THESIS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

Frontiers of Multilingual Grammar Development

Ramona Enache

CHALMERS | GÖTEBORG UNIVERSITY

Department of Computer Science and Engineering
Chalmers University of Technology and Göteborg University
SE-412 96 Göteborg
Sweden

Göteborg, 2013

Frontiers of Multilingual Grammar Development
Ramona Enache
ISBN 978-91-628-8787-2

Technical Report no. 99D
Department of Computer Science and Engineering
Language Technology Research Group

Department of Computer Science and Engineering
Chalmers University of Technology and Göteborg University
SE-412 96 Göteborg, Sweden

Printed at Chalmers, Göteborg, 2013

Abstract

The thesis explores a number of ways for developing multilingual grammars written in GF (Grammatical Framework). The goal is to enhance both the coverage of the grammars, in terms of content and number of languages, and to reduce the development effort by automating a larger part of the process.

The first direction in grammar development targets the creation of general language resources. These are the starting point for building domain-specific grammars for the language. Developing resource grammars gives a good overview of the effort required and provides a solid base for subsequent experiments in automation. Our work resulted in building computational grammars for Romanian and Swedish.

A further development step is multilingual domain-specific grammar creation. The technique we employed is converting structured models into grammars, which preserves the original structure of the model as a backbone of the grammar and uses the general GF resources for a smooth multilingual verbalization of the model. The use cases considered are an upper-domain ontology, a business model and an ontology describing cultural heritage artefacts, each posing a different challenge and illustrating another aspect of the GF grammars-ontology interoperability and its advantages.

An orthogonal approach to multilingual grammar development aims at increasing the number of languages from a domain grammar. Our solution is an example-based prototype which partially replaces grammar programming with feedback from native informants and SMT tools (such as Google Translate).

Last but not least, as an attempt to not only enhance GF grammars, but also use them in a novel way, we present the grammar-based hybrid system architecture combining GF grammars and SMT systems. This marks some of the first steps in using grammars for translating free text. As a side-effect of the work, we propose a technique for building bilingual GF lexicon resources from SMT phrase tables.

Keywords: multilingual grammar development, ontology verbalization, resource grammar development, hybrid machine translation, functional programming, domain specific languages.

Acknowledgements

The thesis is dedicated first and foremost to my dearest grandmother Gica, who raised me since I was 2 weeks old, taught me everything about life that's worth knowing and has been my best friend even before I knew what a best friend was. Unfortunately, she won't read these lines, as she left this world in March 2010, one week after I started PhD studies and went to a better place. Even so, there's no day that passes by, without me thinking about her and about all her unconditional love and support.

On my path to PhD studies, for every crossroad I found, there was also someone who steered me in the right direction and I am very grateful for that! It all started with my Mathematics teacher from high-school, Mr. Constantin Ursu, who believed that even I can do Math, and eventually convinced me of that, too! It's due to him I chose to study Mathematics and Computer Science at the University, which I still regard as one of my wisest choices to date.

Later on, during my BSc studies at the University of Bucharest, I got exposed to research for the first time by joining the GLAU research group (Group for Logic and Universal Algebra), thanks to Prof. George Georgescu, the supervisor of my BSc thesis. It's due to him that I discovered how amazing research is and that I want to continue my studies and become a researcher myself.

With this thought in mind and also longing for a change of scenery and new challenges, I came to Sweden for MSc studies in 2008. There I was lucky to meet Krasimir Angelov, from whom I heard about Language Technology for the first time. It was our stimulating discussions and the amazing PhD defence of Björn Bringert, which convinced me that there's nothing else I would rather do than Language Technology! I would like to also thank Krasimir for introducing me to the GF formalism, with which I worked in the next 5 years, for being a great supervisor for the MSc thesis and a good colleague and collaborator ever since.

Regarding the time of my PhD studies, in the first place, I would like to thank my supervisor, Aarne Ranta, for being such an inspiring researcher. I greatly appreciate working with him, his patience and vast knowledge. Also, I'm grateful that he provided collaboration opportunities for me with academic and industrial partners during the MOLTO project and gave me freedom to find my own research path.

I would also like to thank my co-supervisor Koen Claessen, with whom it's always so inspiring to discuss research and not only. Our meeting were always too short! Luckily, we will get to work together more in the future project, where I will continue as a post-doc.

Many thanks also go to my examiner, Patrik Jansson, who gave valuable feedback at each follow-up meeting and on the thesis manuscript. Also, I would like to thank Wolfgang Ahrendt, not only for his role in the follow-up committee, but also for our collaboration in the Software Engineering using Formal Method and illuminating discussions about football and life.

I would like to express my gratitude to the partners from the MOLTO project, with whom it was so inspiring to collaborate during the last 3 years. In particular, Cristina España-Bonet, Lluís Màrquez and Meritxell Gonzàlez, who hosted me at Universitat Politècnica de Catalunya, from Barcelona, during the summer of 2012. Moreover, I would like to thank Dana Dannélls for our collaboration, our refreshing walks downtown and the most inspiring discussions about everything. I would also like to thank Jeroen van Grondelle, from Be Informed, Laurette Pretorius from UNISA and Brian Davis from DERI for all the inspiration they provided during our short, but fruitful collaboration. Special thanks also go to Malin Ahlberg, whom I had the pleasure to supervise for the BSc and MSc thesis and who is a most brilliant young researcher.

I would also like to acknowledge the past and present members of the Language Technology group, which is such a great environment for research: Thomas Hallgren, Bengt Nordström, John J. Camilleri, Grégoire Détrez, Peter Ljunglöf, Håkan Burden, Olga Caprotti, Shafqat Virk, K.V.S. Prasad and Inari Listenmaa.

The Computer Science and Engineering Department from Chalmers was not only a fabulous and inspiring working environment, but also the place with the highest density of amazing people that I've ever encountered. I would like to thank from the bottom of my heart all the people from CSE who were like another family to me during these 3.5 years!

Special thanks also go to Christina Lidbeck, who helped me starting a life in Gothenburg, especially in the beginning, when times were rough. She also gave me the best advice about Swedish society and helped me integrate here. Knowing her and her family made all the difference to me!

To my dearest friends from Gothenburg – Nui, Camilo, Maryana, Gaël – and elsewhere – Oana and Andrei – for their overall awesomeness that words can't describe and for making my life wonderful, I would like to say: Kob kun ka / Gracias / Spasiba / Merci / Mulţumesc!

Last, but not least, the thesis is dedicated to the world's most wonderful mom, Lidia, to my amazing little brother, Bogdan and to my miniature muse, François-Frederic. Vă iubesc mult!

Since arriving to Sweden 5 years ago, my life has changed in so many ways. However hard I tried, I just couldn't fully describe the whole series of fortunate events that made me grow so much as a person, without risking to turn the thesis into my own Bildungsroman. Everyone that had a part in it, knows it already and they know how lucky I am that our paths crossed. As a conclusion, to all of you who made my life better, who provided inspiration, friendship and support, I would like to say (again): *Thank you so much!*

This work has been funded by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement number 247914 (MOLTO project, FP7-ICT-2009-4-247914) and by the Swedish Research Council (Vetenskapsrådet) (REMU project 2013-2017).

Contents

Chapter 1

Introduction

The thesis explores several ways of automating the development of multilingual grammars for the purpose of semantics-preserving machine translation within a limited or semi-limited domain. The grammars are written in the type-theoretical grammar formalism **GF** (Grammatical Framework) [12]. In addition to enhancing grammar development, we also present a novel way of using GF grammars - for translating free text.

The key feature of GF is the representation of a grammar as a pair consisting of an *abstract syntax* acting as a semantic interlingua along with a number of *concrete syntaxes* corresponding to target languages. Because of this division, translation is possible between any pair of languages for which a concrete syntax is defined.

In addition to this, GF is a functional programming language equipped with a run-time system featuring parsing and verbalization capabilities. Looking at GF grammars from a machine translation point of view, one can note that a grammar defines a rule-based translation system between any two concrete languages. The translation results as a composition of parsing text from the source language and linearizing the abstract syntax tree thus obtained in the target language.

The separation between the concrete syntaxes and the abstract syntax and the capability to translate between any two languages differentiates GF from existing rule-based translation systems like Apertium [2], where translation is defined only for certain pairs of closely related languages, and is defined by specific transfer rules and bilingual lexica.

The largest and most general GF grammar is the resource library [13], where the interlingua describes basic syntactic constructs such as predication or complementation that describe the grammar of a natural language. In addition to this, there are 27 concrete grammars corresponding to languages for which these features are implemented[1]. The resource library is further used for developing concrete domain grammars for languages represented in the library.

