tree is created only once, which means that the input cannot be varied to suit a specific language.

For municipalities, this was done by fetching the `shortname` property from a municipality entry in Wikidata. 54 municipalities did not have this property but their shortnames were added by hand directly in the script since they follow a commonly observed pattern: if a proper name ended in a consonant, an 's' was appended to the end of the proper name before adding the noun for county or municipality. The only observable exceptions were if the proper name ended with an 's' or if the name was the urban area Kalmar: Kalmar Municipality (Kalmar kommun) or Kalmar County (Kalmar län).

For counties, the process was simpler since no Swedish county ends in a vowel or the letter s. Therefore the names only needed to be split on the last white space and then if the last word ended in an 's', have it removed:

```
Västra Götalands län    =>
Västra Götalands        =>
Västra Götaland
```

The process for shortening the name of a diocese was identical to that for counties but there needed to be an additional check in case the name to be shortened was either the diocese of Strängnäs or the diocese of Västerås, since both Strängnäs and Västerås are names of urban areas ending with an 's'. Hence

```
Stockholms stift     =>
Stockholms           =>
Stockholm
```

and

```
Strängnäs stift     =>
Strängnäs
```

3.2.2.2 Text Structuring

Text structuring, in this case, determining which *varying property* is present and ordering those properties, is done in Python.

A varying property can be seen as a property not always guaranteed to be present. For example, while all urban areas belong to some county, not all urban areas are the seat of some municipality. Therefore, being the seat of a municipality can be seen as a varying property. In the sentence below all of the varying properties are underlined:

Linköping är en tätort i Östergötlands län samt centralort i Linköpings kommun, residensstad i Östergötlands län och stiftsstad för Linköpings stift.

Each varying property can then be seen as a noun phrase, and utilizing the `mkListNP` function, the noun phrases can be collected into a list of noun phrases, a `ListNP`. The list of noun phrases can then be turned into one noun phrase with the `mkNP : Conj -> ListNP -> NP` function. The first argument is a function of type `Conj`, a conjunction, and the second argument is a `ListNP`. The function creates a noun phrase out of all the noun phrases, separating them by commas except for the last two noun phrases which are separated by the chosen conjunction:

```
mkNP and_Conj [centralort i Linköpings kommun,
               residensstad i Östergötlands län,
               stiftsstad för Linköpings stift]
```

```
=> "centralort i Linköpings kommun, residensstad i
    Östergötlands län och stiftsstad för Linköpings
    stift"
```

Each varying property is looked for by querying Wikidata. If the varying property is present, the query result is used as input for a GF function defined in the grammar. These GF functions can be seen as functions detached from the grammar; not involved in generating the final text - instead, their job is to generate noun phrase trees in Python to be used as input for the GF functions that generate the final text.

The sentence above has three varying properties, so what needs to be defined in the grammar is three functions each returning a noun phrase. Furthermore, there need to be three main sentence functions: one for the case when there are no varying properties, one for the case when there is one varying property (input is a noun phrase) and one function for the case when there are two or more varying properties (input is a list of noun phrases). The process of structuring the first sentence is explained below:

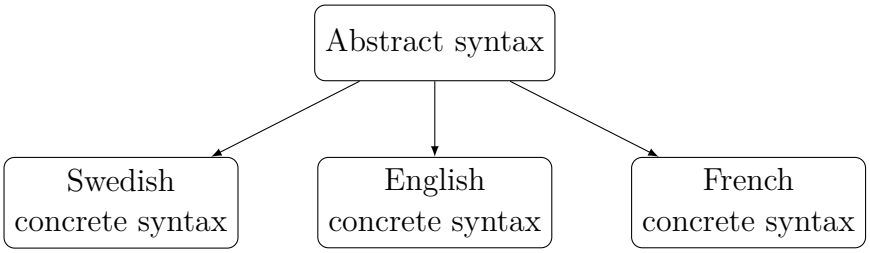
1. Check each varying property. If a property exists use the data from the property to create a noun phrase tree and put the object in an array.
2. - If the array is empty, create a tree with the function used for the case when there are no varying properties.
   - If the array only has one element, create the tree with the function used for the case when there is only one varying property, using the sole noun phrase in the array as input.
   - If the array has two or more elements, create a noun phrase list tree out of the noun phrases in the list and then create the sentence with the function that takes a noun phrase list as one of its input arguments.

Structuring the second sentence, *Linköping är också en universitetsstad med Linköpings universitet grundat 1975*, follows a similar pattern, but since zero or more universities can exist in an urban area, there is no fixed amount of available varying properties (universities). If there exist one or more universities in the urban area, a collection containing each university is created. This can be seen as a varying property which has

varying properties itself, where the varying property is whether or not a university exists or not, and its varying properties are all the universities that exist. If there are no universities then the sentence is makes no sense and is skipped entirely. This implies that there must also be a rule in the grammar which creates a text with two sentences and not three, skipping the second sentence.

3.2.3 Linguistic Realisation - Grammar Construction

This section outlines the construction of the abstract syntax and concrete syntaxes of the grammar. Figure 3.1 presents an overview of the relationship between the abstract syntax and the concrete syntaxes.



3.2.3.1 Design

The abstract syntax has been structured in such a way that each sentence is represented by a `Phr` type. This means that the function producing a sentence has a type signature of the form

```
func : arg_1_type -> arg_2_type ... -> arg_n_type -> Phr
```

Furthermore, the sentence is divided into smaller parts generated by functions based on its varying properties.

Types

The types need to be defined in the abstract syntax and then the implementation of said types is defined in the concrete syntax. This means that just like for functions, a type can be different depending on the concrete syntax. This allows for additional processing of inputs for specific concrete syntaxes. It can also simply be used to change the name of a type to make it more specific to the project. In this project, the type `Sentence` is used as a return value for each sentence line, but it is just

defined to be of the pre-defined `Phr` (defined as ”phrase in text” in GF [3])) type.

Abstract Syntax:

```
Sentence ;
```

Concrete Syntax:

```
Sentence = Phr ;
```

Words
Each word used in the text needs to be defined not as a string but as a GF type. While the proper names are supplied as run-time variables from Python, nouns, verbs and other words need to be defined in the grammar as GF types. For example, nouns such as *tätort* (urban area) and *stiftsstad* (episcopal see) are defined with the `mkN` function:

```
tatort_N = mkN "tätort" "tärtorter"

stiftsstad_N = mkN "stiftsstad" "stiftsstäder" ;
```

Most nouns and verbs need to be defined in the grammar, while the most common prepositions and articles already exist in the language-specific syntax library and can be used directly in the functions without needing to be defined in the local grammar, *e.g.,*

```
mkAdv SyntaxEng.in_Prep (mkNP SyntaxEng.a_Det
                                (mkN "house"))

=> "in a house"
```

Functions
Each sentence was assigned its own function in GF, where the return value was the category *Sentence* of the type `Phr`. In turn, the body of each sentence function consisted of either a set of RGL functions, a set of manually defined functions and/or operations or a set of both.

3.2.3.2 Abstract Syntax and Swedish Concrete Syntax

The abstract syntax and the Swedish concrete syntax were developed concurrently, and the English and French concrete syntax were developed

after, and so the abstract syntax (and subsequently the grammar) will be explained using the examples in Swedish.

First Sentence
The first sentence, *Linköping är en tätort i Östergötlands län samt centralort i Linköpings kommun, residensstad i Östergötlands län och stiftsstad för Linköpings stift*, (Linköping is an urban area in Östergötland County as well as the seat of Linköping Municipality, the capital of Östergötland County and the episcopal see of the Diocese of Linköping in English) is constructed with the function `Line1`, which in turn calls on four different functions in its body. Each of those four functions generates a different part of the sentence:

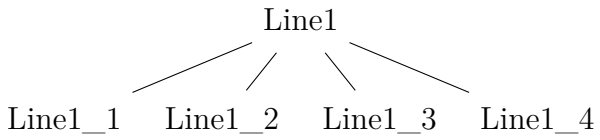


Figure 3.1: Syntax tree of the sentence generated by the function Line_1

Looking at `Line1_1` specifically, its type signature (abstract syntax representation) is defined as

```
Line1_1 : Str -> Str -> Phr ;
```

and the function is implemented in the Swedish concrete syntax as

```
Line1_1 city city_county =
    let city_NP : NP = str_PN_to_NP city ;
        county_NP : NP = create_county_NP city_county ;
        a_urbanarea : NP = mkNP a_Det urbanarea_N ;
        in_county : NP = mkAdv in_Prep county_NP
    in mkPhr (mkCl city_NP
                   (mkNP a_urbanarea in_county)) ;
```

where the operations `str_PN_to_NP` and `create_county_NP` are defined as

```
oper str_PN_to_NP : Str -> NP =
    \city -> mkNP (mkPN city)
```

and

```
oper create_county_NP : Str -> NP =
    \name -> mkNP (mkPN (name ++ "län"))
```

For this example, most expressions in the function are defined in a let statement in order to be more readable. Furthermore, the use of the SyntaxSwe library (in the style of `SyntaxSwe.`*`FunctionName`*) is omitted as to not draw away from the code itself.

The syntax tree for `Line1_1` can be seen in 3.3 and it further illustrates how the functions are used to build sentences:

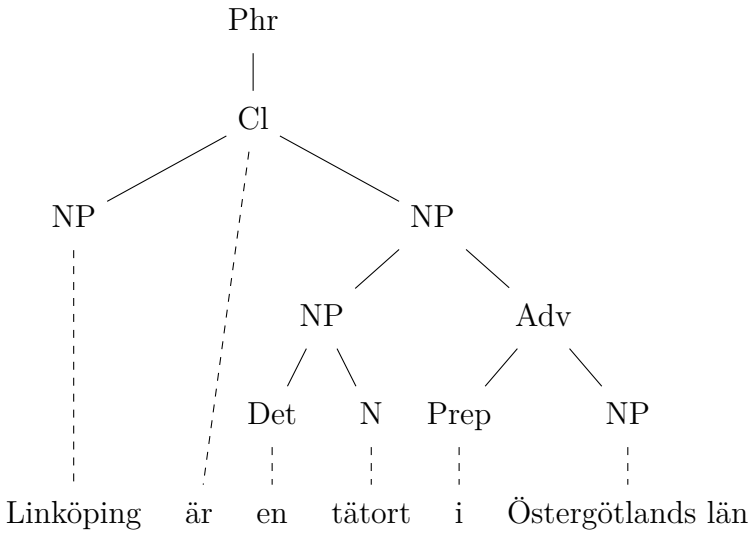


Figure 3.2: Syntax tree of the sentence generated by the function Line1_1

The operation `str_PN_to_NP` is used to do what its name implies, create a noun phrase out of a string. It first creates a proper name out of a string, and then that proper name is made into a noun phrase. Thus this operation should preferably only be used to create noun phrases out of the proper names that are supplied through Python (as strings). This is a necessary operation to have since the only RGL functions taking strings as input are the functions that create numerals and digits. Furthermore, there is only one function in the RGL common abstract syntax that takes a proper name as an argument, the `mkNP` function, and that is why the conversion to a noun phrase has to happen as well. Most, if not all paradigms have a function which creates a proper name out of string, making this conversion rather simple in theory. Though in some cases, as will be seen in the implementation of the French concrete

syntax, applying functions on run-time strings can be more difficult than it seems.

The function `mkAdv : Prep -> NP -> Adv` is used to create an adverbial phrase out of a preposition and a noun phrase: *i Östergötlands län*. Then the function `mkNP : NP -> Adv -> NP` is used to create a noun phrase out of a noun phrase (*en tätort*) and an adverbial phrase, in this case generating *en tätort i Östergötlands län*.

Since the rest of the sentence contains varying properties; whether an urban area is a seat of a municipality, the capital of a county, or an episcopal see of a diocese, each of the existing varying properties is created as noun phrases with their own dedicated functions and collected into a `ListNP`. The `ListNP` is then made into a noun phrase together with a chosen conjunction using the `mkNP: Conj -> ListNP -> NP` function. The noun phrase is in turn made into an utterance type using `mkUtt : NP -> Utt` in order to be used in the function `mkPhr : PConj -> Utt -> Phr` where `PConj` is a phrase beginning conjunction. The ”gluing” of the two phrases can be visualised in the syntax tree below:

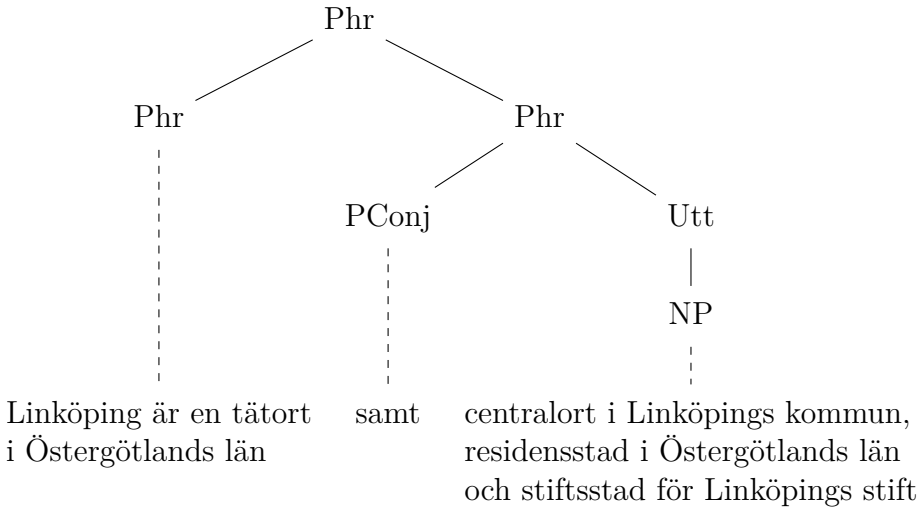


Figure 3.3: Syntax tree of the sentence generated by the function `Line_1`

Second Sentence

The second sentence, *Linköping är också en universitetsstad med Linköpings universitet grundat 1975*, (Linköping is also a university city with Linköpings universitet founded in 1975 in English), describes the

universities of the urban area. While there is only one university in Linköping, there are three universities in Stockholm. Generating an article about Stockholm should then result in a second sentence of the form

> *Stockholm är också en universitetsstad med Kungliga Tekniska högskolan grundad 1827, Karolinska Institutet grundat 1810 och Stockholms universitet grundat 1878.*

The sentence is generated by the function `Line2` and its syntax tree is presented in 3.4.