The GF formalism is comparable to theoretical grammar models like HPSG [4] and LFG [5], but it differs from them because of the specific representation of a grammar which distinguishes between the abstract and concrete parts. The GF interpretation of multilinguality is different from the HPSG and LFG one, because despite the fact that these formalisms are used to build grammars for a number of languages, GF allows translation among any pair. The reason is that the languages are strongly connected by

[1]http://www.grammaticalframework.org/lib/doc/synopsis.html

the abstract syntax. The multilingual grammar project based on HPSG is called Lingo Matrix [6] and features a grammar library for 71 languages[2]. The coverage of the languages is variable, because despite of the code sharing, the grammars are ultimately stand-alone. There is a similar multilingual grammar system for LFG, named Pargram [7], which contains 6 grammars, but have some mechanisms for parallel analysis of the languages, as there is a stronger focus on this aspect compared to Lingo.

Another category of GF grammars are domain grammars. They have a clear abstract syntax, that could abstract very much over the specific implementation details. For this reason, when writing the concrete grammars, one benefits greatly from using and combining together the basic constructs from the resource library, and not encode again the language-specific features. In this way, the domain grammar writer need not have linguistic training, but only knowledge of the domain and the languages for which she develops concrete grammars.

For this reason, the creation of a resource grammar facilitates the future development of several application grammars for the given languages. Moreover, a resource grammar is a valuable resource by itself, due to the fact that the GF runtime system provides a parser and linearizer. In this way it can be used for other natural language processing applications. One important point to mention is the fact that the GF project is entirely open-source and the grammars are freely available to be used and modified. This makes a difference for languages where the computational linguistic resources are scarce and mostly proprietary.

Below is a first example of a GF domain grammar for representing Latin proverbs. For a better readability, the abstract syntax uses the English translations, though. For the moment, we assume that the basic components are noun phrases and verb phrases. The basic grammatical category for representing a proverb is the sentence, obtained by combining the noun phrase, acting as a subject and verb phrase, acting as a predicate:

```
abstract Gram1 = {

cat NP, VP, S ;

fun
Time   : NP ;
Fly    : VP ;
MkS : NP -> VP -> S ;
}
```

The abstract syntax from above describes the 3 categories - `NP`, `VP`, `S`. In addition to this, we have the functions, which can be nullary (lexical items like `Time` and `Fly`) or require a number of arguments (like `MkS`).

Further more, we will give two concrete grammars corresponding to English and Latin:

```
concrete Gram1Eng of Gram1 = {

lincat NP, VP, S = {s : Str} ;

lin
Time         = {s = "time"} ;
Fly          = {s = "flies"} ;
MkS np vp = {s = np.s ++ vp.s} ;
```

[2]http://depts.washington.edu/uwcl/twiki/bin/view.cgi/Main/LanguagesList

```
}


concrete Gram1Lat of Gram1 = {

lincat NP, VP, S = {s : Str} ;

lin
Time        = {s = "tempus"} ;
Fly         = {s = "fugit"} ;
MkS np vp = {s = np.s ++ vp.s} ;
}
```

In this simple example we only use one form for each word, which is adequate for the purpose of the grammar - translating the latin proverb *Tempus fugit* into a number of languages. Below is a tentative sketch of the same grammar, ported to 8 more languages:

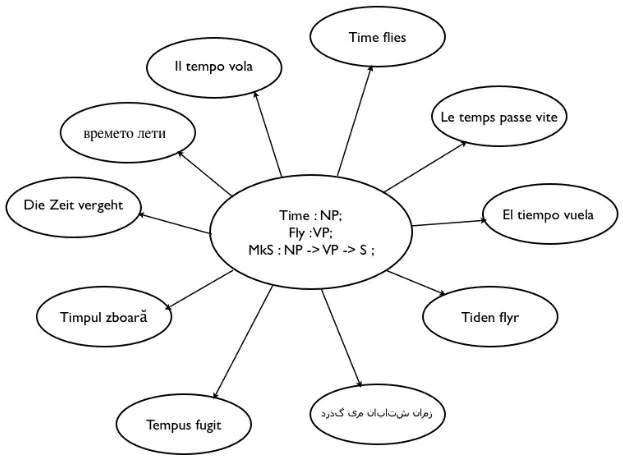


Figure 1.1: *Time flies* in English, French, Spanish, Swedish, Farsi, Latin, Romanian, German, Bulgarian and Italian

It could appear that GF grammars rely on replacing words with their translations, but adding more languages show that this sort of approach does not scale up well. First, one can see that the Latin correspondent for `Fly`, literally means *run*. Also, for other languages the translation is a phrase that has a different grammatical structure than the English counterpart. For example in French, the translation is *Le temps passe vite* (The time passes quickly), which would map `Time` to *le temps* and `Fly` to *passe vite*. One can see that the mapping is dictated by the context, and is not the most likely translation of each word.

If one would like to scale up the grammar to include more proverbs, then the string representation could prove to be insufficient, because one needs to take into account declension forms of the nouns and verbs from the grammar. If one wants to introduce complementation, the situation becomes even more complicated because of word order and clitics.

Hence, a more robust representation of the same grammar would be:

```
abstract Gram2 = Cat **{

fun
Time   : NP ;
Fly    : VP ;
MkS    : NP -> VP -> S ;
}
```

For this reason, domain grammars like the one described above are normally never built in this manner and then scaled up, but are developed on top of the resource grammar library, from where they use the syntactic categories and operations.

The English concrete syntax in this case would be:

```
concrete Gram2Eng of Gram2 = CatEng **
                        open SyntaxEng,
                             ParadigmsEng,
                             IrregEng in {

lin
Time       = mkNP (mkN "time") ;
Fly        = mkVP fly_V ;
MkS np vp = mkS (mkCl np vp) ;

}
```

Here we import the modules `ParadigmsEng`, `SyntaxEng` and `IrregEng` from the English resource grammar in order to write the linearizations of the functions in a more scalable way. For `Time`, we use the function `mkN` from `ParadigmsEng`, for getting the declension forms of the noun *time*. On top of that, we use the function `mkNP` from `SyntaxEng` for creating the noun phrase without a determiner which corresponds to the noun. For `Fly`, as it is an irregular verb in English, we get the correct conjugation forms from `IrregEng` and create a verb phrase on top of it with `mkVP` from `SyntaxEng`. The function that creates a sentence, `MkS` is obtained as a composition of the functions `mkS` and `mkCl` from `SyntaxEng`. The reason is that the resource grammar needs to build the intermediate category `Cl` (clause), before creating a sentence. The clause implements all combinations of times, polarities and topicalizations of the sentence in cause. The function `mkS` that we used selects the default parameters – present tense, positive polarity and direct topicalization.

Although the development effort might seem greater for the second grammar, work on larger examples showed the importance of developing the concrete syntaxes in a structured and robust manner, as illustrated by the grammars described in the **Contribution** subsection. This way of writing GF grammars is also featured as a best practice in the reference literature for GF programming [12], [8].

It might not be directly noticeable from the example above, but the coverage of a GF grammar is strictly limited to the language that it defines. One can identify two important aspects that limit GF grammar coverage: lexicon resources and syntactic constructions. However, GF grammars compensate by providing an in-depth analysis of the accepted input in the form of the abstract syntax tree, which can be exploited for a more sophisticated processing of the input.

For this reason, GF grammars are better at describing controlled natural languages than real-life text, although substantial work in this direction is in progress [24]. For this reason, despite their potential usage for machine translation, they have a different focus than mainstream phrase-based statistics-based translating tools [10] such as Google translate [11]. This is because the statistic approach to machine translation favours coverage and provides a translation for any input, although there are no guarantees on the correctness of the result. The results are better when translating from another language into English, because of the simple English morphology and word order and also because large corpora are available [12]. However, it is not the case that any pair of languages has a large bilingual corpus, and in this case the results will not be of the same quality. With the GF approach, the coverage (as defined by the abstract syntax) is the same for any language in the resource library. Regarding the quality of the translation, GF offers the advantage that the source of errors is easier to spot and fix because of the fact that grammars are viewed as programs and more control over their content is possible.

GF has been the main technology in the European project MOLTO (Multilingual Online Translation)[3] that aimed at making multilingual grammars scale up in more directions, namely to fit larger domains and real-world text, to make grammar development accessible to a larger category of people, without prior linguistic or programming training and to reduce the effort for developing grammars in general. The thesis was carried out as a part of the MOLTO project, embraces its goals and tries to contribute to their fulfilment.

Overall, the thesis gives a comprehensive overview of both traditional and novel ways of developing multilingual GF grammars. From creating language resources as part of the GF resource library to combining external resources for semi-automated grammar development and using GF grammars for developing hybrid translation systems, the work investigates the process of multilingual grammar development and usage from a number of angles. The emerging results show the potential that automating grammar development has in terms of making grammar programming easier, faster and more scalable. Moreover, we analyse the usage of GF grammars both in a controlled context and for handling legacy text.

1 Contributions

The thesis addresses three main directions of grammar development – the creation of language resources, grammars verbalizing structured models, bootstrapping grammars from external resources and an emerging direction for grammar utilization – the development of grammar-based hybrid translation systems.

More concretely, the work deals with the matter of reducing the effort needed to develop multilingual grammars by using language skills from SMT systems and native informants in order to build a concrete domain grammar. Moreover, we investigate the grammar-ontology interoperability, by projecting structured models into GF grammars, which we further use for verbalizing the models. Last but not least, the effects of integrating GF in a hybrid system for translating patent claims from the biomedical domain are discussed.

1.1 Creating Language Resources

Creating high-quality general language resources has always been one of the most important direction in computational linguistics. They represent perennial assets that aid

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

the further development of computational resources for the language. In GF, having a general resource for a language opens the way for developing domain grammars for the language and integrating them in multilingual applications. Creating and enhancing resource grammars is a constant priority for the GF community because the experience for developing one resource grammar can be easily transferred to benefit resources for similar languages, thus leading to an even faster growth of the GF resource library which currently contains 27 languages, as of August 2013.

An Open-Source Computational Grammar for Romanian

The first major contribution in chronological order that the thesis makes, is "An Open-Source Computational Grammar for Romanian", which describes the resource grammar for Romanian. To our knowledge, it is the only open-source computational grammar for Romanian, and also the first computational grammar that deals with the Romanian clitics - an interesting and complex linguistic phenomenon. Moreover the Romanian clitic system is different than the clitic system from the other languages in the Romance family. The work on the Romanian resource grammar made it possible to develop other GF domain grammars such as the Phrasebook and SUMO-GF for Romanian. From the author's perspective, the work on the Romanian resource grammar is meaningful as it gives an overview of the effort that a GF grammar needs and the knowledge gained after completing this work inspired some directions in which the grammar writing workflow can be aided.

An example that illustrates the Romanian clitics system is the English sentence *I heard my friend* which would be translated to Romanian as *Eu l-am auzit pe prietenul meu* where the following annotated version indicates the details of the analysis:

Eu *[I, nominative]* l-*[he, accusative]* am auzit *[hear, past tense]* pe *[accusative preposition for animate direct objects]* prietenul *[friend, accusative]* meu *[my, masculin, singular].*

The sentence illustrates the phenomenon of clitic doubling of animate nouns in Romanian - unique in the Romance language family. It applies to animate direct objects, which can be proper nouns denoting people, pronouns (in certain cases for stylistic purposes) and common noun phrases in definite form. In this example, the noun phrase *prietenul meu* (my friend) is preceded by *l-*, the clitic corresponding to the accusative form of the third person singular pronoun.

Moreover, clitics can be combined, as it happens with two-place verbs, and there is a systematic, yet complex set of rules that specify the process and which is described in the paper.

The paper was published in In A. Geldbuch (Ed.), Intelligent Text Processing and Computational Linguistics Conference (CICLing -2010), Iasi, Romania, March 2010, LNCS 6008. My contribution to the work is the development of the main part of the Romanian resource grammar, reflected in the paper by parts 2-6.

A Type-Theoretical Wide-Coverage Computational Grammar for Swedish

A second direction in developing language resources is the work on an enhanced version of the Swedish resource grammar. The resource was used for parsing Talbanken [8], an open-source manually-built treebank containing around 6,000 sentences from newspaper text.

Previous work of the same authors [14] presents the development of resources - the extraction of a large-scale GF lexicon from the SALDO resource [15] and an interactive lexicon acquisition tool for unknown words from Talbanken, as well as a mapping strategy from treebank trees to GF trees.

The current paper focuses on extending the Swedish resource grammars with common Swedish-specific constructions from the treebank, such as the s-passive. Moreover, the current grammar extends the type system with dependent types for encoding reflexive-possessive pronouns:

Han har tappat *sina* vantar. (He lost his (own) gloves)

Hans vantar är borta. (His (someone's) gloves are gone)

The rule is that one uses the reflexive-possessive forms *sin/sitt/sina* only when they refer to an object in the sentence. Otherwise the personal pronoun form is used *hans/hennes/deras*. One can also use the personal pronoun form for objects, but then it means that the pronouns refer to another person. For example:

Han väntar på sin kompis. (He waits for his (own) friend)

Han väntar på hans kompis. (He waits for his (someone else's) friend)

We solve this problem by extending the data type for noun phrases with an argument that is either `Object` or `Subject`. When expressing the other syntactic constructs, we either require a certain argument – for predication and direct complementation, or pass the argument along:

```
PredVP : NP Subject -> VP -> Cl ; -- predication
ComplSlash : VPSlash -> NP Object -> VP ; -- complementation
PrepNP : (a : NPType) -> Prep -> NP a -> Adv a ;
                             -- building an adverbial phrase
```

In addition to the extra grammar constructs, the paper describes further work on creating a GF treebank from Talbanken, chunk parsing of free text and treebank-based disambiguation of parse trees.

The paper is based on the results of the MSc thesis of Malin Ahlberg [16], supervised by the author and was published in the Proceedings of the International Conference on Text, Data and Speech (TSD) September 2012, LNCS 7499. My contribution to the work is mainly in the GF part and the high-level parts of the algorithms. The authors contributed to the paper in equal amounts.

1.2 Grammars Describing Structured Models

The work on representing structured models in GF investigates a number of solutions to the grammar-ontology interoperability problem and the advantages of using GF as a host language for structured models, both in terms of type system and possibilities for multilingual verbalization.

The work on encoding the SUMO ontology in GF focuses more on the type system that models the structure of the ontology, whereas the work on verbalizing the BeInformed business model features a simpler type system, but puts more emphasis on verbalization mechanisms. In addition to this, work on ontology representation in GF featured a multilingual grammar for the CIDOC-CRM ontology describing paintings and related concepts which was extended with 15,000 instances from the Gothenburg City Museum [17], [18], [19], [20] and [21]. The grammar encoding the ontology also features a simple type system and puts more focus on verbalization, as it generates multi-sentence descriptions of the painting and aims at high-quality and fluency of the resulting texts.

Typeful Ontologies with Direct Multilingual Verbalization

The paper describes a type-theoretical grammar that can model the concepts and relations from an ontology, along with the benefits of this encoding for ontology verbalization. As a proof of concept, a large part of SUMO (Suggested Upper-Merged Ontology) [5], the largest open-source ontology, was represented in GF. The results show that in term of ontology reasoning capabilities, SUMO-GF has a coverage which is comparable to the original SUMO, whereas regarding natural language generation capabilities, the results obtained from SUMO-GF, by providing concrete grammars for English, Romanian and French, even though obtained with straightforward techniques, proved to be superior from syntactical correctness point of view and readability to the ones obtained for SUMO with external verbalization tools, especially for Romanian and French. The results obtained with the SUMO-GF ontology are promising and show that GF is a good environment for representing ontologies, which could be incorporated as part of other grammars in order to provide a semantically robust abstract syntax, that is easier to fit the concrete syntaxes.

An example of a SUMO axiomatic construction that can be formed in the original ontology is:

equal (ComplementFn (IntersectionFn ?X ?Y)) NullSet

which expresses the fact that the complement of the intersection of two sets, ?X and ?Y is null. The same axiom expressed in SUMO-GF is:

```
forall SetOrClass (\X →
    forall SetOrClass (\Y →
        equal (el (ComplementFn (el (IntersectionFn (var X) (var y))))) (el NullSet)))
```

Since axioms are closed expressions, the lack of quantification entails that the variables are quantified universally by default. Also, since GF is strictly typed and there is no type information about ?X and ?Y, we need to perform type inference on the variables. Since they are used by the function *IntersectionFn* that takes 2 arguments of type *SetOrClass*, we infer that they should be of a type coercible to *SetOrClass*, and because we lack further information to make other inferences, we just assign the type *SetOrClass* to the variables.