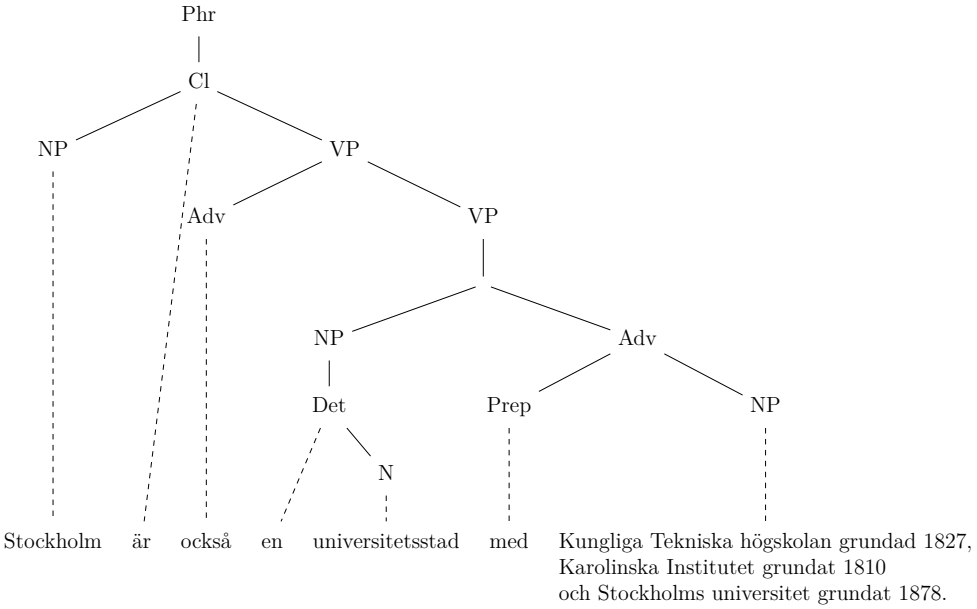


Figure 3.4: Syntax tree for the second sentence.

```
Line2 urban_area universities =
    let ua_NP : NP = str_PN_to_NP urban_area ;
    in mkPhr (mkCl ua_NP (mkVP also_Adv (mkVP (mkNP
        (mkNP a_Det universitycity_N)
        (Line2_1 universities))))) ;
```

An NP is first created out of the run-time argument string representing the name of the urban area. Then the `mkCl : NP -> VP -> Cl` function is used to create a clause, using the name as the noun phrase. Since the second argument is a VP the copula is not automatically applied to the arguments and instead the verb used is whatever verb is the head verb

in the VP. The VP itself is created with the function `mkVP : Adv -> VP -> VP` which creates a verb phrase out of an adverb and a verb phrase. The second parameter is then a verb phrase, which in this case is created with the line of code below.

```
(mkVP (mkNP (mkNP a_Det universitycity_N)
            (Line2_1 universities)))
```

Here the function `mkVP : NP -> VP` is used to create a verb phrase out of a noun phrase. Since no verb is supplied, the copula is automatically applied to the noun phrase. The noun phrase itself is built out of the noun phrase ”en universitetsstad” (a university city) and the adverbial phrase returned by the function `Line2_1`. `Line2_1` is a helper function which takes a `ListNP` (a list of all universities as noun phrases) as its argument and uses the `mkNP : Conj -> ListNP -> NP` function to transform the list of noun phrases into one single noun phrase *.

```
Line2_1 universities = mkAdv
                           with_Prep
                           (mkNP and_Conj universities)
```

The noun phrase below (which corresponds to the ”universities” variable above)

> *Kungliga Tekniska högskolan grundat 1827, Karolinska Institutet grundat 1810 och Stockholms universitet grundat 1878.*

is created by first querying Wikidata for all universities existing in the urban area, and then putting them in a Python dictionary, where each element is a tuple where the first element in the tuple is the university name and the second element in the tuple is the date the university was founded. Then the dictionary is iterated through and the elements contained in each tuple is used as input to the function `create_university : Str -> Str -> NP`, which takes a university name as a string and year as a string as input and then returns a noun phrase in the form *university founded in year*. For each iteration, a new noun phrase is created and appended to a list. The first two GF trees in this list are then used as input arguments for the function `create_universities_list: NP -> NP -> ListNP` which creates a

* The phrase tree 3.4 does not have that step explicitly written out in order to make it more readable

`ListNP` out of two noun phrases. If there are more than two trees in the list the function `add_university: NP -> ListNP -> ListNP` is used to add additional noun phrases to the list of noun phrases.

Third Sentence
The third sentence, *Linköping är Sveriges åttonde största tätort med över 115 000 invånare*, (Linköping is Sweden's eighth largest urban area with more than 115,000 inhabitants in English), is not broken down into smaller parts, since the structure is simple and there is no varying properties present. The phrase structure tree can be seen below:

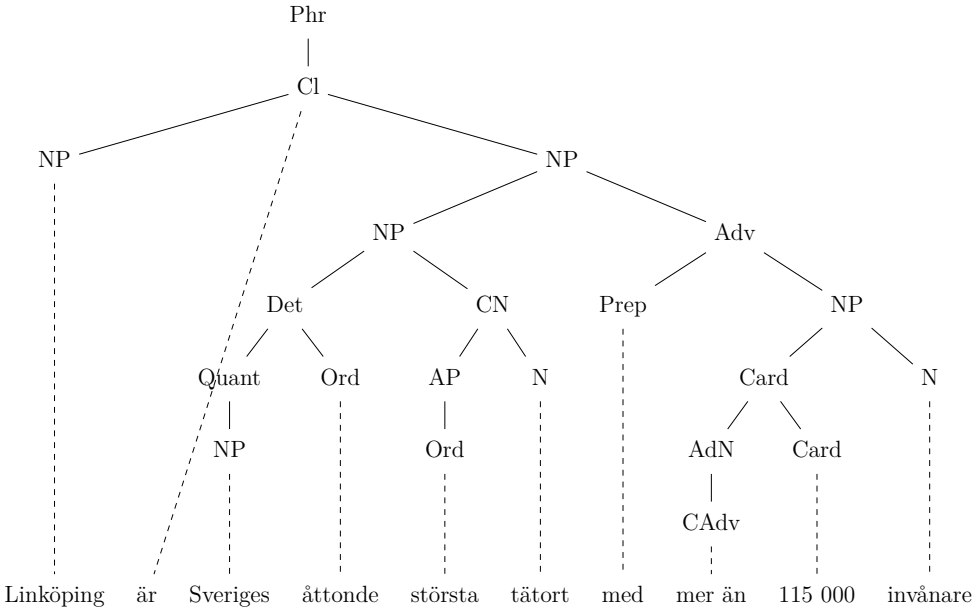


Figure 3.5: Syntax tree for the third line

What is needed as input from Python is a number that denotes the rank of the population in relation to the population size of the other urban areas, as well as a number representing the population size. In the case of Linköping, the numbers 8 and 115 000 are needed. RGL functions of arities higher than 1 don't use strings as input arguments, and the RGL functions with arity 1 that do only serves the purpose of transforming the string into a GF type. This means that the input needs to be transformed into GF types before being usable in the code (as can be seen in previous sections when transforming strings representations of names into noun phrases). In order to create *åttonde* (eighth) out of the string "8" (which

is calculated in Python), normally the RGL function `mkNumeral : Str -> Numeral` would first be applied to the string, and then the function `mkOrd : Numeral -> Ord` would be applied to that result. But GF does not allow for pattern matching run-time strings [31], and looking into the implementation of the `mkNumeral : Str -> Numeral` function in the source code of GF reveals that the function pattern matches the input argument. This is a recurring problem that needs to be handled in different ways depending on the situation and what options are available. What seems to be the easiest way to get around this is to define a set of GF functions in the grammar that returns numerals: `mkNumerali` up to `mkNumeraln` where in this case i is 2 (there is no need for an ordinal when describing the largest urban area) and n is 10 (10 is chosen as a cut-off, since at the higher numbers including the ordinal doesn't sound as good, *e.g., thirteenth largest*). Then, depending on the rank, one of those GF functions is used to create a GF numeral in Python which is then used as input to the sentence function. For Linköping, the rank is 8, and so the function `mkNumeral8` is used which returns a numeral. This is made possible with the python line

```
rank_gf = pgf.Expr('mkNumeral{}'.format(rank),[])
```

where the PGF `Expr` method is used to create a GF tree from a GF function, and where the GF function name is created by using the built-in Python `format` method to insert the supplied argument inside of the curly brackets in the string object it is called on. And so the name of the GF function that is called on changes depending on the rank variable.

The sentence function is seen below:

```
Line3 city rank population =
mkPhr (mkCl (str_PN_to_NP city)
(mkNP
    (mkNP
        (mkDet (GenNP (mkNP sverige_PN)) (mkOrd rank))
        (mkCN (mkAP (mkOrd large_A)) urbanarea_N))
    (mkAdv with_Prep (mkNP (mkCard (mkAdN more_CAdv)
                                   (mkCard population))
                           inhabitants_N))));
```

As before, the `mkCl : NP -> NP -> Cl` function is used to apply the copula to two noun phrases. The second argument is a noun phrase

created by a noun phrase and an adverbial phrase. This noun phrase, *Sveriges åttonde största tätort*, is created out of a determiner, *Sveriges åttonde*, and a common noun, *största tätort*. The determiner is further broken down into smaller components as seen in 3.5. The determiner consists of a quantifier*Sveriges*, and an ordinal number *åttonde*. The quantifier is created by applying the RGL function `GenNP : NP -> Quant` on the noun phrase *Sverige*. The ordinal is created by applying the RGL function `mkOrd : Numeral -> Ord` on the parameter rank. Rank is a GF numeral created in Python from the number representing the population of the urban area in relation to other urban areas in Sweden, and used as input to the sentence function. The quantifier and ordinal can then be combined into a determiner as seen in the syntax tree 3.5.

Since no RGL function takes two ordinals as input, it is not possible to directly combine *åttonde* (eighth) and *största* (largest) with each other, and so instead the first ordinal becomes part of the determiner, while as can be seen in 3.5, the second ordinal is transformed into an adjectival phrase and together with a noun made into a common noun with the `mkCN : AP -> N -> CN` function. The determiner and common noun are then used as arguments to the function `mkNP : Det -> CN -> NP`, creating the noun phrase *Sveriges åttonde största tätort* (Sweden's eighth largest urban area).

The sentence *med mer än 155 000 invånare* is created as an adverbial phrase, where *med* (with) is a preposition, and *mer än 115 000 invånare* is a noun phrase. In order to express *mer än 115 000* purely through the RGL, the function `mkCard : AdN -> Card -> Card` is used. This function creates a cardinal number out of an AdN (numeral modifying adverb) and a cardinal (a cardinal in GF will only consists of letters up to 999, hence 115 000). In order to get a AdN expressing "more than", the RGL-defined comparative adverb (CAdv) `more_CAdv : CAdv` is used. Together with the function `mkAdN : CAdv -> AdN`, it is transformed to an AdN and used as input for the `mkCard : AdN -> Card -> Card` function:

```
(mkCard (mkAdN more_CAdv) (mkCard population))
```

The noun phrase *Sveriges åttonde största tätort* and the adverbial phrase *med mer än 155 000 invånare* is combined into the noun phrase *Sveriges åttonde största tätort med mer än 155 000 invånare*. Using the `mkCl :`

`NP -> NP` function where the first argument is the name of the urban area, in this case Linköping, the result is *Linköping är Sveriges åttonde största tätort med över 115 000 invånare.*

There are two more sentence functions used:

`Line3_largest` which is only used when the rank of the city is 1. It takes an urban area name, x, and its population amount, y, and returns the sentence *X is Sweden's largest urban area with more than y inhabitants.*. This GF function will only be chosen in Python if the rank is 1, meaning it is only used when the urban area in question is the largest.

`Line3_norank` which is only used when the rank is higher than 10. It takes an urban area name, x, and its population amount, y, and returns the sentence *there are more than y inhabitants in x.*.

3.2.3.3 English Concrete Syntax

The English concrete syntax is practically identical to the Swedish concrete syntax. This could be expected as the languages are fairly similar grammatically for the text chosen.

3.2.3.4 French Concrete Syntax

The RLG functions and their order in the the French concrete syntax is similar to the other concrete syntaxes as well. GF frees the user from needing to think about grammatical rules such as gender agreement, and so in this case, only the words need to be changed.

One difference is the genitive case, which can be created in Swedish and in English by appending "'s" to the end of a noun phrase [32], but expressed in French with the preposition *de* (of) [33]:

Linköping est l'huitième plus grande zone urbaine de Suède avec plus que 115000 habitants.

Therefore, the implementation of the function creating the sentence *Linköping är Sveriges åttonde största tätort med över 115 000 invånare* in the French concrete syntax will look different compared to the implementation in the Swedish and English concrete syntax. Instead of making the proper name *Suède* (Sweden) into the genitive case with the `GenNP` function, the `mkAdv` function is used to create the adverbial phrase *de Suède* with

```
mkAdv genitive (mkNP (mkPN "Suède"))
```

The syntax tree of how the sentence should be built in GF can be seen below:

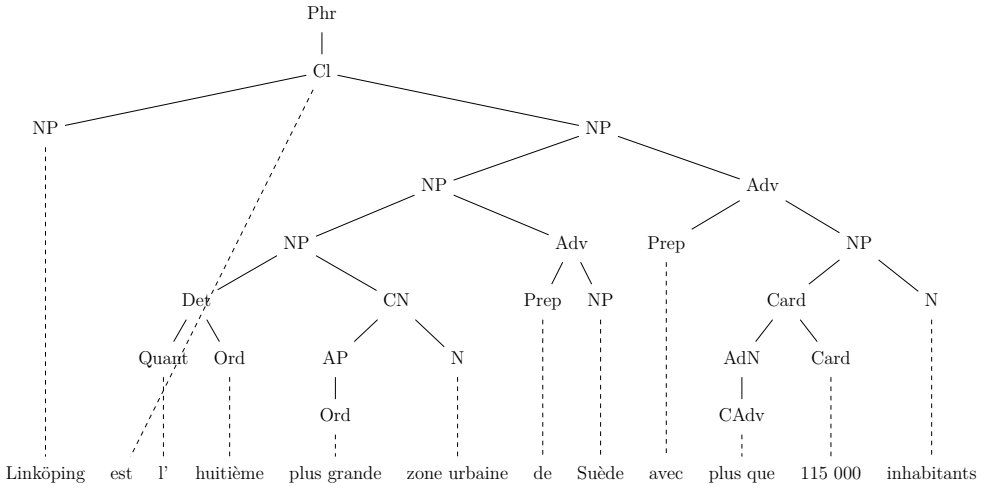


Figure 3.6: Syntax tree for last sentence in French.

3.2.4 Grammatical Framework and Python

In order to fully understand the design of the grammar, one needs to look at it in the context of how the run-time arguments for the grammar is decided in Python.

Using Run-time Strings in GF

GF does not allow for pattern-matching run-time strings, which can be an issue since the implementations of many RGL functions make use of pattern-matching in some way. This can be hard to work around when one language paradigm uses pattern matching while the rest does not. For example, the `mkPN : Str -> PN` implementation in the French paradigm library uses pattern matching to see if a string ends with an ”e” in order to assign it the correct gender. The fix here was to do the same pattern matching in Python and then use the `mkPN : Str -> Gender -> PN` function to set the gender manually and create a GF tree to be used as input instead of a string. To keep the other concrete syntaxes unaffected by this, a new type called `PN_String` is introduced. In the French syntax, it is defined to be a `PN`, while in the Swedish and English

concrete syntax it is defined to be a string. So when the function for creating this `PN_String` is used in Python, the implementation in the Swedish and English syntax will take a string and just return that string, while the implementation in the French syntax will take the string and return a `PN`. In other words, the `PN_String` is linearised into different types depending on the concrete syntax.

Another example is the issue with numbers. GF has several functions for making a string into a number type such as `Digit` or `Numeral`. The problem is that all of these functions use pattern matching, thus preventing the user from converting the string into a number. One workaround is to use predefined GF types to recursively build a number out of string input. The method and the code for implementing this are shown in appendix B.

Chapter 4

Results

4.1 Results from Generating 100 Urban Areas

The result of the experiment is a document containing the result of applying the program outlined in chapter 3 to generate texts in Swedish, English and French for 100 different urban areas. The 40 most populated urban areas were chosen along with 60 randomly chosen urban areas.

A completely correct example without any spelling or grammatical mistakes is the result for the urban area Malmö in Swedish:

> `Malmö är en tätort i Skåne län samt centralort i Malmö kommun och residensstad i Skåne län. Malmö är också en universitetsstad med Malmö universitet grundat 1998. Malmö är Sveriges tredje största tätort med mer än 325000 invånare.`

The English generation almost produced a correct result:

> `Malmö is a urban area in Skåne County as well as the seat of Malmö Municipality and the capital of Skåne County. Malmö is also a university city with Malmö universitet founded in 1998. Malmö is Sweden's third largest urban area with more than 325,000 inhabitants.`

The incorrect part is the article referring to an urban area. It should

be ”an urban area” and not ”a urban area”. One of the strengths of the resource grammar library is to know which form an article would be in if placed in front of a specific noun. This information (”a” or ”an” in this case) is encoded in the noun when it is created through the `mkN` function. But in this case, the result is wrong.

The French generation had a grammatical mistake as well:

```
Malmö est une zone urbaine dans le comté de Skåne et
la siège dans la commune de Malmö et le chef-lieu dans
le comté de Skåne. Malmö est aussi une ville universitaire
avec Malmö universitet fondé en 1998. Malmö est la
troisième zone urbaine plus grande de Suède avec plus
que 325000 habitants.
```

The mistake is in the last sentence where the adjectival phrase ”plus grande” is put after the noun zone urbaine. In French, adjectives are commonly placed after nouns, but there are some special adjective that is placed before the nouns they are describing, one of them being ”grande”. In the French Paradigm there exists a function `prefixA : A -> A` which takes an adjective and transforms it into an adjective that when used with a noun will be placed before the noun instead of after. The code

```
mkCN (prefixA (mkA "grand")) (mkN "zone urbaine)
```

generates the sentence *grande zone urbaine*, the adjective being put in front of the noun (as well as agreeing on gender).

The code

```
mkCN (mkAP (mkOrd (prefixA (mkA "grand"))))
     (mkN "zone urbaine)
```

should generate *plus grande zone urbaine* but instead *zone urbaine plus grande* is generated, this time the adjective being placed after the noun instead of in front of it.

There might be something that is missing from the code to produce the desired result, or it can simply just be a bug.

There is also something to be said about leaving the university names unchanged, which is especially noticeable in the French text. Leaving the university names unchanged simplifies a lot of things but does actually impact the result. The consequences can be seen in the second sentence

in the French text, *Malmö est aussi une ville universitaire avec Malmö universitet fondé en 1998.*, where had the university name been in French instead (*Universitée de Malmö* it would've been recognised by GF to have a feminine gender, and thus the verb fonder following it would have been inflected into fondée instead of fondé. A similar thing can be seen when looking at the Swedish text for the generated text about Stockholm:

```
Stockholm är en tätort i Stockholms län samt centralort
i Stockholms kommun, residensstad i Stockholms län och
stiftsstad för Stockholms stift. Stockholm är också
en universitetsstad med Kungliga Tekniska högskolan
grundat 1827, Karolinska Institutet grundat 1810 och
Stockholms universitet grundat 1878. Stockholm är Sveriges
största tätort med mer än 1600000 invånare.
```

The second sentence describes all the universities in Stockholm. It should say *Kungliga Tekniska högskolan grundad 1827* and not *Kungliga Tekniska högskolan grundat 1827*. In this case the verb ”grunda” is inflected according to the object which is the university, but the word ”högskola” should have the utrum gender and so the verb should be inflected with regards to the utrum gender, resulting in ”grundad”.

Looking at one of the random results

```
Glömminge är en tätort i Kalmar län. det finns mer
än 885 invånare i Glömminge.

Glömminge is a urban area in Kalmar County. there are
more than 885 inhabitants in Glömminge.

Glömminge est une zone urbaine dans le comté de Kalmar.
il y a plus que 885 habitants dans Glömminge.
```

Cross-checking the Wikipedia page för Glömminge shows that the latest recorded population size of Glömminge is precisely 885. Therefore the second sentence in all three languages is factually incorrect.

Another randomly chosen urban area is Forsvik:

```
Forsvik är en tätort i Västra Götalands län. det finns
mer än 320 invånare i Forsvik.

Forsvik is a urban area in Västra Götaland County. there
are more than 320 inhabitants in Forsvik.
```

```
Forsvik est une zone urbaine dans le comté de Västra
Götaland. il y a plus que 320 habitants dans Forsvik.
```

According to its Wikipedia page, the latest recorded population size is 323. It is also indeed located in Västra Götaland county. This means that the Swedish and French texts are factually and grammatically correct, while the English sentence still suffers from having the wrong article in front of ”urban area”.

Below is a table showing the number of grammatical mistakes as well as any factual mistakes for the eight largest urban areas and the three smaller urban areas Glömminge and Forsvik which were both mentioned before, and the randomly chosen urban area Långared. The rest of the (smaller) urban areas are left out of the table since the results are the same for most of them.

Urban area	**Grammatical mistakes in Swedish**	**Grammatical mistakes in English**	**Grammatical mistakes in French**	**factual mistakes**
Stockholm	1	1	1	1
Göteborg	1	1	1	0
Malmö	0	1	2	1
Uppsala	0	1	2	0
Upplands Väsby och Sollentuna	0	1	1	0
Västerås	0	1	2	0
Örebro	0	1	1	1
Linköping	0	1	2	0
Glömminge	1	2	1	1
Forsvik	1	2	1	0
Långared	1	2	1	0

Table 4.1: An overview of the number of grammatical and factual mistakes for each generated text

Chapter 5

Discussion

This section contains a discussion on the results presented in chapter 4 as well as a discussion on topics related to the construction of the grammar.

5.1 Text Generation with Grammatical Framework

While having some grammatical mistakes, the results show that it is possible to use Grammatical Framework to generate fluent and coherent text using data from Wikidata. This is true, especially for the language in which the data is in, since the queried data such as names of locations are already in their correct form, thus not needing as much pre-processing.

The goal of the Abstract Wikipedia project is to be able to share information in more languages, and this project shows that GF is a tool which has the potential to reach that goal by successfully generating texts in different languages using data from Wikidata. But even if it is possible to further develop the program and fix the issues observed in the results, the sheer amount of time needed for the construction of such a program might call into question the use of GF.

Since the main use of the resource grammar library is to alleviate the programmer of needing to think about gender, inflection, and other grammatical structures and operations, the construction of such

a program presented in this thesis would at first glance seem relatively straightforward and simple. In theory, this is true, but by adding the data from Wikidata and the constraint on pattern-matching run-time strings into the mix, the construction of the program can get complicated and tedious. The data is easy to use when it is in the same language as the concrete syntax but requires much more pre-processing if it is not. And since GF uses the same AST for all languages, the tree creation process happens one time and so there can only be one type of input, meaning that the input to the GF functions are the same for each language. If one of the languages has a different writing system, this will be a big problem. Take Japanese as an example: if the result from a query was ”Stockholms universitet” then this could never be used as input for a Japanese text. It would not be desirable to mix two very different writing systems in such a way. One approach to solving issues like this is to add functions in each concrete syntax that contains the translation of names from the language the data is into the language of the concrete syntax. While it can seem that such a solution means regressing back to using hand-written templates, GF and the strength of its resource grammar library can still be used to perform more efficient text generation. An example is the Swedish version of the second sentence where the verb needs to agree on the gender of the proper name (university name) it is preceded by. The only thing that GF needs the programmer to do is to assign the correct gender to the name of the university. In a hand-written template, just determining the gender of the university name would not suffice; additional code would be needed to be written for choosing the correct verb form based on the preceding noun's gender. One can see how this could quickly become a lot of work when generating longer texts.

Even though the issues described can be frustrating, GF makes for an excellent choice when performing natural language generation. This is, again, mainly because of its resource grammar library. However, GF is also used to translate between languages but it is not as easy as advertised. The programmer needs to have some sort of knowledge about the language that is being implemented. *e.g.,* if the programmer spoke English and wanted to generate the noun phrase *Stockholms universitet founded in 1960* in Swedish, the programmer would need to know that in this case, no preposition is used between ”founded” and ”1960”. Thus it is not as simple as just switching libraries. Instead, an accurate syntax tree needs to be built up using RGL functions and/or self-written GF

functions. And even switching out words need to be done carefully, since words can mean different things in different contexts and so just directly translating between words might produce text that can sound unnatural to a native person. Based on the points made in this paragraph, being fluent or having a fluent person at hand seems to be not just the most optimal way but the only way to ensure that the text produced is fluent and natural sounding.

5.2 Wrong article in front of ”urban area”

One apparent grammatical mistake that was present in all texts generated in English was that the article referring to urban area was in the wrong form. It should've been ”an” instead of ”a”. Trying other nouns starting with a ”u” in the GF shell yields mixed results; for some words such as ”underdog” or ”uncle” the correct article is produced, while the wrong article is produced for *e.g.,* for the word ”umbrella”. Upon further investigation, it seems that the word ”urban area” is an exception to the rule that handles the choice of which article to use in the source code. The solution to this would be to add a pattern which handles the case when the word is exactly ”urban area”.

Even though it can be assumed that the majority of words will have the correct articles assigned to them, for these experiments this small grammatical mistake lowered the quality of the whole text.

5.3 University Names

As could be seen in the results, the names of the universities were not processed like the names of municipalities or counties. Therefore they did not go through the same process of having their string representation transformed first into a proper name then a noun phrase. One might ask why not? It was something that was unsuccessfully attempted. There is no clear reason to why it failed since the same process worked well for the sentence before when applying the `str_PN_NP` operation to the runt-time strings. From the error message, it has something to do with the string being a run-time string. But how to handle the problem is not clear and so another solution was used where the `symb` function from the `symbolic` module was used, which transforms the string into a noun

phrase. From the results, it seems the university names' gender was utrum in the Swedish texts and masculine in the French texts.

To actually have the correct verb inflection after the university name the same process of pre-processing municipality and county names would need to be applied to the university names after querying for them in Python. One solution could then be to look at what word the university name is ending with, then use whatever the string ends with as a key in a pre-made dictionary that maps an ending with a gender to acquire the correct gender to be used as an argument when creating a string. For example, when processing "Stockholms universitet" it ends with "universitet" and so that is used as a key in the dictionary containing the element `"universitet" : neutrum` and then a GF tree is created with the function `mkPN : Str -> Gender -> PN` with "Stockholms universitet" and neutrum as input.

The solution above only works for Swedish since the university names are all in Swedish. Since one might need to take other languages into account as well, a different approach would be to trim the word down to only include the location (*e.g.,* Stockholms universitet => Stockholm) and then add the appropriate words using GF functions which are implemented differently for each language. For example since the name "Stockholms universitet" ends in "universitet", a GF function specific for the case when the name ends with "universitet" could be called in Python with "Stockholm" as an argument. This function would then have a different implementation for each language. But there will be cases where this can get tricky as well, such as in the case of the university name "Uppsalas lantbruksuniversitet" where there is not only the location "Uppsala" but also the noun "lantbruk" (agriculture) that needs to be taken in account for.

Since there are not too many universities in Sweden, a specific but 100% accurate solution would be to have a function for each university returning the correct name. This means that 16 functions have to be created for each language. Though a solution like this could be seen as too manual if *e.g.,* this had to be done for each university in the US. Overall the goal is to do as less manual work as possible with the help of tools like GF, otherwise simple hand-written templates could be used instead.