The wrapper functions *el* for function results and *var* for variables, are coercion functions which require proof objects that a certain type coercion is possible in order to asses that no type error occurs in the axiom. For example the *el* that takes *ComplementFn* as argument, would need a coercion between *SetOrClass*, the return type of *ComplementFn* and *Entity*, the type that *equal* expects as argument.

Regarding the verbalization capabilities, the axiom would generate the following sentences:

for every set or class X and every set or class Y, we have that the complement of the intersection of X and Y is equal to the null set (English)

pour chaque ensemble ou classe X et chaque ensemble ou classe Y, le complément de l'intersection de X et Y est égal à l'ensemble nul (French) *pentru fiecare mulţime sau clasă X şi fiecare mulţime sau clasă X, complementara intersecţiei lui X şi Y este egală cu mulţimea vidă* (Romanian)

The paper was published in the LNCS Post-Proceedings of the Controlled Natural Languages Workshop (CNL 2010), Marettimo, Italy, November 2011. My contribution is the representation of the SUMO ontology in GF, the multilingual natural language generation part and the investigation on the automated reasoning capabilities of the new ontology, reflected in the paper by parts 1-6, 8, 9.

Multilingual Verbalization of Modular Ontologies using GF and lemon

The work on verbalizing the BeInformed[4] business model represents one step forward in representing and verbalizing ontologies in GF. The focus here shifts from building a type-theoretical framework for encoding the ontology structure to having an efficient and practical solution that could fit an industrial project. As the use of GF is mainly for the verbalization part, the type system has been kept simple, by translating concepts to GF categories and ontological relations to simple GF functions. For example:

```
cat
  Activity;
  Fragment; -- GF category for predicate verbalization

fun
  requires_completed : Activity -> Activity -> Fragment ;
```

One important change in the current grammar representation is the division into

- **T-box** - encoding the basic structure of the ontology (concepts and relations)
- **A-box** - encoding instances from the ontology

The difference is that the type system and the primitives for the concrete grammar that are defined in the T-box can be later used to extend the A-box with more instances, while preserving the same basic structure of the ontology encoding. The goal is to build the A-box automatically from external sources (*lemon*) or developers of the ontology without GF training, after the T-box has been built for a fragment of the ontology.

This is one more approach to automate grammar development, by importing the abstract syntax of the grammar directly from the ontology (A-box and T-box), developing the T-box concrete grammars by GF experts and building the A-box concrete grammars in an automated way.

Going back to the T-box concrete grammars, it is important to mention that the grammar verbalizing the ontology features more complex linguistic representations of the concepts. For example, for `Activity`, which is normally verbalized as a noun phrase, we offer the possibility to verbalize it as a sentence, for particular shapes of instance labels. It is the case of verb phrases with a complement used in gerund form such as `PublishingOfResult`, which can either be verbalized in the noun phrase form *publishing the result* or in a sentence form *the result is published* (when the activity is completed). This becomes more obvious when looking at the verbalization of the predicate `requires_completed` applied to the activities `Intake` and `PublishingOfResult`, which would render:

[4]http://www.beinformed.com/

Intake is completed if the results are published

The paper was published in the LNCS Post-Proceedings of the Controlled Natural Languages Workshop (CNL 2012), Zurich, Switzerland, September 2012. My contribution is the architecture of the grammar verbalizing the business model (A-box and T-box representations), reflected in the paper by parts 4.1 (GF introduction), 4.2, 4.3, 4.4.

Multilingual Grammar for Museum Object Descriptions

The final contribution on the topic of ontology representation as GF grammars is the work on describing cultural artefacts from the Gothenburg City Museum (GCM). Along the way, a number of solutions were proposed for this problem, since the focus is both to describe the cultural heritage artefacts in a semantically consistent manner and to generate high-quality descriptions, given the information existing in the database.

The first solution, described in [17] verbalizes the concepts from the CIDOC-CRM ontology and its instances from the GCM database in a manner identical to the representation of the SUMO ontology. The reason is that the CIDOC-CRM ontology, builds on SUMO, which provided a good starting point for the work. The downside is that the SUMO grammar was not tailored for verbalization, so the output that the first GF grammar for cultural heritage could provide is of the shape:

Big Garden is a painting.

Big Garden is painted on canvas.

Big Garden is painted by Carl Larsson.

Big Garden was created in 1937.

Since the target is a paragraph-like text and not single sentences, and the ontology does not need consistency checks, a new structure of the grammars was needed. The new approach, briefly described in [20] is based on studies [23], [24] on the most common linguistic patterns for describing cultural heritage objects. The patterns were translated into GF functions and some examples of these have the following signature:

DP0 : Painting -> Painter -> Year -> Text ;

DP1 : Painting -> Museum -> Painter -> Size -> Text ;

DP2 : Painting -> Painter -> Material -> Year -> Text ;

The representation of cultural Heritage objects is a done via a dependent type encoding all known features that can be found in the database. Since not all artefacts have all features that a description pattern would require, the missing features are also encoded, in order to allow a single dependent type (`PaintingDescription`) to encode all objects. The solution that differentiates between known and unknown features is inspired from the `Maybe` type from Haskell, and would use a special object to denote that a certain feature is missing. For example, `NoSize` indicates that the size of a painting is unknown.

The semantic definitions of the pattern functions match on missing features and will generate no text in case that a relevant one is missing. For example, assuming that the example above is encoded in our new system as

```
fun Obj : PaintingDescription BigGarden CarlLarsson
Y1937 NoMuseum NoColour NoSize Wood ;
```

We could not apply `DP1` to it, because we have no information on the size and museum. On the other hand, by applying `DP0`, we would get *Big Garden was painted by Carl Larsson in 1937.*

The advantage of the method is that the implementation of each pattern yields a coherent paragraph. Also, due to the multilingual context, it is more advantageous to aim for a higher-level structure, such as the paragraph, since the syntactic structure at sentence level can be different across languages – for example the use of the passive voice. The disadvantage is that each pattern needs to be implemented separately, which entails code duplication. Also one needs to impose a certain order on the patterns and to select the most informative pattern that could describe a given artefact by methods external to the grammar.

The final solution, which is currently in use and is described in the MOLTO Deliverable 8.2 builds on the previous one, replacing the patterns described above with only one pattern that finds the most comprehensive description of the artefact, by pattern matching on its features. In this way, we retain the unique representation of artefacts with the dependent type `PaintingDescription`, which ensures the semantic consistency of the descriptions, and we use a single verbalization function, which combines the information described in the previous patterns, allowing for potential paraphrasing.

Further work on the Cultural Heritage use case, not included in the thesis, targets the integration of the CIDOC-CRM ontology with the database entries from GCM [25], the lexicalization and multilingual translation of database entities and work on a multilingual query grammar for cultural heritage artefacts [26].

The manuscript corresponds to the Deliverable 8.2, where the authors contributed in equal amounts. Regarding the writing part, the author is responsible for 3.1, 3.2, 3.3 and partly 4.

1.3 Bootstrapping Grammars from External Sources

As a natural direction for scaling up GF grammars, one can try to reduce the effort for grammar development by automatically importing parts of the grammar from external sources. Whereas representing ontologies in GF already showed how one can import an abstract syntax into GF, the largest bottleneck of automating grammar development is still building the concrete syntax. Here, the two main challenges, in order of their difficulty, are building lexical resources and verbalizing complex functions.

Steps in automating the acquisition of a multilingual lexicon from an aligned corpus with the help of SMT lexical tables are also described in [2] and [28]. In addition to this, there has been considerable work on porting monolingual morphological resources to GF in order to aid further grammar development.

In terms of automating the verbalization of functions, we employed the example-based grammar writing technique, for which a prototype has been implemented and tested for building concrete grammars for a Tourist Phrasebook. However, the process is not fully automatic yet, because for complex concepts like the ones described in the

T-box verbalizing the BeInformed business model, one still needs manual intervention. However, the encouraging results showed by the work on the Phrasebook, show that bootstrapping is an effective tool for grammar development.