There is also a script on Github that lets the user download the same

names in different languages from Wikidata and then from the names create separate grammars. These grammars can then be used to create trees out of input names which then linearise into the target language. Using this method and extending it might be the best solution to the problem since the automated creation of grammars save time and also for it being a solution more in line with GFs methodology.

5.4 Adding a Language

Adding a language gets harder the more complex the text is. Basic sentences can be verified through tools like Google Translate, but it would probably not be enough. A good example is what was shown in the result where the superlative *plus grande* was put behind the noun and not in front of it. These are intrinsic rules that do require the programmer to have had some experience with the language they are implementing. Languages similar to the ones one knows can still be successfully implemented if the sentences are simple enough. If the programmer knows Swedish, English, and French, adding Spanish to the scope of this thesis would not be too complicated. Implementing languages from other language families or with different writing systems would be a much harder challenge.

5.5 Universal Dependencies Trees as Formal Representations for Grammatical Framework

A smaller part of this thesis was to investigate how Universal Dependency trees from Wikipedia could be used as formal representation input for Grammatical Framework. It was concluded that this investigation was too insignificant to have in the results section, but significant enough to at least be discussed.

In order to investigate how Universal Dependency trees from Wikipedia could be used as formal representation input for Grammatical Framework, the program gf-ud[34] was used to create GF type signatures out of UD trees and their respective sub-trees. Presented below is an explanation of that process, as well as a discussion on how and if this

method will have any practical relevance.

The observation was the comparison of two short sentences where the only difference was in the definiteness of the determiner and the noun following it:

```
Linköping är en tätort i Östergötland
```

and

```
Linköping är den tätorten i Östergötland
```

with their corresponding UD trees in the .conllu format, respectively:

```
1   Linköping Linköping PROPN PM|NOM Case=Nom 4 nsubj _
    start_char=0|end_char=9|ner=S-LOC
2   är vara AUX VB|PRS|AKT Mood=Ind|Tense=Pres|VerbForm=Fin|
    Voice=Act 4 cop _ start_char=10|end_char=12|ner=O
3   en en DET DT|UTR|SIN|IND Definite=Ind|Gender=Com|Number=Sing|
    PronType=Art 4 det _ start_char=13|end_char=15|ner=O
4   tätort tätort NOUN NN|UTR|SIN|IND|NOM Case=Nom|Definite=Ind|
    Gender=Com|Number=Sing 0 root _ start_char=16|end_char=22|
    ner=O
5   i i ADP PP _ 6 case _ start_char=23|end_char=24|ner=O
6   Östergötland Östergötland PROPN PM|NOM Case=Nom 4 obl _
    start_char=25|end_char=37|ner=S-LOC
```

```
1   Linköping Linköping PROPN PM|NOM Case=Nom 4 nsubj _
    start_char=0|end_char=9|ner=S-LOC
2   är vara AUX VB|PRS|AKT Mood=Ind|Tense=Pres|VerbForm=Fin|
    Voice=Act 4 cop _ start_char=10|end_char=12|ner=O
3   den en DET DT|UTR|SIN|DEF Definite=Def|Gender=Com|Number=Sing|
    PronType=Art 4 det _ start_char=13|end_char=16|ner=O
4   tätorten tätort NOUN NN|UTR|SIN|DEF|NOM Case=Nom|Definite=Def|
    Gender=Com|Number=Sing 0 root _ start_char=17|end_char=25|
    ner=O
5   i i ADP PP _ 6 case _ start_char=26|end_char=27|ner=O
6   Östergötland Östergötland PROPN PM|NOM Case=Nom 4 obl _
    start_char=28|end_char=40|ner=S-LOC
```

Creating GF type signatures from the UD trees using gf-ud returns the exact same two type signatures shown below:

```
 (["PROPN(nsubj) -> AUX(cop) -> DET(det) -> NOUN(head) ->
    PROPN(obl) -> NOUN(head)"],1)
(["ADP(case) -> PROPN(head) -> PROPN(head)"],1)
```

The second type signature refers to *i Östergötland*, while the first one refers to the whole sentence. The idea would be to build the whole sentence using the type signatures by implementing them with the appropriate GF and RGL functions.

Going back to UD trees, the difference is not just the determiners and nouns but also in the value of the feature `Definite` where for the first line, the value of `Definite` for the determiner ”en” is `Ind` meaning that it is an indefinite article, and for the second line the value of `Definite` for the determiner ”den” is `Def` meaning that it is a definite article. This same difference can be observed in the words ”tätort” and ”tätorten”.

Now that this difference is established, the reason why the type signatures are identical becomes apparent: The GF determiner type, `Det`, does not carry any specific knowledge about what type of determiner it is. And so when converting the UD trees to GF type signatures, that additional information is lost. Thus it will be a challenge for an automated algorithm to know what type of RGL functions to use to create the right sentence because of ambiguities like these.

Another problem is that it is not just a matter of replacing names and numbers to get a new article; a sentence describing two universities cannot be used to describe one university just by swapping data, the program also needs to take into account the number of universities. It is then fair to conclude that an algorithm automating the process just outlined in this section will not yield desirable results compared to the program presented in this thesis.

5.6 Ethics and Sustainability

Using grammar-based tools like GF for natural language generation is more sustainable compared to popular approaches such as neural networks which consume large amounts of energy to train models on data.

Since having a human inspecting every article generated would defeat the purpose of this kind of automation, one ethical implication would be

the correctness of the facts presented in the generated texts. To solve this, Wikipedia could place a banner at the top of the article notifying the user that the current article has been generated by a program and not written by a human.

Manually written articles also have their issues; they are written by humans and so one must take into account that the writer's biases and amount of knowledge about the topic being written about will have ethical implications as well. To not use automated pages because of the risk of incorrect facts means that humans need to write these pages. It might be hard to motivate someone to write about 2000 Swedish urban areas for the Turkish Wikipedia, whereas a program such as the one presented in this case study can generate those articles instantly. This means more information for more people. The concern of incorrect facts is a valid one, but if it can be shown that a large percentage of the articles will come out factually correct, then not using the automated approach could be seen as unethical since that decision would mean that a potentially large amount of people would miss out on information.

Chapter 6

Conclusions and Future work

6.1 Conclusions

While the generated texts were not perfect and without errors, the experiments showed that it is possible to use a structure of a Wikipedia article and using Grammatical Framework to generate other articles with the same structure for different but related topics.

The results showed that the generation was not perfect, and most texts contained at least one grammatical mistake. In some cases, they contained factual mistakes as well. But these issues can be fixed with some time and experience, and overall the results showed that GF is indeed a tool which can be used to successfully perform natural language generation.

The biggest advantage of using Grammatical Framework is that it relieves the programmer from needing to pay attention to tasks related to grammar such as inflection and gender agreement, which ends up saving a lot of time and effort. The biggest challenges were how to apply GF functions to run-time strings, and how to pre-process the data.

6.2 Future work

There are several ways to expand on this project. In order to create more fluent text that looks like current articles on Wikipedia, a first step would be to go into more detail regarding the locations of the urban

areas. Many articles about Swedish urban areas mention the location in relation to some geographical landmarks. In the Wikipedia article about Linköping, it is stated that Linköping is situated on *Östgötaslätten*, one of many plains in Sweden. So the next step could be to extend the text generation to express the location of urban areas geographically. It seems that Wikidata contains coordinates for most urban areas and so one way would be to use the coordinates to find the closest geographical landmark. For example, it is written in the article about Stockholm that Stockholm is situated next to the lake *Mälaren*.

Another way to expand on the project would be to fix the issues presented in the results and discussion and then add a language which does not have entries about Swedish urban areas on its Wikipedia. The results from the generation could be used to populate that Wikipedia.

Automating the process of creating GF functions from UD trees is another thing that could be worth doing. The focus should first be on creating proper GF functions out of UD trees, and not so on needing to generate proper text with the GF functions. The idea would be to take a text about an urban area, take its UD tree and then run it through the program and get a set of GF functions; these GF functions can then be used as templates when writing the grammar. This would dramatically decrease the amount of time needed to write grammars.

The biggest limiting factor in all of this is Wikidata, which does not contain much information. Most of the information available has already been used in this project.

References

[1] A. Gatt and E. Krahmer, "Survey of the State of the Art in Natural Language Generation: Core tasks, applications and evaluation," *Journal of Artificial Intelligence Research*, vol. 61, pp. 65–170, Jan. 2018. doi: 10.1613/jair.5477. [Online]. Available: https://www.jair.org/index.php/jair/article/view/11173 [Pages i, 1, 9, and 10.]

[2] A. Ranta, *Grammatical Framework: Programming with Multilingual Grammars*. Stanford: CSLI Publications, 2011, iSBN-10: 1-57586-626-9 (Paper), 1-57586-627-7 (Cloth). [Pages i, 2, 11, 12, 14, 15, and 16.]

[3] "GF Resource Grammar Library: Synopsis." [Online]. Available: https://www.grammaticalframework.org/lib/doc/synopsis/index.html [Pages i and 32.]

[4] "Wikipedia:about," 2022, [Online; accessed 7-April-2022]. [Online]. Available: https://en.wikipedia.org/wiki/Wikipedia:About#Uses [Page 1.]

[5] Meta, "List of Wikipedias — meta, discussion about Wikimedia projects," 2022, [Online; accessed 7-April-2022]. [Online]. Available: https://meta.wikimedia.org/w/index.php?title=List_of_Wikipedias&oldid=23110242 [Page 1.]

[6] ——, "Abstract wikipedia — meta, discussion about wikimedia projects," 2021, [Online; accessed 17-December-2021]. [Online]. Available: https://meta.wikimedia.org/w/index.php?title=Abstract_Wikipedia&oldid=22459473 [Page 1.]

[7] W. Ng' ang' a, "Building Swahili Resource Grammars for the Grammatical Framework," in *Shall We Play the Festschrift Game?* Springer, 2012, pp. 215–226. [Page 2.]

[8] L. Pretorius, L. Marais, and A. Berg, "A GF Miniature Resource Grammar for Tswana: Modelling the Proper Verb," *Lang. Resour. Eval.*, vol. 51, no. 1, p. 159‑189, mar 2017. doi: 10.1007/s10579-016-9341-z. [Online]. Available: https://doi.org/10.1007/s10579-016-9341-z [Page 2.]

[9] "Catholic Dictionary, EPISCOPAL SEE," [Online; accessed 03-September-2022]. [Online]. Available: https://www.catholicculture.org/culture/library/dictionary/index.cfm?id=33355 [Page 2.]

[10] D. Jurafsky and J. H. Martin, *Chapter 12*. [Online]. Available: https://web.stanford.edu/~jurafsky/slp3/ [Pages 5 and 6.]

[11] J. Boye, "Lecture notes in language engineering." [Page 5.]

[12] A. Meduna, *Formal Languages and Computation : Models and Their Applications*, first edition.. ed., 2014. ISBN 0-429-09859-6 [Page 6.]

[13] R. Debusmann, "An introduction to dependency grammar," *Hausarbeit fur das Hauptseminar Dependenzgrammatik SoSe*, vol. 99, pp. 1–16, 2000. [Page 7.]

[14] D. Jurafsky and J. H. Martin, *Chapter 14*. [Online]. Available: https://web.stanford.edu/~jurafsky/slp3/ [Pages ix and 7.]

[15] A. Ranta, "Lecture 2: Abstract and Concrete Syntax." [Online]. Available: http://www.cse.chalmers.se/edu/year/2011/course/TIN321/lectures/proglang-02.html [Page 8.]

[16] J. Hirschberg and C. D. Manning, "Advances in natural language processing," *Science*, vol. 349, no. 6245, pp. 261–266, 2015. [Page 9.]

[17] [Online]. Available: https://universaldependencies.org/introduction.html [Page 11.]

[18] M.-C. de Marneffe, C. D. Manning, J. Nivre, and D. Zeman, "Universal Dependencies," *Computational Linguistics*, vol. 47, no. 2, pp. 255–308, 07 2021. doi: 10.1162/coli_a_00402. [Online]. Available: https://doi.org/10.1162/coli_a_00402 [Page 11.]

[19] J. Nivre, M.-C. de Marneffe, F. Ginter, J. Hajič, C. D. Manning, S. Pyysalo, S. Schuster, F. Tyers, and D. Zeman, "Universal Dependencies v2: An evergrowing multilingual treebank collection," in *Proceedings of the Twelfth Language Resources and Evaluation Conference*. Marseille, France: European Language Resources Association, May 2020. ISBN 979-10-95546-34-4 pp. 4034–4043. [Online]. Available: https://aclanthology.org/2020.lrec-1.497 [Page 11.]

[20] J. Nivre, M.-C. de Marneffe, F. Ginter, Y. Goldberg, J. Hajič, C. D. Manning, R. McDonald, S. Petrov, S. Pyysalo, N. Silveira, R. Tsarfaty, and D. Zeman, "Universal Dependencies v1: A Multilingual Treebank Collection," in *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*. Portorož, Slovenia: European Language Resources Association (ELRA), May 2016, pp. 1659–1666. [Online]. Available: https://aclanthology.org/L16-1262 [Pages ix and 11.]

[21] A. Ranta, "Grammatical Framework," *Journal of Functional Programming*, vol. 14, no. 2, pp. 145–189, 2004. [Page 11.]

[22] A. Ranta, K. Angelov, N. Gruzitis, and P. Kolachina, "Abstract Syntax as Interlingua: Scaling Up the Grammatical Framework from Controlled Languages to Robust Pipelines," *Computational Linguistics*, vol. 46, no. 2, pp. 425–486, Jun. 2020. doi: 10.1162/coli_a_00378. [Online]. Available: https://doi.org/10.1162/coli_a_00378 [Pages 18 and 23.]

[23] T. Narahara, *The Japanese copula*. Springer, 2002. [Online]. Available: https://doi.org/10.1057/9780230504530 [Page 19.]

[24] A. Ranta, "The GF Resource Grammar Library," *Linguistic Issues in Language Technology*, vol. 2, Dec. 2009. doi: 10.33011/lilt.v2i.1205. [Online]. Available: https://journals.colorado.edu/index.php/lilt/article/view/1205 [Page 20.]

[25] "Paradigms for swedish." [Online]. Available: https://www.grammaticalframework.org/lib/doc/synopsis/index.html#toc121 [Pages xi and 20.]

[26] A. Ranta, "Natural Language Generation with the Grammatical Framework," August 2021. [Online]. Available: https://github.com/aarneranta/NLG-examples/blob/main/doc/gf-nlg.pdf [Page 22.]

[27] K. Angelov, "Using the Python Haskell Java C# binding to the C runtime," 2017, [Online; accessed 8-August-2022]. [Online]. Available: http://www.grammaticalframework.org/doc/runtime-api.html#python [Page 22.]

[28] "Lsjbot." [Online]. Available: https://en.wikipedia.org/wiki/Lsjbot [Page 23.]

[29] "The World's Second Largest Wikipedia Is Written Almost Entirely by One Bot," [Online; accessed 23-February-2023]. [Online]. Available: https://www.vice.com/en/article/4agamm/the-worlds-second-largest-wikipedia-is-written-almost-entirely-by-one-bot [Page 23.]

[30] "SPARQL." [Online]. Available: https://en.wikipedia.org/wiki/SPARQL [Page 27.]

[31] "Programming in GF: tips and gotchas." [Online]. Available: https://inariksit.github.io/gf/2018/08/28/gf-gotchas.html [Page 39.]

[32] C. Lyons, "The syntax of English genitive constructions," *Journal of Linguistics*, vol. 22, no. 1, p. 123-143, 1986. doi: 10.1017/S0022226700010586 [Page 41.]

[33] P. Ten Hacken, "Compounds in English, in French, in Polish, and in General." *SKASE Journal of Theoretical Linguistics*, vol. 10, no. 1, 2013. [Page 41.]

[34] "gf-ud." [Online]. Available: https://github.com/GrammaticalFramework/gf-ud [Page 53.]

Appendix A

Results

A.1 Code

The code can be found here.

A.2 Table

The complete result of the text generation is shown below.

> Stockholm är en tätort i Stockholms län samt centralort i Stockholms kommun, residensstad i Stockholms län och stiftsstad för Stockholms stift. Stockholm är också en universitetsstad med Kungliga Tekniska högskolan grundat 1827, Karolinska Institutet grundat 1810 och Stockholms universitet grundat 1878. Stockholm är Sveriges största tätort med mer än 1600000 invånare.
>
> Stockholm is a urban area in Stockholm County as well as the seat of Stockholm Municipality, the capital of Stockholm County and the episcopal see of the Diocese of Stockholm. Stockholm is also a university city with Kungliga Tekniska högskolan founded in 1827, Karolinska Institutet founded in 1810 and Stockholms universitet founded in 1878. Stockholm is Sweden's largest urban area with more than 1,600,000 inhabitants.
>
> Stockholm est une zone urbaine dans le comté de Stockholm

et la siège dans la commune de Stockholm, le chef-lieu dans le comté de Stockholm et la siège l'évêché de Stockholm. Stockholm est aussi une ville universitaire avec Kungliga Tekniska högskolan fondé en 1827, Karolinska Institutet fondé en 1810 et Stockholms universitet fondé en 1878. Stockholm est la plus grande zone urbaine de Suède avec plus que 1600000 habitants.

Göteborg är en tätort i Västra Götalands län samt centralort i Göteborgs kommun, residensstad i Västra Götalands län och stiftsstad för Göteborgs stift. Göteborg är också en universitetsstad med Göteborgs universitet grundat 1954 och Chalmers tekniska högskola grundat 1829. Göteborg är Sveriges andra största tätort med mer än 605000 invånare.

Göteborg is a urban area in Västra Götaland County as well as the seat of Göteborg Municipality, the capital of Västra Götaland County and the episcopal see of the Diocese of Göteborg. Göteborg is also a university city with Göteborgs universitet founded in 1954 and Chalmers tekniska högskola founded in 1829. Göteborg is Sweden's second largest urban area with more than 605,000 inhabitants.

Göteborg est une zone urbaine dans le comté de Västra Götaland et la siège dans la commune de Göteborg, le chef-lieu dans le comté de Västra Götaland et la siège l'évêché de Göteborg. Göteborg est aussi une ville universitaire avec Göteborgs universitet fondé en 1954 et Chalmers tekniska högskola fondé en 1829. Göteborg est la deuxième zone urbaine plus grande de Suède avec plus que 605000 habitants.

Malmö är en tätort i Skåne län samt centralort i Malmö kommun och residensstad i Skåne län. Malmö är också en universitetsstad med Malmö universitet grundat 1998. Malmö är Sveriges tredje största tätort med mer än 325000 invånare.

Malmö is a urban area in Skåne County as well as the seat of Malmö Municipality and the capital of Skåne County. Malmö is also a university city with Malmö universitet founded in 1998. Malmö is Sweden's third largest urban area with more than 325,000 inhabitants.

Malmö est une zone urbaine dans le comté de Skåne et la siège dans la commune de Malmö et le chef-lieu dans le comté de Skåne. Malmö est aussi une ville universitaire avec Malmö universitet fondé en 1998. Malmö est la troisième zone urbaine plus grande de Suède avec plus que 325000 habitants.

Uppsala är en tätort i Uppsala län samt centralort i Uppsala kommun, residensstad i Uppsala län och stiftsstad för Uppsala stift. Uppsala är också en universitetsstad med Uppsala universitet grundat 1477 och Sveriges lantbruksuniversitet grundat 1977. Uppsala är Sveriges fjärde största tätort med mer än 165000 invånare.

Uppsala is a urban area in Uppsala County as well as the seat of Uppsala Municipality, the capital of Uppsala County and the episcopal see of the Diocese of Uppsala. Uppsala is also a university city with Uppsala universitet founded in 1477 and Sveriges lantbruksuniversitet founded in 1977. Uppsala is Sweden's fourth largest urban area with more than 165,000 inhabitants.

Uppsala est une zone urbaine dans le comté d'Uppsala et la siège dans la commune d'Uppsala, le chef-lieu dans le comté d'Uppsala et la siège l'évêché d'Uppsala. Uppsala est aussi une ville universitaire avec Uppsala universitet fondé en 1477 et Sveriges lantbruksuniversitet fondé en 1977. Uppsala est la quatrième zone urbaine plus grande de Suède avec plus que 165000 habitants.

Upplands Väsby och Sollentuna är en tätort i Stockholms län. Upplands Väsby och Sollentuna är Sveriges femte största tätort med mer än 145000 invånare.

Upplands Väsby och Sollentuna is a urban area in Stockholm County. Upplands Väsby och Sollentuna is Sweden's fifth largest urban area with more than 145,000 inhabitants.

Upplands Väsby och Sollentuna est une zone urbaine dans le comté de Stockholm. Upplands Väsby och Sollentuna est la cinquième zone urbaine plus grande de Suède avec plus que 145000 habitants.

Västerås är en tätort i Västmanlands län samt centralort i Västerås kommun, residensstad i Västmanlands län och stiftsstad för Västerås stift. Västerås är Sveriges sjätte största tätort med mer än 125000 invånare.

Västerås is a urban area in Västmanland County as well as the seat of Västerås Municipality, the capital of Västmanland County and the episcopal see of the Diocese of Västerås. Västerås is Sweden's sixth largest urban area with more than 125,000 inhabitants.

Västerås est une zone urbaine dans le comté de Västmanland et la siège dans la commune de Västerås, le chef-lieu dans le comté de Västmanland et la siège l'évêché de Västerås. Västerås est la sixième zone urbaine plus grande de Suède avec plus que 125000 habitants.

Örebro är en tätort i Örebro län samt centralort i Örebro kommun och residensstad i Örebro län. Örebro är också en universitetsstad med Örebro universitet grundat 1977. Örebro är Sveriges sjunde största tätort med mer än 125000 invånare.

Örebro is a urban area in Örebro County as well as the seat of Örebro Municipality and the capital of Örebro County. Örebro is also a university city with Örebro universitet founded in 1977. Örebro is Sweden's seventh largest urban area with more than 125,000 inhabitants.

Örebro est une zone urbaine dans le comté de Örebro et la siège dans la commune de Örebro et le chef-lieu dans le comté de Örebro. Örebro est aussi une ville universitaire avec Örebro universitet fondé en 1977. Örebro est la septième zone urbaine plus grande de Suède avec plus que 125000 habitants.

Linköping är en tätort i Östergötlands län samt centralort i Linköpings kommun, residensstad i Östergötlands län och stiftsstad för Linköpings stift. Linköping är också en universitetsstad med Linköpings universitet grundat 1975. Linköping är Sveriges åttonde största tätort med mer än 115000 invånare.

Linköping is a urban area in Östergötland County as

well as the seat of Linköping Municipality, the capital of Östergötland County and the episcopal see of the Diocese of Linköping. Linköping is also a university city with Linköpings universitet founded in 1975. Linköping is Sweden's eighth largest urban area with more than 115,000 inhabitants.

Linköping est une zone urbaine dans le comté de Östergötland et la siège dans la commune de Linköping, le chef-lieu dans le comté de Östergötland et la siège l'évêché de Linköping. Linköping est aussi une ville universitaire avec Linköpings universitet fondé en 1975. Linköping est l'huitième zone urbaine plus grande de Suède avec plus que 115000 habitants.

Helsingborg är en tätort i Skåne län samt centralort i Helsingborgs kommun. Helsingborg är Sveriges nionde största tätort med mer än 110000 invånare.

Helsingborg is a urban area in Skåne County as well as the seat of Helsingborg Municipality. Helsingborg is Sweden's ninth largest urban area with more than 110,000 inhabitants.

Helsingborg est une zone urbaine dans le comté de Skåne et la siège dans la commune d'Helsingborg. Helsingborg est la neufième zone urbaine plus grande de Suède avec plus que 110000 habitants.

Jönköping är en tätort i Jönköpings län samt centralort i Jönköpings kommun och residensstad i Jönköpings län. Jönköping är Sveriges tionde största tätort med mer än 100000 invånare.

Jönköping is a urban area in Jönköping County as well as the seat of Jönköping Municipality and the capital of Jönköping County. Jönköping is Sweden's tenth largest urban area with more than 100,000 inhabitants.

Jönköping est une zone urbaine dans le comté de Jönköping et la siège dans la commune de Jönköping et le chef-lieu dans le comté de Jönköping. Jönköping est la dixième zone urbaine plus grande de Suède avec plus que 100000 habitants.

Norrköping är en tätort i Östergötlands län samt centralort i Norrköpings kommun. det finns mer än 98000 invånare i

Norrköping.

Norrköping is a urban area in Östergötland County as well as the seat of Norrköping Municipality. there are more than 98,000 inhabitants in Norrköping.

Norrköping est une zone urbaine dans le comté de Östergötland et la siège dans la commune de Norrköping. il y a plus que 98000 habitants dans Norrköping.

Umeå är en tätort i Västerbottens län samt centralort i Umeå kommun och residensstad i Västerbottens län. Umeå är också en universitetsstad med Umeå universitet grundat 1965. det finns mer än 91500 invånare i Umeå.

Umeå is a urban area in Västerbotten County as well as the seat of Umeå Municipality and the capital of Västerbotten County. Umeå is also a university city with Umeå universitet founded in 1965. there are more than 91,500 inhabitants in Umeå.

Umeå est une zone urbaine dans le comté de Västerbotten et la siège dans la commune d'Umeå et le chef-lieu dans le comté de Västerbotten. Umeå est aussi une ville universitaire avec Umeå universitet fondé en 1965. il y a plus que 91500 habitants dans Umeå.

Gävle är en tätort i Gävleborgs län samt centralort i Gävle kommun och residensstad i Gävleborgs län. det finns mer än 79000 invånare i Gävle.

Gävle is a urban area in Gävleborg County as well as the seat of Gävle Municipality and the capital of Gävleborg County. there are more than 79,000 inhabitants in Gävle.

Gävle est une zone urbaine dans le comté de Gävleborg et la siège dans la commune de Gävle et le chef-lieu dans le comté de Gävleborg. il y a plus que 79000 habitants dans Gävle.

Södertälje är en tätort i Stockholms län samt centralort i Södertälje kommun. det finns mer än 76000 invånare i Södertälje.

Södertälje is a urban area in Stockholm County as well as the

seat of Södertälje Municipality. there are more than 76,000 inhabitants in Södertälje.

Södertälje est une zone urbaine dans le comté de Stockholm et la siège dans la commune de Södertälje. il y a plus que 76000 habitants dans Södertälje.

Borås är en tätort i Västra Götalands län samt centralort i Borås kommun. det finns mer än 74000 invånare i Borås.

Borås is a urban area in Västra Götaland County as well as the seat of Borås Municipality. there are more than 74,000 inhabitants in Borås.

Borås est une zone urbaine dans le comté de Västra Götaland et la siège dans la commune de Borås. il y a plus que 74000 habitants dans Borås.

Halmstad är en tätort i Hallands län samt centralort i Halmstads kommun och residensstad i Hallands län. det finns mer än 71000 invånare i Halmstad.

Halmstad is a urban area in Halland County as well as the seat of Halmstad Municipality and the capital of Halland County. there are more than 71,000 inhabitants in Halmstad.

Halmstad est une zone urbaine dans le comté d'Halland et la siège dans la commune d'Halmstad et le chef-lieu dans le comté d'Halland. il y a plus que 71000 habitants dans Halmstad.

Växjö är en tätort i Kronobergs län samt centralort i Växjö kommun, residensstad i Kronobergs län och stiftsstad för Växjö stift. Växjö är också en universitetsstad med Linnéuniversitetet grundat 2010. det finns mer än 71000 invånare i Växjö.

Växjö is a urban area in Kronoberg County as well as the seat of Växjö Municipality, the capital of Kronoberg County and the episcopal see of the Diocese of Växjö. Växjö is also a university city with Linnéuniversitetet founded in 2010. there are more than 71,000 inhabitants in Växjö.

Växjö est une zone urbaine dans le comté de Kronoberg et la

siège dans la commune de Växjö, le chef-lieu dans le comté de Kronoberg et la siège l'évêché de Växjö. Växjö est aussi une ville universitaire avec Linnéuniversitetet fondé en 2010. il y a plus que 71000 habitants dans Växjö.