Controlled Language for Everyday Use: the MOLTO Phrasebook

The paper "Controlled Language for Everyday Use: the MOLTO Phrasebook", describes the development of a tourist phrasebook grammar available in 14 languages, where a considerable part was developed as a part of an experiment to automate the development of concrete syntaxes and to investigate the relation between language skills, GF programming skills and the effort needed to develop a concrete syntax for a medium scale grammar. The results of the experiment show that in principle, one need not have language skills in order to develop a concrete syntax, provided that one can use the resource grammar for the given language and use the language skills of native informants or statistical tools. In this way, both syntactic constructions and unknown lexicon entries can be added, with the condition that the newly acquired words need to be POS-tagged and lemmatised. Not surprisingly, the effort was proportional to the morphological and syntactical complexity of the language, but the example-based method proved to be quite effective in alleviating the burden of writing linearizations manually for syntactically complex constructions and for combining the work of the GF developer and the human informant or SMT system. In the future, the method will be available as a GF development tool, aimed at reducing the grammar-writing effort by reducing the need of manual GF programming for concrete grammar development.

An example for developing the Phrasebook with the example-based method is the case of the question *What is your name?* for German. A native speaker would be asked to translate the question and the answer, *Wie heißt du?* would be parsed with the German resource grammar parser, assuming that all words are available in the lexicon. From the resulting expression, we generalize over *you* since the initial function should allow any pronoun to appear as argument.

The end result looks like:

QWhatName p = mkQS (mkQCl how_IAdv (mkCl p.name heißen_V)) ;

which would be literally translated to *How you 'are_called'?*, where the verb *heißen* is translated by *are_called*, with the difference that the verb is not a passive voice, but a direct equivalent active verb doesn't exist in English. One can see that the German phrase differs substantially from the English one:

QWhatName p = mkQS (mkQCl whatSg_IP (mkVP (nameOf p))) ;

The native informant can be given another example of the *QWhatName* function, used with a different argument this time, for testing.

The advantage of using the example-based method is that the parser helps abstracting over the difficulties of expressing a more complex syntactic construction, like the one above, making it possible to combine resources more efficiently and speed up grammar development.

The paper was published in the LNCS Post-Proceedings of the Controlled Natural Languages Workshop (CNL 2010), Marettimo, Italy, November 2011. My contributions are developing the concrete Phrasebook grammars for 4 languages within the example-based grammar writing experiment, contributing to the development of the

GF runtime system in Java for the Phrasebook application on the Android platform and to the abstract syntax of the Phrasebook at a later stage. The main contribution is the example-based system prototype and the algorithm to write grammars by examples, reflected in part 5 of the paper.

1.4 Grammar-Based Hybrid Systems for Machine Translation

In addition to the ways of automating grammar development presented before, work has also been done on using grammars in combination with SMT for building hybrid machine translation systems. Since the pros and cons of the two systems are complementary, the goal of the hybrid system is to get the best of both worlds - wide coverage from the SMT and syntactic knowledge from the grammar.

There have been a number of approaches for building a GF-based hybrid translation system in the MOLTO project. Chronologically, the first one is a bilingual grammar (French and English) extended with a lexicon extracted from the phrase tables of a state-of-the-art MOSES system. The grammar was used for translating patent claims from the biomedical domain from English to French, in the first experiment of using GF on free text [2].

In addition to this, there has been work done on developing a robust GF parser written in C [24] and using it along with a bilingual dictionary extracted from WordNet [29] for translating free text from English into Finnish, Urdu, German and Bulgarian [28].

Patent Translation within the MOLTO Project

The first paper on combining GF with SMT tools, named "Patent Translation within the MOLTO Project" marks the first experiences in using GF for legacy text, by building a grammar for translating English patent claims from the biomedical domain to French. As mentioned before, GF grammars are limited first by the lexicon and secondly by the syntactic structures that they describe. A solution to the first problem was to build the lexicon corresponding to the source English text by POS-tagging them with a tagger trained on the biomedical domain [14], lemmatise the results and build a lexicon grammar from them. The French lexicon is obtained with SMT methods. In addition to this, the syntactic structure of patent claims has been studied, and significant constructions which were not covered by the resource library were added to a patent claims grammar. Although in terms of lexicon coverage, the method appears to achieve the desired result, in terms of syntax coverage, the grammar only covered around 15% of the full patent claims. However, initial experiments showed that the coverage is considerably better for syntactic chunks, which could be recombined at a later stage, which is an interesting direction for future work. On the other hand, the SMT baseline system trained on patent claims achieves better results than other general SMT systems like Google Translate and Systran for translating patent claims (as of 2011). The main problems with the translation are syntactical errors that could affect understanding, such as problems with the agreement for long-distance dependencies and translation of chemical formulas.

The patent claim come from the MAREC corpus belonging to the European Patents Office[5], used in a patent retrieval task during the CLEF 2010 Conference[6].

An example of such a claim is

[5]http://www.epo.org/

[6]http://clef2010.org/

The pharmaceutical composition of claim 1 , wherein the aqueous solution of arginine and ibuprofen has been lyophilized.

which the SMT system trained on patent claims would translate to:

Composition pharmaceutique selon la revendication 1 , dans lequel la solution aqueuse de l' arginine et l' ibuprofène a été lyophilisée.

whereas, the standard translation is:

*Composition pharmaceutique selon la revendication 1 , **dans laquelle** la solution aqueuse **d'** arginine et **d'** ibuprofène **est** lyophilisée.*

By looking at the changes between the two French claims, as highlighted in the standard translation, one can notice that although they don't affect the understanding for this claim, they are grammatically incorrect and could make the text harder to read and understand.

Regarding the GF translation of the same claim, several parse trees corresponding to the English claim have been found. The closest to the reference translation is:

*Composition pharmaceutique selon la revendication 1 , dans laquelle la solution aqueuse d' arginine et d' ibuprofène **a été** lyophilisée.*

The only difference between the GF and the standard translation is the use of past tense (*passé composé*) for the innermost subordinate clause in the GF translation, whereas the reference one uses present. Since the original English claim uses present perfect, and in French no tense would have the same functionality, the choice between present/past tense is still debatable. Still there are sources[7] claiming that the past tense in French (*passé composé*) is the most grammatically similar to the present perfect tense from English. The two agreement errors that the SMT translation displayed, did not occur in the grammar-based translation.

We noted that for the claims that the GF patents grammar could parse, the translated results are better than the SMT system, but since the coverage of the grammar is still very low and the claims that it can parse are rather simple, there is a great need to make the two technologies interact on a deeper level in order to have a robust hybrid translation system.

The paper was accepted for publishing in the Proceedings of the 4th Workshop on Patent Translation, MT Summit XIII, Xiamen, China, September 2011. My contribution was the part about the GF grammar for patent claims, reflected in parts 2, 3.1 and 4.1.

A Hybrid System for Patent Translation

The paper "A Hybrid System for Patent Translation'" presents a second experiment in building a GF-based hybrid system for parsing patent claims, after [2]. The system was used for translating the same corpus as before, from English to French.

The grammars are used in combination with a state of the art SMT system for translating English patent claims from the biomedical domain from English to French and German.

[7]http://en.wikipedia.org/wiki/Present_perfect#French

The key idea is to split claims into chunks first, so that the grammar has a better chance of parsing them, since the previous experiment showed the limited coverage of GF grammars, mainly due to their strictness when it comes to syntactic constructs.

Further on, the chunks that can be translated (parsed and linearized) are included, along with their translation in the Moses phrase tables and used by the SMT system in translation. A concrete example showing how the GF-based system would translate an English claim can be found in the paper.

An important part of the process is building a good bilingual lexicon, which is done with the help of the same tool used for chunking - Genia. From the English words lemmatized by Genia, we find the French correspondents with SMT methods. The GF representations of the words are found with the help of the large monolingual dictionaries for English and French and the mechanisms provided by the GF resource grammars for inferring the additional forms of a lexical entry based on the lemma.

Since for the pair English-French, the results obtained by the SMT system were very good already, the main improvement that GF aims to bring is to increase syntactic correctness of the translations, as one of the largest problems that the SMT translations have are agreement errors which could affect understanding. For example, the translation of the *the pharmaceutical composition of claim 1, wherein ...* is translated by the SMT system as *composition pharmaceutique selon la revendication 1, **dans lequel**...*, where *dans lequel* is the French translation for *wherein*. The only problem is that the agreement is not correct, since the relative pronoun should agree with the noun that it determines (*composition*), which in French is feminine. The hybrid system renders the correct translation *composition pharmaceutique selon la revendication 1, **dans laquelle***.

The results of the automated evaluation show a slight improvement of the hybrid system compared to the SMT and GF components, but the differences are small. However, human evaluation favoured the hybrid system in a more definite manner.

The paper was published in the Proceedings of the 16th Annual Conference of the European Association for Machine Translation, Trento, Italy, May 2012. My contribution was the part about the GF grammar for patent claims, reflected in parts 2, 3.1 and 4.1.

Hybrid Translation for European Biomedical Patents

The final contribution related to a grammar-based hybrid translation system, named "Hybrid Translation for European Biomedical Patents" is a manuscript based on the Deliverable 5.3 of the MOLTO project [28]. It is a direct continuation of the work described by the previous paper "A Hybrid System for Patent Translation" [31].

The main differences compared to previous work are the refinements in lexicon acquisition and the addition of German as a target language in the system.

There are 3 directions for lexicon acquisition that the work proposes, which are meant to replace the previous method, which required manual intervention in the end. Also the integration with the SMT tools is automatic, so that the whole pipelined system can be available as a demo. The approaches are:

static – builds large bilingual lexical resources for translation and does not require additional lexical resources which are built at runtime. For French, a bilingual one-to-many dictionary of almost 4,000 words is built from the SMT translation tables.

For German, in addition to the almost 40,000 words resource extracted from Wiktionary, we add a dictionary for translating German compounds to English of almost 8,000 words.

runtime safe – starts from a *core lexicon* of almost 200 words which are the most frequent in the corpus and complets it with nouns, adverbs, adjectives and verbs, tagged by the POS-tagger and translated from the lexical tables. The important constraint that the pairs of words need to fullfil is that they must both be found in the monolingual resources that exist already for English, French and German. In this way, we avoid introducing wrong declension forms in the grammar.

runtime unsafe – starts from the same *core lexicon* and adds pairs of words as described in the **runtime safe** method, with a weaker constraint on the words – nouns, adjectives and adverbs need not be present in the monolingual dictionaries, and their GF representation table will be inferred with the help of the smart paradigms [32].

Since the chunks translated with GF will be used by the Moses system, which needs probabilities in order to choose the best translations, we also assign probabilities to the translations obtained by our grammar, using all 3 lexicon acquisition techniques. More details can be found in [28].

We will reflect more on the German compound dictionary, as it is the most novel feature of the work. One can note the difficulty to translate multiword compounds, such as *Blutersatz* (*blood substitute*) with a word-to-word dictionary, which would either map the German word to *blood* or to *substitute*, but not to both, since *Blutersatz* is a `N` and *blood substitute* is a `CN` in GF.

Our method relies on building a grammar for German compounds, that reflects the rules for compounding in German. In this manner, we can express the compounds as a function of their basic constituents, in order to get the correct declension forms and gender.

Further on, we use the SMT phrase tables in order to identify German compounds along with their English translation. For the moment, we focus on noun phrase compounds, so we only retain pairs where the English side can be parsed as such by the English resource grammar. We proceed by splitting the German compounds in a greedy manner, until we find the smallest number of substrings such that they can be found in the German monolingual dictionary and they can be composed according to the rules of the above-mentioned compound grammar in order to retrieve the original word. The German compound and its English translation, thus obtained are added to the resource dictionary.

Despite the refinements in the lexical acquisition and compound integration, the performance of the hybrid system does not improve on the previous system for French and obtains lower results than the SMT system for German. The reasons might be that the SMT translation already obtained high scores which are not easy to improve, and also that for German, the chunks obtained from Genia [14] were not large enough to allow the grammar to render the correct word reordering, thus improving over SMT. However, the lexicon acquisition techniques are general and effective ways for improving the coverage of GF grammars, and could be used in future applications that automate grammar development.

The paper is available as a manuscript. My contribution was the part about the GF grammar and lexicon acquisition, reflected in parts 2.3, 3.1, 3.2 and 3.3.

2 Further Frontiers of Multilingual Grammar Development

The thesis enumerates a number of directions for automating multilingual grammar development, as well as using grammars for building hybrid systems for machine translation. Each of these directions could lead to multiple further developments.

The first of them would be the extension of the prototype for example-based grammar writing to a stand-alone application. A possibility would be to integrate the algorithm within the GF web editor[8], where the method could be combined with traditional GF programming. Moreover, having an example-based grammar writing system would increase the community of GF developers, since it would alleviate the difficulties of developing concrete syntax grammars. Not only the grammar writing effort would be reduced in this way, but also the effort for correcting and maintaining application grammars, which could make GF solutions more sustainable in the long run.

An important component of the example-based grammar writing technique is obtaining the most specific generalization, when abstracting over arguments in order to get the linearization of a function. The same technique can be used for *grammar induction* - which would allow building application grammars (abstract + concrete syntaxes) from an aligned bilingual corpus. The technique assumes that the aligned sentences from the corpus are parsed with the resource grammar for each language and then one applies the abstraction algorithm, which should find where the trees don't align and these subtrees are candidates for idiomatic phrases or unmatched word correspondences, which should be added to the grammar.

Another direction that the Phrasebook grammar inspired is a framework for grammar testing. This is an important step, especially for grammars generated from external sources or with a larger degree of automation. There is work in progress for generating the smallest number of abstract syntax trees that cover all constructions from the grammar, where the trees have roughly the same number of nodes. This is an NP-complete problem and for solving it without brute-force techniques that would not be possible to implement for large scale grammars, we are investigating the use of automated theorem proving methods.

Moreover we are investigating a novel method to test grammars, named *grammar-based grammar testing*, which generates a new grammar from the internal representation of a concrete grammar. This gives a more detailed taxonomy of the categories and functions of the grammar, since it divides categories into equivalence classes, according to their inherent parameters. For example, assuming that nouns have the representation `N = {s : Str; g : Gender}` and `Gender` can be either `Masculine` or `Feminine`, the category `N` from the generated grammar would have two subcategories `N_Masculine` and `N_Feminine`.

Similarly, the generated forms of GF functions would feature all the different forms/ behaviour patterns that the function could have. For example, the function applying the definite article to a noun would have a different behaviour for masculine and feminine nouns, if the forms of the definite article are different.

Generating such a grammar, makes it easier to profile the original grammar, and to trace the exact rules that lead to a natural language construction that the original grammar generates. The transition is between the two grammars is smooth since the concrete syntax of the generated grammar is the abstract syntax of the original grammar, so one could parse a natural language example twice and get a systematic profiling

[8]http://www.grammaticalframework.org/demos/gfse/

of all the rules that the concrete grammar used.

One more direction for grammar testing, especially considering the context of the work — medium-to-large scale grammars for more than 5 languages, resulting from a collaborative effort of several developers is ambiguity detection. This is not only a theoretically interesting problem, as it has not yet been investigated for PMCFG grammars, but it could also lead to practical improvements of using multilingual GF grammars for translation. For instance in English, the pronoun *"you"* denotes the familiar and the formal forms of the 2^{nd} person singular and plural. In most other languages, the four pronouns would not linearize into the same form, so when translating from English, a number of distinct alternatives will be displayed. A tentative solution to this problem would be to let the user choose the right one by making the context explicit (which pronoun would be used by each alternative, in our example). A grammar with this sort of additional information is called *disambiguation grammar* and was first devised for the English Phrasebook. These grammars are however hand-written and assume that all ambiguities are known before. Moreover, one needs a disambiguation grammar for each pair of languages, which makes the development effort for writing disambiguation grammars exceed the effort for developing the original concrete grammars and it also requires knowledge about all ambiguities from all languages. Having a method for automatically detecting ambiguities would make it possible to generate disambiguation grammars automatically, by analysing both grammars for ambiguities and filtering out the ones that do not make a difference in translation.

Last, but not least, a direction emerging from the work on hybrid machine translation systems is the construction of multiword lexicons from SMT phrase tables. After the experiment about generating a lexicon that covers German compound nouns and their English translation, similar experiments can be performed for other languages where compounds are frequent (such as Finnish or the Scandinavian languages). Moreover, if a bilingual lexicon for single words with multiple variants is available, one can perform a similar experiment to find multiword-to-multiword correspondences, by parsing the phrases in the phrase tables and keeping the ones that differ in the syntactic structure or where at least one of the words does not have a correspondent in the other phrase. Having a multiword lexicon would not only be a reusable GF resource, but would also give GF-based translation systems an advantage over SMT, since one of the biggest disadvantages of GF (when it can parse and translate an entry) is that the translations are literal, and do not handle idiomatic expressions in the way an SMT would do.

Additional Publications

The following articles were accepted for publication in peer-reviewed conferences and workshops during the author's PhD studies, but are not included in the thesis:

2013

1. Damova, Mariana; Dannélls, Dana; Enache, Ramona; Mateva, Maria; Ranta, Aarne: **Natural Language Interaction with Semantic Web Knowledge Bases and LOD**. Chapter in "Towards the Multilingual Semantic Web", Springer, to appear in autumn 2013.