Eskilstuna är en tätort i Södermanlands län samt centralort i Eskilstuna kommun. det finns mer än 69500 invånare i Eskilstuna.

Eskilstuna is a urban area in Södermanland County as well as the seat of Eskilstuna Municipality. there are more than 69,500 inhabitants in Eskilstuna.

Eskilstuna est une zone urbaine dans le comté de Södermanland et la siège dans la commune d'Eskilstuna. il y a plus que 69500 habitants dans Eskilstuna.

Karlstad är en tätort i Värmlands län samt centralort i Karlstads kommun, residensstad i Värmlands län och stiftsstad för Karlstads stift. det finns mer än 67000 invånare i Karlstad.

Karlstad is a urban area in Värmland County as well as the seat of Karlstad Municipality, the capital of Värmland County and the episcopal see of the Diocese of Karlstad. there are more than 67,000 inhabitants in Karlstad.

Karlstad est une zone urbaine dans le comté de Värmland et la siège dans la commune de Karlstad, le chef-lieu dans le comté de Värmland et la siège l'évêché de Karlstad. il y a plus que 67000 habitants dans Karlstad.

Täby är en tätort i Stockholms län samt centralort i Täby kommun. det finns mer än 61000 invånare i Täby.

Täby is a urban area in Stockholm County as well as the seat of Täby Municipality. there are more than 61,000 inhabitants in Täby.

Täby est une zone urbaine dans le comté de Stockholm et la siège dans la commune de Täby. il y a plus que 61000 habitants dans Täby.

Sundsvall är en tätort i Västernorrlands län samt centralort

i Sundsvalls kommun. Sundsvall är också en universitetsstad med Mittuniversitetet grundat 1993. det finns mer än 58500 invånare i Sundsvall.

Sundsvall is a urban area in Västernorrland County as well as the seat of Sundsvall Municipality. Sundsvall is also a university city with Mittuniversitetet founded in 1993. there are more than 58,500 inhabitants in Sundsvall.

Sundsvall est une zone urbaine dans le comté de Västernorrland et la siège dans la commune de Sundsvall. Sundsvall est aussi une ville universitaire avec Mittuniversitetet fondé en 1993. il y a plus que 58500 habitants dans Sundsvall.

Östersund är en tätort i Jämtlands län samt centralort i Staare kommun och residensstad i Jämtlands län. Östersund är också en universitetsstad med Mittuniversitetet grundat 1993. det finns mer än 52500 invånare i Östersund.

Östersund is a urban area in Jämtland County as well as the seat of Staare Municipality and the capital of Jämtland County. Östersund is also a university city with Mittuniversitetet founded in 1993. there are more than 52,500 inhabitants in Östersund.

Östersund est une zone urbaine dans le comté de Jämtland et la siège dans la commune de Staare et le chef-lieu dans le comté de Jämtland. Östersund est aussi une ville universitaire avec Mittuniversitetet fondé en 1993. il y a plus que 52500 habitants dans Östersund.

Trollhättan är en tätort i Västra Götalands län samt centralort i Trollhättans kommun. det finns mer än 50500 invånare i Trollhättan.

Trollhättan is a urban area in Västra Götaland County as well as the seat of Trollhättan Municipality. there are more than 50,500 inhabitants in Trollhättan.

Trollhättan est une zone urbaine dans le comté de Västra Götaland et la siège dans la commune de Trollhättan. il y a plus que 50500 habitants dans Trollhättan.

Luleå är en tätort i Norrbottens län samt centralort i Luleå

kommun, residensstad i Norrbottens län och stiftsstad för Luleå stift. Luleå är också en universitetsstad med Luleå tekniska universitet grundat 1971. det finns mer än 49000 invånare i Luleå.

Luleå is a urban area in Norrbotten County as well as the seat of Luleå Municipality, the capital of Norrbotten County and the episcopal see of the Diocese of Luleå. Luleå is also a university city with Luleå tekniska universitet founded in 1971. there are more than 49,000 inhabitants in Luleå.

Luleå est une zone urbaine dans le comté de Norrbotten et la siège dans la commune de Luleå, le chef-lieu dans le comté de Norrbotten et la siège l'évêché de Luleå. Luleå est aussi une ville universitaire avec Luleå tekniska universitet fondé en 1971. il y a plus que 49000 habitants dans Luleå.

Nordöstra Göteborg är en tätort i Västra Götalands län. det finns mer än 47000 invånare i Nordöstra Göteborg.

Nordöstra Göteborg is a urban area in Västra Götaland County. there are more than 47,000 inhabitants in Nordöstra Göteborg.

Nordöstra Göteborg est une zone urbaine dans le comté de Västra Götaland. il y a plus que 47000 habitants dans Nordöstra Göteborg.

Tumba är en tätort i Stockholms län samt centralort i Botkyrka kommun. det finns mer än 46000 invånare i Tumba.

Tumba is a urban area in Stockholm County as well as the seat of Botkyrka Municipality. there are more than 46,000 inhabitants in Tumba.

Tumba est une zone urbaine dans le comté de Stockholm et la siège dans la commune de Botkyrka. il y a plus que 46000 habitants dans Tumba.

Borlänge är en tätort i Dalarna län samt centralort i Borlänge kommun. det finns mer än 44500 invånare i Borlänge.

Borlänge is a urban area in Dalarna County as well as the seat of Borlänge Municipality. there are more than 44,500

inhabitants in Borlänge.

Borlänge est une zone urbaine dans le comté de Dalarna et la siège dans la commune de Borlänge. il y a plus que 44500 habitants dans Borlänge.

Lidingö är en tätort i Stockholms län samt centralort i Lidingö kommun. det finns mer än 44000 invånare i Lidingö.

Lidingö is a urban area in Stockholm County as well as the seat of Lidingö Municipality. there are more than 44,000 inhabitants in Lidingö.

Lidingö est une zone urbaine dans le comté de Stockholm et la siège dans la commune de Lidingö. il y a plus que 44000 habitants dans Lidingö.

Kalmar är en tätort i Kalmar län samt centralort i Kalmar kommun, residensstad i Kalmar län och stiftsstad för Kalmar stift. Kalmar är också en universitetsstad med Linnéuniversitetet grundat 2010. det finns mer än 41500 invånare i Kalmar.

Kalmar is a urban area in Kalmar County as well as the seat of Kalmar Municipality, the capital of Kalmar County and the episcopal see of the Diocese of Kalmar. Kalmar is also a university city with Linnéuniversitetet founded in 2010. there are more than 41,500 inhabitants in Kalmar.

Kalmar est une zone urbaine dans le comté de Kalmar et la siège dans la commune de Kalmar, le chef-lieu dans le comté de Kalmar et la siège l'évêché de Kalmar. Kalmar est aussi une ville universitaire avec Linnéuniversitetet fondé en 2010. il y a plus que 41500 habitants dans Kalmar.

Kristianstad är en tätort i Skåne län samt centralort i Kristianstads kommun. det finns mer än 41000 invånare i Kristianstad.

Kristianstad is a urban area in Skåne County as well as the seat of Kristianstad Municipality. there are more than 41,000 inhabitants in Kristianstad.

Kristianstad est une zone urbaine dans le comté de Skåne et

la siège dans la commune de Kristianstad. il y a plus que 41000 habitants dans Kristianstad.

Skövde är en tätort i Västra Götalands län samt centralort i Skövde kommun. det finns mer än 39500 invånare i Skövde.

Skövde is a urban area in Västra Götaland County as well as the seat of Skövde Municipality. there are more than 39,500 inhabitants in Skövde.

Skövde est une zone urbaine dans le comté de Västra Götaland et la siège dans la commune de Skövde. il y a plus que 39500 habitants dans Skövde.

Falun är en tätort i Dalarna län samt centralort i Faluns kommun och residensstad i Dalarna län. det finns mer än 39000 invånare i Falun.

Falun is a urban area in Dalarna County as well as the seat of Falun Municipality and the capital of Dalarna County. there are more than 39,000 inhabitants in Falun.

Falun est une zone urbaine dans le comté de Dalarna et la siège dans la commune de Falun et le chef-lieu dans le comté de Dalarna. il y a plus que 39000 habitants dans Falun.

Nyköping är en tätort i Södermanlands län samt centralort i Nyköpings kommun och residensstad i Södermanlands län. det finns mer än 38500 invånare i Nyköping.

Nyköping is a urban area in Södermanland County as well as the seat of Nyköping Municipality and the capital of Södermanland County. there are more than 38,500 inhabitants in Nyköping.

Nyköping est une zone urbaine dans le comté de Södermanland et la siège dans la commune de Nyköping et le chef-lieu dans le comté de Södermanland. il y a plus que 38500 habitants dans Nyköping.

Upplands Väsby är en tätort i Stockholms län samt centralort i Upplands Väsby kommun. det finns mer än 37500 invånare i Upplands Väsby.

Upplands Väsby is a urban area in Stockholm County as well

as the seat of Upplands Väsby Municipality. there are more than 37,500 inhabitants in Upplands Väsby.

Upplands Väsby est une zone urbaine dans le comté de Stockholm et la siège dans la commune d'Upplands Väsby. il y a plus que 37500 habitants dans Upplands Väsby.

Sollentuna är en tätort i Stockholms län. det finns mer än 37000 invånare i Sollentuna.

Sollentuna is a urban area in Stockholm County. there are more than 37,000 inhabitants in Sollentuna.

Sollentuna est une zone urbaine dans le comté de Stockholm. il y a plus que 37000 habitants dans Sollentuna.

Karlskrona är en tätort i Blekinge län samt centralort i Karlskrona kommun och residensstad i Blekinge län. det finns mer än 36500 invånare i Karlskrona.

Karlskrona is a urban area in Blekinge County as well as the seat of Karlskrona Municipality and the capital of Blekinge County. there are more than 36,500 inhabitants in Karlskrona.

Karlskrona est une zone urbaine dans le comté de Blekinge et la siège dans la commune de Karlskrona et le chef-lieu dans le comté de Blekinge. il y a plus que 36500 habitants dans Karlskrona.

Skellefteå är en tätort i Västerbottens län samt centralort i Skellefteå kommun. det finns mer än 36000 invånare i Skellefteå.

Skellefteå is a urban area in Västerbotten County as well as the seat of Skellefteå Municipality. there are more than 36,000 inhabitants in Skellefteå.

Skellefteå est une zone urbaine dans le comté de Västerbotten et la siège dans la commune de Skellefteå. il y a plus que 36000 habitants dans Skellefteå.

Varberg är en tätort i Hallands län samt centralort i Varbergs kommun. det finns mer än 36000 invånare i Varberg.

Varberg is a urban area in Halland County as well as the seat of Varberg Municipality. there are more than 36,000 inhabitants in Varberg.

Varberg est une zone urbaine dans le comté d'Halland et la siège dans la commune de Varberg. il y a plus que 36000 habitants dans Varberg.

Uddevalla är en tätort i Västra Götalands län samt centralort i Uddevalla kommun. det finns mer än 35500 invånare i Uddevalla.

Uddevalla is a urban area in Västra Götaland County as well as the seat of Uddevalla Municipality. there are more than 35,500 inhabitants in Uddevalla.

Uddevalla est une zone urbaine dans le comté de Västra Götaland et la siège dans la commune d'Uddevalla. il y a plus que 35500 habitants dans Uddevalla.

Åkersberga är en tätort i Stockholms län samt centralort i Österåkers kommun. det finns mer än 35500 invånare i Åkersberga.

Åkersberga is a urban area in Stockholm County as well as the seat of Österåker Municipality. there are more than 35,500 inhabitants in Åkersberga.

Åkersberga est une zone urbaine dans le comté de Stockholm et la siège dans la commune de Österåker. il y a plus que 35500 habitants dans Åkersberga.

Harkie är en tätort i Västmanlands län. det finns mer än 315 invånare i Harkie.

Harkie is a urban area in Västmanland County. there are more than 315 inhabitants in Harkie.

Harkie est une zone urbaine dans le comté de Västmanland. il y a plus que 315 habitants dans Harkie.

Färila är en tätort i Gävleborgs län. det finns mer än 1300 invånare i Färila.

Färila is a urban area in Gävleborg County. there are more

than 1,300 inhabitants in Färila.

Färila est une zone urbaine dans le comté de Gävleborg. il y a plus que 1300 habitants dans Färila.

Gislaved är en tätort i Jönköpings län samt centralort i Gislaveds kommun. det finns mer än 10000 invånare i Gislaved.

Gislaved is a urban area in Jönköping County as well as the seat of Gislaved Municipality. there are more than 10,000 inhabitants in Gislaved.

Gislaved est une zone urbaine dans le comté de Jönköping et la siège dans la commune de Gislaved. il y a plus que 10000 habitants dans Gislaved.

Berthåga är en tätort i Uppsala län. det finns mer än 260 invånare i Berthåga.

Berthåga is a urban area in Uppsala County. there are more than 260 inhabitants in Berthåga.

Berthåga est une zone urbaine dans le comté d'Uppsala. il y a plus que 260 habitants dans Berthåga.

Grycksbo är en tätort i Dalarna län. det finns mer än 1850 invånare i Grycksbo.

Grycksbo is a urban area in Dalarna County. there are more than 1,850 inhabitants in Grycksbo.

Grycksbo est une zone urbaine dans le comté de Dalarna. il y a plus que 1850 habitants dans Grycksbo.

Forsvik är en tätort i Västra Götalands län. det finns mer än 320 invånare i Forsvik.

Forsvik is a urban area in Västra Götaland County. there are more than 320 inhabitants in Forsvik.

Forsvik est une zone urbaine dans le comté de Västra Götaland. il y a plus que 320 habitants dans Forsvik.

Bergströmshusen är en tätort i Skåne län. det finns mer än 400 invånare i Bergströmshusen.

Bergströmshusen is a urban area in Skåne County. there are more than 400 inhabitants in Bergströmshusen.

Bergströmshusen est une zone urbaine dans le comté de Skåne. il y a plus que 400 habitants dans Bergströmshusen.

Bergkarlås är en tätort i Dalarna län. det finns mer än 170 invånare i Bergkarlås.