2. Damova, Mariana; Dannélls, Dana; Enache, Ramona; Mateva, Maria; Ranta, Aarne: **Multilingual Access to Cultural Heritage Content on the Semantic Web**. 7th Workshop on Language Technology for Cultural Heritage, Social Sciences and Humanities, ACL 2013, Sofia, Bulgaria.

3. Gonzàlez, Meritxell; Enache, Ramona; Mateva, Maria; España-Bonet, Cristina: **MT Techniques in a Retrieval System of Semantically Enriched Patents**. 14th MT Summit, System Demonstrations, September 2013, Nice, France.

2012

1. Ahlberg, Malin; Enache, Ramona: **Combining Language Resources into a Grammar-Driven Swedish Parser**. 8th International Conference on Language Resources and Evaluation (LREC'12), May 2012, Instanbul, Turkey.

2. Dannélls, Dana; Enache, Ramona; Damova, Mariana; Chechev, Milen: **Multilingual Online Generation from Semantic Web Ontologies**. World Wide Web Conference (WWW'12), April 2012, Lyon, France.

2011

1. Dannélls, Dana; Damova, Mariana; Enache, Ramona; Chechev, Milen: **A Framework for Improved Access to Museum Databases**. Language Technologies for Digital Humanities and Cultural Heritage (RANLP '11), September 2011, Hissar, Bulgaria.

2010

1. Caprotti, Olga; Angelov, Krasimir; Enache, Ramona; Hallgren, Thomas; Ranta, Aarne: **The MOLTO Phrasebook**. Swedish Language Technology Conference (SLTC'10), October 2010, Linköping, Sweden.

2. Détrez, Grégoire; Enache, Ramona: **A Framework for Multilingual Applications on the Android Platform**. Swedish Language Technology Conference (SLTC'10), October 2010, Linköping, Sweden.

Bibliography

[1] Ranta, A.: Grammatical Framework: Programming with Multilingual Grammars. CSLI Publications, Stanford (2011) ISBN-10: 1-57586-626-9 (Paper), 1-57586-627-7 (Cloth).

[2] Forcada, M., Ginestí-Rosell, M., Nordfalk, J., O'Regan, J., Ortiz-Rojas, S., Pérez-Ortiz, J., Sánchez-Martínez, F., Ramírez-Sánchez, G., Tyers, F.: Apertium: a free/open-source platform for rule-based machine translation. Machine Translation **25** (2011) 127–144 10.1007/s10590-011-9090-0.

[3] Ranta, A.: The GF resource grammar library. Linguistic Issues in Language Technology **2**(1) (2009)

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

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

[6] Bender, E.M., Flickinger, D., Oepen, S.: The grammar matrix: an open-source starter-kit for the rapid development of cross-linguistically consistent broad-coverage precision grammars. In: COLING-02 on Grammar engineering and evaluation, Morristown, NJ, USA, Association for Computational Linguistics (2002) 1–7

[7] Butt, M., Dyvik, H., King, T.H., Masuichi, H., Rohrer, C.: The parallel grammar project. In: COLING-02 on Grammar engineering and evaluation, Morristown, NJ, USA, Association for Computational Linguistics (2002) 1–7

[8] Ranta, A., Camilleri, J., Détrez, G., Enache, R., Hallgren, T.: Grammar tool manual and best practices (June 2012)

[9] Angelov, K.: The Mechanics of the Grammatical Framework. PhD thesis, Chalmers University of Technology, Gothenburg, Sweden (2011)

[10] Koehn, P., Och, F.J., Marcu, D.: Statistical phrase-based translation. In: Proceedings of the Human Language Technology and North American Association for Computational Linguistics Conference (HLT/NAACL), Edomonton, Canada (May 27-June 1 2003)

[11] Och, F.: Statistical machine translation live (April 2006)

[12] Och, F.J., Tillmann, C., Ney, H.: Improved alignment models for statistical machine translation. In: Proc. of the Conference on Empirical Methods in Natural Language Processing and Very Large Corpora, University of Maryland, College Park, MD (June 1999) 20–28

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

[14] Ahlberg, M., Enache, R.: Combining language resources into a grammar-driven swedish parser. In Chair), N.C.C., Choukri, K., Declerck, T., Doğan, M.U., Maegaard, B., Mariani, J., Odijk, J., Piperidis, S., eds.: Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12), Istanbul, Turkey, European Language Resources Association (ELRA) (May 2012)

[15] Borin, L., Forsberg, M., Lönngren, L.: Saldo 1.0 (svenskt associationslexikon version 2). (2008)

[16] Ahlberg, M.: Towards a wide-coverage grammar for swedish using GF (2012)

[17] Dannélls, D., Ranta, A., Enache, R.: Multilingual grammar for museum object descriptions (March 2011)

[18] Dannélls, D., Damova, M., Enache, R., Chechev, M.: A framework for improved access to museum databases in the semantic web. In: RECENT ADVANCES IN NATURAL LANGUAGE PROCESSING. Language Technologies for Digital Humanities and Cultural Heritage, Hissar, Bulgaria (September 2011)

[19] Dannélls, D., Enache, R., Damova, M., Chechev, M.: Multilingual online generation from semantic web ontologies. In: www2012. EU projects track, Lyon, France (April 2012)

[20] Dannélls, D., Ranta, A., Enache, R., Damova, M., Mateva, M.: Multilingual access to cultural heritage content on the semantic web. In: Language Technology for Cultural Heritage, Social Sciences, and Humanities Workshop (LaTeCH). (2013)

[21] Damova, M., Dannélls, D., Mateva, M., Enache, R., Ranta, A.: Natural language interaction with semantic web knowledge bases and lod. In: Towards multilingual Semantic Web. Springer, Berlin (2013)

[22] Niles, I., Pease, A.: Towards a standard upper ontology. In: FOIS '01: Proceedings of the international conference on Formal Ontology in Information Systems, New York, NY, USA, ACM (2001) 2–9

[23] Dannélls, D.: Ontology and corpus study of the cultural heritage domain (September 2011)

[24] Dannélls, D.: Multilingual text generation from structured formal representations. PhD thesis, University of Gothenburg, Sweden (2013)

[25] Dannélls, D., Damova, M.: Reason-able view of linked data for cultural heritage. In: Advances in Intelligent and Soft Computing / The Third International Conference on Software, Services Semantic Technologies (S3T). Volume 101. (2011) 17–24

[26] Dannélls, D., Ranta, A., Enache, R., Damova, M., Mateva, M.: Translation and retrieval system for museum object descriptions (March 2013)

[27] España-Bonet, C., Enache, R., Slaski, A., Ranta, A., Màrquez, L., Gonzàlez, M.: Patent translation within the MOLTO project. In: Proceedings of the 4th Workshop on Patent Translation, MT Summit XIII, Xiamen, China (September 2011) 70–78

[28] España-Bonet, C., Enache, R., Angelov, K., Virk, S., Galgóczy, E., Gonzàlez, M., Ranta, A., Màrquez, L.: Wp5 final report: Statistical and robust machine translation (April 2013)

[29] Virk, S.M., Prasad, K.V.S.: Developing an interlingual translation lexicon using wordnets and grammatical framework. In: NoDaLiDa 2013. (2013)

[30] Tsuruoka, Y., Tateishi, Y., Kim, J., Ohta, T., McNaught, J., Ananiadou, S., Tsujii, J.: Developing a robust part-of-speech tagger for biomedical text. In Bozanis, P., Houstis, E.N., e., eds.: Advances in Informatics. Volume 3746. Springer Berlin Heidelberg (2005) 382–392

[31] Enache, R., España-Bonet, C., Ranta, A., Màrquez, L.: A hybrid system for patent translation. In: Proceedings of the 16th Annual Conference of the European Association for Machine Translation (EAMT12), Trento, Italy (May 2012) 269–276

[32] Détrez, G., Ranta, A.: Smart paradigms and the predictability and complexity of inflectional morphology. In: Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics. EACL '12, Stroudsburg, PA, USA, Association for Computational Linguistics (2012) 645–653

[33] Melamed, I.D., Green, R., Turian, J.P.: Precision and Recall of Machine Translation. In: Proceedings of the Joint Conference on Human Language Technology and the North American Chapter of the Association for Computational Linguistics (HLT-NAACL). (2003)

Chapter 2

Creating Language Resources

1 An Open-Source Computational Grammar for Romanian

Ramona Enache   Aarne Ranta   Krasimir Angelov

Abstract: We describe the implementation of a computational grammar for Romanian as a resource grammar in the GF project (Grammatical Framework). Resource grammars are the basic constituents of the GF library. They consist of morphological and syntactical modules which implement a common abstract syntax, also describing the basic features of a language. The present paper explores the main features of the Romanian grammar, along with the way they fit into the framework that GF provides. We also compare the implementation for Romanian with related resource grammars that exist already in the library. The current resource grammar allows generation and parsing of natural language and can be used in multilingual translations and other GF-related applications. Covering a wide range of specific morphological and syntactical features of the Romanian language, this GF resource grammar is the most comprehensive open-source grammar existing so far for Romanian.