Bergkarlås is a urban area in Dalarna County. there are more than 170 inhabitants in Bergkarlås.

Bergkarlås est une zone urbaine dans le comté de Dalarna. il y a plus que 170 habitants dans Bergkarlås.

Oxnö och Svärdsö är en tätort i Stockholms län. det finns mer än 335 invånare i Oxnö och Svärdsö.

Oxnö och Svärdsö is a urban area in Stockholm County. there are more than 335 inhabitants in Oxnö och Svärdsö.

Oxnö och Svärdsö est une zone urbaine dans le comté de Stockholm. il y a plus que 335 habitants dans Oxnö och Svärdsö.

Lugnvik är en tätort i Västernorrlands län. det finns mer än 220 invånare i Lugnvik.

Lugnvik is a urban area in Västernorrland County. there are more than 220 inhabitants in Lugnvik.

Lugnvik est une zone urbaine dans le comté de Västernorrland. il y a plus que 220 habitants dans Lugnvik.

Mölnlycke är en tätort i Västra Götalands län samt centralort i Härryda kommun. det finns mer än 18000 invånare i Mölnlycke.

Mölnlycke is a urban area in Västra Götaland County as well as the seat of Härryda Municipality. there are more than 18,000 inhabitants in Mölnlycke.

Mölnlycke est une zone urbaine dans le comté de Västra Götaland et la siège dans la commune d'Härryda. il y a plus que 18000 habitants dans Mölnlycke.

Ljungs Berg är en tätort i Västra Götalands län. det finns mer än 205 invånare i Ljungs Berg.

Ljungs Berg is a urban area in Västra Götaland County. there are more than 205 inhabitants in Ljungs Berg.

Ljungs Berg est une zone urbaine dans le comté de Västra Götaland. il y a plus que 205 habitants dans Ljungs Berg.

Sydöstra Fullerö är en tätort i Uppsala län. det finns mer än 930 invånare i Sydöstra Fullerö.

Sydöstra Fullerö is a urban area in Uppsala County. there are more than 930 inhabitants in Sydöstra Fullerö.

Sydöstra Fullerö est une zone urbaine dans le comté d'Uppsala. il y a plus que 930 habitants dans Sydöstra Fullerö.

Näs bruk är en tätort i Dalarna län. det finns mer än 190 invånare i Näs bruk.

Näs bruk is a urban area in Dalarna County. there are more than 190 inhabitants in Näs bruk.

Näs bruk est une zone urbaine dans le comté de Dalarna. il y a plus que 190 habitants dans Näs bruk.

Rutvik är en tätort i Norrbottens län. det finns mer än 870 invånare i Rutvik.

Rutvik is a urban area in Norrbotten County. there are more than 870 inhabitants in Rutvik.

Rutvik est une zone urbaine dans le comté de Norrbotten. il y a plus que 870 habitants dans Rutvik.

Nordvära och Källstorp är en tätort i Hallands län. det finns mer än 265 invånare i Nordvära och Källstorp.

Nordvära och Källstorp is a urban area in Halland County. there are more than 265 inhabitants in Nordvära och Källstorp.

Nordvära och Källstorp est une zone urbaine dans le comté d'Halland. il y a plus que 265 habitants dans Nordvära och

Källstorp.

Gubbo är en tätort i Dalarna län. det finns mer än 200 invånare i Gubbo.

Gubbo is a urban area in Dalarna County. there are more than 200 inhabitants in Gubbo.

Gubbo est une zone urbaine dans le comté de Dalarna. il y a plus que 200 habitants dans Gubbo.

Abbekås är en tätort i Skåne län. det finns mer än 850 invånare i Abbekås.

Abbekås is a urban area in Skåne County. there are more than 850 inhabitants in Abbekås.

Abbekås est une zone urbaine dans le comté de Skåne. il y a plus que 850 habitants dans Abbekås.

Bäckseda är en tätort i Jönköpings län. det finns mer än 400 invånare i Bäckseda.

Bäckseda is a urban area in Jönköping County. there are more than 400 inhabitants in Bäckseda.

Bäckseda est une zone urbaine dans le comté de Jönköping. il y a plus que 400 habitants dans Bäckseda.

Ytterån är en tätort i Jämtlands län. det finns mer än 200 invånare i Ytterån.

Ytterån is a urban area in Jämtland County. there are more than 200 inhabitants in Ytterån.

Ytterån est une zone urbaine dans le comté de Jämtland. il y a plus que 200 habitants dans Ytterån.

Mockfjärd är en tätort i Dalarna län. det finns mer än 2000 invånare i Mockfjärd.

Mockfjärd is a urban area in Dalarna County. there are more than 2,000 inhabitants in Mockfjärd.

Mockfjärd est une zone urbaine dans le comté de Dalarna. il y a plus que 2000 habitants dans Mockfjärd.

Vittskövle är en tätort i Skåne län. det finns mer än 235 invånare i Vittskövle.

Vittskövle is a urban area in Skåne County. there are more than 235 inhabitants in Vittskövle.

Vittskövle est une zone urbaine dans le comté de Skåne. il y a plus que 235 habitants dans Vittskövle.

Sjöboviken är en tätort i Örebro län. det finns mer än 200 invånare i Sjöboviken.

Sjöboviken is a urban area in Örebro County. there are more than 200 inhabitants in Sjöboviken.

Sjöboviken est une zone urbaine dans le comté de Örebro. il y a plus que 200 habitants dans Sjöboviken.

Grebbestad är en tätort i Västra Götalands län. det finns mer än 1900 invånare i Grebbestad.

Grebbestad is a urban area in Västra Götaland County. there are more than 1,900 inhabitants in Grebbestad.

Grebbestad est une zone urbaine dans le comté de Västra Götaland. il y a plus que 1900 habitants dans Grebbestad.

Bredviken är en tätort i Norrbottens län. det finns mer än 370 invånare i Bredviken.

Bredviken is a urban area in Norrbotten County. there are more than 370 inhabitants in Bredviken.

Bredviken est une zone urbaine dans le comté de Norrbotten. il y a plus que 370 habitants dans Bredviken.

Ulvkälla är en tätort i Jämtlands län. det finns mer än 390 invånare i Ulvkälla.

Ulvkälla is a urban area in Jämtland County. there are more than 390 inhabitants in Ulvkälla.

Ulvkälla est une zone urbaine dans le comté de Jämtland. il y a plus que 390 habitants dans Ulvkälla.

Röda holme är en tätort i Hallands län. det finns mer än 670 invånare i Röda holme.

Röda holme is a urban area in Halland County. there are more than 670 inhabitants in Röda holme.

Röda holme est une zone urbaine dans le comté d'Halland. il y a plus que 670 habitants dans Röda holme.

Lenhovda är en tätort i Kronobergs län. det finns mer än 1750 invånare i Lenhovda.

Lenhovda is a urban area in Kronoberg County. there are more than 1,750 inhabitants in Lenhovda.

Lenhovda est une zone urbaine dans le comté de Kronoberg. il y a plus que 1750 habitants dans Lenhovda.

Djurås är en tätort i Dalarna län samt centralort i Gagnefs kommun. det finns mer än 2150 invånare i Djurås.

Djurås is a urban area in Dalarna County as well as the seat of Gagnef Municipality. there are more than 2,150 inhabitants in Djurås.

Djurås est une zone urbaine dans le comté de Dalarna et la siège dans la commune de Gagnef. il y a plus que 2150 habitants dans Djurås.

Fullerö hage är en tätort i Uppsala län. det finns mer än 0 invånare i Fullerö hage.

Fullerö hage is a urban area in Uppsala County. there are more than 0 inhabitants in Fullerö hage.

Fullerö hage est une zone urbaine dans le comté d'Uppsala. il y a plus que 0 habitants dans Fullerö hage.

Bäckebo är en tätort i Kalmar län. det finns mer än 225 invånare i Bäckebo.

Bäckebo is a urban area in Kalmar County. there are more than 225 inhabitants in Bäckebo.

Bäckebo est une zone urbaine dans le comté de Kalmar. il y a plus que 225 habitants dans Bäckebo.

Mysterna är en tätort i Västra Götalands län. det finns mer än 3400 invånare i Mysterna.

Mysterna is a urban area in Västra Götaland County. there are more than 3,400 inhabitants in Mysterna.

Mysterna est une zone urbaine dans le comté de Västra Götaland. il y a plus que 3400 habitants dans Mysterna.

Rolfstorp är en tätort i Hallands län. det finns mer än 585 invånare i Rolfstorp.

Rolfstorp is a urban area in Halland County. there are more than 585 inhabitants in Rolfstorp.

Rolfstorp est une zone urbaine dans le comté d'Halland. il y a plus que 585 habitants dans Rolfstorp.

Skebokvarn är en tätort i Södermanlands län. det finns mer än 200 invånare i Skebokvarn.

Skebokvarn is a urban area in Södermanland County. there are more than 200 inhabitants in Skebokvarn.

Skebokvarn est une zone urbaine dans le comté de Södermanland. il y a plus que 200 habitants dans Skebokvarn.

Glömminge är en tätort i Kalmar län. det finns mer än 885 invånare i Glömminge.

Glömminge is a urban area in Kalmar County. there are more than 885 inhabitants in Glömminge.

Glömminge est une zone urbaine dans le comté de Kalmar. il y a plus que 885 habitants dans Glömminge.

Mullhyttan är en tätort i Örebro län. det finns mer än 530 invånare i Mullhyttan.

Mullhyttan is a urban area in Örebro County. there are more than 530 inhabitants in Mullhyttan.

Mullhyttan est une zone urbaine dans le comté de Örebro. il y a plus que 530 habitants dans Mullhyttan.

Bunkeflostrand är en tätort i Skåne län. det finns mer än 15000 invånare i Bunkeflostrand.

Bunkeflostrand is a urban area in Skåne County. there are more than 15,000 inhabitants in Bunkeflostrand.

Bunkeflostrand est une zone urbaine dans le comté de Skåne. il y a plus que 15000 habitants dans Bunkeflostrand.

Högsjö är en tätort i Södermanlands län. det finns mer än 650 invånare i Högsjö.

Högsjö is a urban area in Södermanland County. there are more than 650 inhabitants in Högsjö.

Högsjö est une zone urbaine dans le comté de Södermanland. il y a plus que 650 habitants dans Högsjö.

Åmsele är en tätort i Västerbottens län. det finns mer än 180 invånare i Åmsele.

Åmsele is a urban area in Västerbotten County. there are more than 180 inhabitants in Åmsele.

Åmsele est une zone urbaine dans le comté de Västerbotten. il y a plus que 180 habitants dans Åmsele.

Enstaberga är en tätort i Södermanlands län. det finns mer än 440 invånare i Enstaberga.

Enstaberga is a urban area in Södermanland County. there are more than 440 inhabitants in Enstaberga.

Enstaberga est une zone urbaine dans le comté de Södermanland. il y a plus que 440 habitants dans Enstaberga.

Glanshammar är en tätort i Örebro län. det finns mer än 790 invånare i Glanshammar.

Glanshammar is a urban area in Örebro County. there are more than 790 inhabitants in Glanshammar.

Glanshammar est une zone urbaine dans le comté de Örebro. il y a plus que 790 habitants dans Glanshammar.

Frövi är en tätort i Örebro län. det finns mer än 2550 invånare i Frövi.

Frövi is a urban area in Örebro County. there are more than 2,550 inhabitants in Frövi.

Frövi est une zone urbaine dans le comté de Örebro. il y a plus que 2550 habitants dans Frövi.

Tyringe är en tätort i Skåne län. det finns mer än 4800 invånare i Tyringe.

Tyringe is a urban area in Skåne County. there are more than 4,800 inhabitants in Tyringe.

Tyringe est une zone urbaine dans le comté de Skåne. il y a plus que 4800 habitants dans Tyringe.

Vallkärra är en tätort i Skåne län. det finns mer än 420 invånare i Vallkärra.

Vallkärra is a urban area in Skåne County. there are more than 420 inhabitants in Vallkärra.

Vallkärra est une zone urbaine dans le comté de Skåne. il y a plus que 420 habitants dans Vallkärra.

Fäggeby är en tätort i Dalarna län. det finns mer än 80 invånare i Fäggeby.

Fäggeby is a urban area in Dalarna County. there are more than 80 inhabitants in Fäggeby.

Fäggeby est une zone urbaine dans le comté de Dalarna. il y a plus que 80 habitants dans Fäggeby.

Stavreviken är en tätort i Västernorrlands län. det finns mer än 205 invånare i Stavreviken.

Stavreviken is a urban area in Västernorrland County. there are more than 205 inhabitants in Stavreviken.

Stavreviken est une zone urbaine dans le comté de Västernorrland. il y a plus que 205 habitants dans Stavreviken.

Hallen är en tätort i Jämtlands län. det finns mer än 230 invånare i Hallen.

Hallen is a urban area in Jämtland County. there are more than 230 inhabitants in Hallen.

Hallen est une zone urbaine dans le comté de Jämtland. il y a plus que 230 habitants dans Hallen.

Långared är en tätort i Västra Götalands län. det finns mer än 280 invånare i Långared.

Långared is a urban area in Västra Götaland County. there are more than 280 inhabitants in Långared.

Långared est une zone urbaine dans le comté de Västra Götaland. il y a plus que 280 habitants dans Långared.

Björkvik är en tätort i Södermanlands län. det finns mer än 435 invånare i Björkvik.

Björkvik is a urban area in Södermanland County. there are more than 435 inhabitants in Björkvik.

Björkvik est une zone urbaine dans le comté de Södermanland. il y a plus que 435 habitants dans Björkvik.

Nöbbele är en tätort i Kronobergs län. det finns mer än 245 invånare i Nöbbele.

Nöbbele is a urban area in Kronoberg County. there are more than 245 inhabitants in Nöbbele.

Nöbbele est une zone urbaine dans le comté de Kronoberg. il y a plus que 245 habitants dans Nöbbele.

Mellbystrand är en tätort i Hallands län. det finns mer än 4150 invånare i Mellbystrand.

Mellbystrand is a urban area in Halland County. there are more than 4,150 inhabitants in Mellbystrand.

Mellbystrand est une zone urbaine dans le comté d'Halland. il y a plus que 4150 habitants dans Mellbystrand.

Tranvik är en tätort i Stockholms län. det finns mer än 235 invånare i Tranvik.

Tranvik is a urban area in Stockholm County. there are more than 235 inhabitants in Tranvik.

Tranvik est une zone urbaine dans le comté de Stockholm. il y a plus que 235 habitants dans Tranvik.

Ödsmål är en tätort i Västra Götalands län. det finns mer än 1450 invånare i Ödsmål.

Ödsmål is a urban area in Västra Götaland County. there are more than 1,450 inhabitants in Ödsmål.

Ödsmål est une zone urbaine dans le comté de Västra Götaland. il y a plus que 1450 habitants dans Ödsmål.

Moliden är en tätort i Västernorrlands län. det finns mer än 285 invånare i Moliden.

Moliden is a urban area in Västernorrland County. there are more than 285 inhabitants in Moliden.

Moliden est une zone urbaine dans le comté de Västernorrland. il y a plus que 285 habitants dans Moliden.

Söderåkra är en tätort i Kalmar län. det finns mer än 930 invånare i Söderåkra.

Söderåkra is a urban area in Kalmar County. there are more than 930 inhabitants in Söderåkra.

Söderåkra est une zone urbaine dans le comté de Kalmar. il y a plus que 930 habitants dans Söderåkra.

Sidus är en tätort i Östergötlands län. det finns mer än 200 invånare i Sidus.

Sidus is a urban area in Östergötland County. there are more than 200 inhabitants in Sidus.

Sidus est une zone urbaine dans le comté de Östergötland. il y a plus que 200 habitants dans Sidus.

Klingsta och Allsta är en tätort i Västernorrlands län. det finns mer än 565 invånare i Klingsta och Allsta.

Klingsta och Allsta is a urban area in Västernorrland County. there are more than 565 inhabitants in Klingsta och Allsta.

Klingsta och Allsta est une zone urbaine dans le comté de Västernorrland. il y a plus que 565 habitants dans Klingsta

och Allsta.

Kareby och Ringby är en tätort i Västra Götalands län. det finns mer än 235 invånare i Kareby och Ringby.

Kareby och Ringby is a urban area in Västra Götaland County. there are more than 235 inhabitants in Kareby och Ringby.

Kareby och Ringby est une zone urbaine dans le comté de Västra Götaland. il y a plus que 235 habitants dans Kareby och Ringby.

Godegård är en tätort i Östergötlands län. det finns mer än 195 invånare i Godegård.

Godegård is a urban area in Östergötland County. there are more than 195 inhabitants in Godegård.

Godegård est une zone urbaine dans le comté de Östergötland. il y a plus que 195 habitants dans Godegård.

Berg är en tätort i Östergötlands län. det finns mer än 300 invånare i Berg.

Berg is a urban area in Östergötland County. there are more than 300 inhabitants in Berg.

Berg est une zone urbaine dans le comté de Östergötland. il y a plus que 300 habitants dans Berg.

Appendix B

A Way of Building GF Numbers in Python

B.1 Method

GF has a set of pre-defined functions that return a `Dig` type. These functions range from `n0_Dig` to `n9_Dig`. Using the function `mkDigits : Dig -> Digits` a `Dig` can be turned into a `Digits` type. Furthermore, the function `mkDigits : Dig -> Digits -> Digits` takes a `Dig` and a `Digits` and returns a `Digits`. According to the example provided in the RGL documentation,

```
mkDigits n1_Dig (mkDigits n2_Dig)

=> 12

mkDigits n1_Dig
    (mkDigits n2_Dig (mkDigits n3_Dig
        (mkDigits n3_Dig (mkDigits n4_Dig
            (mkDigits n8_Dig (mkDigits n6_Dig))))))

=> 1,233,486
```

Thus, there is a way to build a number with the `Digits` type by recursively calling on `mkDigits` with the appropriate arguments.

Using these functions the string representation of the number can be converted into a `Digits` in Python. The first step is to define a set of

functions in the abstract syntax and all concrete syntaxes:

Abstact syntax:

```
mkDigit0 : Digits ;
mkDigit1 : Digits ;
mkDigit2 : Digits ;
mkDigit3 : Digits ;
mkDigit4 : Digits ;
mkDigit5 : Digits ;
mkDigit6 : Digits ;
mkDigit7 : Digits ;
mkDigit8 : Digits ;
mkDigit9 : Digits ;

mkDigits0 : Digits -> Digits ;
mkDigits1 : Digits -> Digits ;
mkDigits2 : Digits -> Digits ;
mkDigits3: Digits -> Digits ;
mkDigits4: Digits -> Digits ;
mkDigits5: Digits -> Digits ;
mkDigits6: Digits -> Digits ;
mkDigits7: Digits -> Digits ;
mkDigits8: Digits -> Digits ;
mkDigits9: Digits -> Digits ;
```

Concrete Syntax:

```
mkDigit0 = mkDigits S.n0_Dig ;
mkDigit1 = mkDigits S.n1_Dig ;
mkDigit2 = mkDigits S.n2_Dig ;
mkDigit3 = mkDigits S.n3_Dig ;
mkDigit4 = mkDigits S.n4_Dig ;
mkDigit5 = mkDigits S.n5_Dig ;
mkDigit6 = mkDigits S.n6_Dig ;
mkDigit7 = mkDigits S.n7_Dig ;
mkDigit8 = mkDigits S.n8_Dig ;
mkDigit9 = mkDigits S.n9_Dig ;

mkDigits0 digits = mkDigits S.n0_Dig digits ;
mkDigits1 digits = mkDigits S.n1_Dig digits ;
```

```
mkDigits2 digits = mkDigits S.n2_Dig digits ;
mkDigits3 digits = mkDigits S.n3_Dig digits ;
mkDigits4 digits = mkDigits S.n4_Dig digits ;
mkDigits5 digits = mkDigits S.n5_Dig digits ;
mkDigits6 digits = mkDigits S.n6_Dig digits ;
mkDigits7 digits = mkDigits S.n7_Dig digits ;
mkDigits8 digits = mkDigits S.n8_Dig digits ;
mkDigits9 digits = mkDigits S.n9_Dig digits ;
```

Then a method needs to be defined in Python which iterates through each index of the string, and calls on the appropriate GF function on that character:

```
def toDigits(population):
population_str = str(population)
l = len(population_str)
c = population_str[-1]
digits_gf = pgf.Expr('mkDigit{}'.format(c),[])
for i in range(2,l+1, 1):
    c = population_str[-i]
    digits_gf = pgf.Expr('mkDigits{}'.format(c),
                         [digits_gf])

return digits_gf
```

The python method uses the `format` method available for a string to choose which function to call based on what character (number) on the string is currently indexed. This is then iterated until each character in the number string has been visited. The tree is built iteratively and then used as an input argument to the function that needs that specific information.

TRITA-EECS-EX-2023:54

€€€€ For DIVA €€€€

{
"Author1": { "Last name": "Matinzadeh",
"First name": "Keivan",
"Local User Id": "u100001",
"E-mail": "keivanm@kth.se",
"organisation": {"L1": "School of Electrical Engineering and Computer Science",
}
},
"Cycle": "2",
"Course code": "DA231X",
"Credits": "30.0",
"Degree1": {"Educational program": "Degree Programme in Computer Science and Engineering"
,"programcode": "CDATE"
,"Degree": "Degree of Master of Science in Engineering"
,"subjectArea": "Computer Science and Engineering"
},
"Title": {
"Main title": "Generating Wikipedia Articles with Grammatical Framework",
"Subtitle": "A Case Study",
"Language": "eng" },
"Alternative title": {
"Main title": "Generering av Wikipedia-artiklar med Grammatical Framework",
"Subtitle": "En fallstudie",
"Language": "swe"
},
"Supervisor1": { "Last name": "Boye",
"First name": "Johan",
"Local User Id": "u100003",
"E-mail": "jboye@kth.se",
"organisation": {"L1": "School of Electrical Engineering and Computer Science",
"L2": "Computer Science" }
},
"Supervisor2": { "Last name": "Ranta",
"First name": "Aarne",
"E-mail": "sb@kth.se",
"Other organisation": "Chalmers Institute of Technology, Department of Computer Science and Engineering"
},
"Examiner1": { "Last name": "Kann",
"First name": "Viggo",
"Local User Id": "u1d13i2c",
"E-mail": "viggo@kth.se",
"organisation": {"L1": "School of Electrical Engineering and Computer Science",
"L2": "Computer Science" }
},
"National Subject Categories": "10201, 10206",
"Other information": {"Year": "2023", "Number of pages": "1,??"},
"Series": { "Title of series": "TRITA-EECS-EX" , "No. in series": "2022:00" },
"Opponents": { "Name": "A. B. Normal & A. X. E. Normalè"},
"Presentation": { "Date": "2022-03-15 13:00"

,"Language":"eng"

,"Room": "via Zoom https://kth-se.zoom.us/j/ddddddddddd"

,"Address": "Isafjordsgatan 22 (Kistagången 16)"

,"City": "Stockholm" },

"Number of lang instances": "2",
"Abstract[eng]": €€€€

Natural language generation is a method used to produce understandable texts in human languages from data \cite{gatt_survey_2018}. Grammatical Framework is a grammar formalism and a functional programming language using a non-statistical approach to build natural language applications. It separates the semantics and the syntax - achieving multilingualism by mapping the same semantic model to several syntaxes \cite{ranta-2011}. Grammatical Framework also has a large library called the Resource Grammar Library which serves the programmer pre-made functions in over 30 languages ready to be used to build words and sentences \cite{gfresourcegrammarsynopsis}.

This report investigates if Grammatical Framework can be successfully used to perform natural language generation in order to create Wikipedia articles from data taken from Wikidata.

A grammar and a program has been built to generate articles in Swedish for urban areas in Sweden. The grammar has been built around the structure of the first three sentences in the Swedish article about the urban area Linköping. Furthermore, the grammar and program is extended in order to support generation of the same articles in English and French.

The results show that Grammatical Framework can be somewhat successfully used to generate small Wikipedia articles in different languages using data from Wikidata as input. While all texts were coherent, the Swedish texts were the ones having the least amount of grammatical mistakes. The biggest drawback is the rule of no pattern matching on run-time arguments, which severely limits the programmer since many functions in the resource grammar library use pattern matching internally.

Even though Grammatical Framework does not solve the whole problem, it serves as a powerful enough tool to be suitable for natural language generation, with the main advantage being that it relieves the programmer from needing to pay attention to tasks related to grammar such as inflection and gender agreement.

€€€€,
"Keywords[eng]": €€€€
Grammatical Framework, Computational Linguistics, Natural Language Generation, Computer Science €€€€,
"Abstract[swe]": €€€€
Textgeneration är en metod som används för att generera naturlig text från data. Grammatical Framework är en grammatikformalism och ett funktionellt programmeringsspråk som använder ett icke-statistiskt tillvägagångssätt för att skapa språkteknologiska applikationer. Grammatical Framework separerar semantik och syntax, och uppnår flerspråkighet genom att länka samma semantiska model till flera syntaxer. Grammatical Framework har också ett stort bibliotek, en resursgrammatik, kallad Resource Grammar Library, som tillhandahåller applikations-programmeraren färdiga funktioner i över 30 språk redo att användas för att skapa ord och meningar.

Syftet med den här rapporten är att undersöka om Grammatical Framework på ett framgångsrikt sätt kan användas för att generera Wikipedia-artiklar genom att använda data taget från Wikidata.

En grammatik och ett program har skapats för att generera artiklar på svenska för svenska tätorter. Grammatiken använder de tre första meningarna i den svenska artikeln om tätorten Linköping som text-struktur. Vidare utökas grammatiken och programmet till att kunna generera samma artiklar på engelska och franska.

Resultaten visar att Grammatical Framework är någorlunda framgångsrik när det kommer till att generera små Wikipedia-artiklar på olika språk. Fastän alla texter var läsbara, så hade de svenska texterna minst antal grammatiska fel. Den största nackdelen är den regel i Grammatical Framework som inte tillåter mönstermatchning med run-time argument, vilket begränsar programmeraren då många funktioner i resursgrammatiken använder möstermatching internt på sina argument.

Även om Grammatical Framework inte löser hela problemet så är det ett tillräckligt kraftfullt verktyg för att vara lämpat till att användas vid textgenerering, där den största fördelen är att den avlastar programmeraren från att behöva tänka på böjning och andra grammatiska aspekter.

€€€€,
"Keywords[swe]": €€€€
Grammatical Framework, Beräkningslingvistik, Textgenerering, Datavetenskap €€€€,
}

acronyms.tex

```
%%% Local Variables:
%%% mode: latex
%%% TeX-master: t
%%% End:
\setabbreviationstyle[acronym]{long-short}
% The form of the entries in this file is \newacronym{label}{acronym}{phrase}
%                                      or \newacronym[options]{label}{acronym}{phrase}
% see "User Manual for glossaries.sty" for the  details about the options, one example is shown below
% note the specification of the long form plural in the line below
\newacronym[longplural={Debugging Information Entities}]{DIE}{DIE}{Debugging Information Entity}
%
% The following example also uses options
\newacronym[plural={OSes}, firstplural={operating systems (OSes)}]{OS}{OS}{operating system}

% note the use of a non-breaking dash in long text for the following acronym
\newacronym{IQL}{IQL}{Independent Q-Learning}

\newacronym{KTH}{KTH}{KTH Royal Institute of Technology}

\newacronym{LAN}{LAN}{Local Area Network}
% note the use of a non-breaking dash in the following acronym
\newacronym{WiFi}{Wi-Fi}{Wireless Fidelity}

\newacronym{WLAN}{WLAN}{Wireless Local Area Network}
\newacronym{UN}{UN}{United Nations}
\newacronym{SDG}{SDG}{Sustainable Development Goal}


\newacronym{GF}{GF}{Grammatical Framework}
\newacronym{UD}{UD}{Universal Dependencies}
\newacronym{NLG}{NLG}{Natural Language Generation}
\newacronym{AST}{AST}{Abstract Syntax Tree}
\newacronym{RGL}{RGL}{Resource Grammar Library}
```