1. Preliminaries

GF[1] [1] is a grammar formalism, which uses type theory to express the semantics of natural languages, for multilingual grammar applications. The GF resource grammars [2] are the basic constituents of the GF library, on top of which applications are built. Notable applications that use GF are the verification tool KeY, for the generation of natural language from the formal language OCL, the dialogue system research project TALK and the educational project WebALT, for generating natural language for mathematical exercises in different languages, and performing multilingual translations.

The two main operations that are regularly performed with resource grammars are the generation of natural language, based on a term in the abstract syntax (linearization) and parsing. Multilingual translation is achieved as a combination of these two processes.

A GF resource grammar basically consists of the abstract syntax, which is a set of rules common to all grammars, and provides the structure of the grammar, and the concrete syntax, which implements the elements of the abstract syntax in the given language, considering its specific features. The abstract syntax provides consistency for the resource library, also ensuring grammatically correct multilingual translations. Resource grammars are general-purpose, as they capture the basic traits of the language. Domain-specific applications use a more restricted domain ontology. In this case, there is more emphasis on the semantical aspect, than in the case of general-purpose grammars. In both cases, only syntactically correct constructions can be generated and parsed.

So far the resource library contains 15 languages : English, French, Italian, Spanish, Catalan, Swedish, Norwegian, Danish, Finnish, Russian, Bulgarian, German, In-

[1] http://www.grammaticalframework.org

terlingua (an artificial language), Polish and Romanian. The last two languages were added in 2009. Considering the Romance languages (French, Italian, Spanish and Catalan) and the Scandinavian ones (Swedish, Norwegian and Danish), as the languages from the same family shared many similarities, they were each implemented as families in the resource library. In this way, in the Romance and Scandinavian module, all the similar features are grouped together, along with an interface that declares the differences among the languages. Regarding syntactical features, members of the same family share more than 75% of the code, through the implementation of the family module.

Although Romanian is a member of the Romance family, it was implemented independently, due to significant differences between it and the existing Romance languages in the GF resource library.

2. Main Categories

Each resource grammar features a complete set of paradigms for the inflectional morphology of the main categories, namely nouns, adjectives, verbs, numerals and pronouns.

In the abstract syntax, lexical entries are represented as nullary functions (constants). They are given linearizations in the concrete syntax, typically of tables with all the inflection forms. For example: `fun airplane_N : N` from the abstract syntax is linearized in the Romanian resource grammar as `lin airplane_N = mkN "avion"` where the function `mkN` generates all the 12 flexion forms needed for a noun, as well as its the gender.

Special categories are the relational nouns, adjectives and verbs, where we specify the case of the object, and the preposition that binds it with the relational category. For example: `fun forget_V2 : V2` will be linearized as `forget_V2 = dirV2 (v_besch18 "uita")` where `v_besch18` indic- ates the group of conjugation for the given verb, according to [3]; the name is a reference to Bescherelle, which is the resource used for implementing verb conjugations for most languages in the Romance family. The function `dirV2` indicates that the verb is transitive, and the corresponding object will in the Accusative cases, with no binding preposition (direct object).

2.1. Nouns

Romanian nouns (N) inflect in case, number and species (definite or indefinite form). The definite article is enclitical, while the indefinite article is proclitical. In the other Romance languages, both the definite and indefinite articles are proclitical. For example:

om → *om***ul** → **un** *om*

man → the man → a man

There are 5 cases: Nominative, Accusative, Dative, Genitive and Vocative, but due to syncretism between Nominative-Accusative and Genitive-Dative, nouns have at most 3 different forms for case inflexion. Other Romance languages have just one inflectional case; case distinctions are expressed by prepositions. For example in French *de* is used for Genitive, and *a* for Dative.

While the other Romance languages have two genders, Romanian has three: Masculine, Feminine, and additionally Neuter. However, the Romanian Neuter has been the center of some linguistic disputes, as it behaves like Masculine for Singular and as Feminine for Plural, from the agreement point of view. This feature allows us to consider only the basic two genders in the syntactic part of the grammar, when reasoning about agreement between nouns and adjectives and noun phrases and verbs.

Another distinguishing feature of Romanian is the Animacy feature, which plays an important role in syntax, particularly for clitic doubling. Inanimate nouns do not have a special form for Vocative. However, compared to the gender which is inherent, animacy can be changed according to use, most frequently from Inanimate to Animate. The features of nouns also apply to adjectives.

In the Romanian resource grammar, the noun is represented as

```
N = {s: Number => Species => ACase => Str;
     g: NGender; a: Animacy};
```

where

```
NGender = NMasc | NFem | Nneut;
Species = Def | Indef;
ACase = ANomAcc | AGenDat | Avoc;
Animacy = Animate | Inanimate;
```

The syntax of parameters in GF follows the model of declaring an algebraic datatype in functional languages, where the elements of the disjunction are constructors of the type. The representation of the noun is a record with three fields, where the `s` field is a multidimensional table storing the 12 forms of the noun. Each of the parameters separated by => defines a new dimension of the table. A function would, hence, need 12 strings, along with a gender and an animacy attribute for a complete representation of a noun.

However, we provide special functions, named smart paradigms, that build the complete representation using at most 3 parameters. These functions can infer the animacy attribute, gender, and declension forms of a noun. The most common ones are the functions that use the Singular and Plural Nominative Indefinite forms of a noun, but other combinations of forms are also considered.

Since for the Vocative case there are no well-established rules, we provide a function that sets this field to a particular value, in case it cannot be inferred by the default rules. Regarding the Animacy feature, since by default nouns are assumed to be inanimate, we provide a function for this case, too.

The gender can be automatically inferred from the last letter of the word with a precision of 77%, on the 186 nouns in the GF Lexicon. The great majority of nouns ending in *-ă*,*-e* or *-a* for the singular form have feminine gender[4]. It is considerably harder to distinguish between masculine and neuter nouns if we have just the singular form, but [4], offers some patterns that characterize masculine words, which are statistically rarer than the neuter ones.

In case the smart paradigm takes both the singular and the plural form as arguments, it can normally differentiate masculine from neuter, as the plural form of neuter nouns ends in *-e* or *-uri*, while masculine nouns always have plural forms ending with *-i*.

For the implementation of a Noun Phrase there are more specific details of the language to take into account.

```
NP= {s: NCase => {comp: Str; clit: Clitics => Str};
     a: Agr; indForm: Str; nForm: NForm;
     isPronoun: Bool};
```

where `NForm = HasClit | HasRef Bool`.

Because pronouns do not have case syncretism, the 5 cases need to be represented separately (`NCase` parameter). The agreement consists of number, gender (Feminine or Masculine) and person.

The parameter `NForm` indicates whether the noun phrase is in referential form and develops clitic doubling (`HasClit`), or, in the absence of clitic doubling, if it is in referential form or not (`HasRef True` or `HasRef False`). We mention that clitic doubling implies referential form, while the reverse does not hold. Nouns in referential form need to be preceded by the preposition *pe* when they act as Direct Objects in Accusative. Although the use of noun phrases in referential form and clitic doubling is very context-dependent in some cases, and subject to discussions in others, we chose the approach suggested in [5]. So, for referential form and clitic doubling, we considered pronouns, animate proper nouns and animate nouns determined by adjectives or possessive pronouns.

Regarding fields from the representation of `NP` :

- `nForm` indicates if the noun phrase needs to be doubled by a clitic in the situations when this phenomenon occurs.
- `isPronoun` is relevant for the clitic doubling situations, because the basic form of the pronoun will be ignored, and the noun phrase will just be represented by the clitic. Situations where clitic doubling also occurs for pronouns are possible, but less common. They are not handled in the resource grammar, since they are meaning dependent, and their semantical role is to emphasize the pronoun.
- `indForm` is used to cover another distinguishing feature in Romanian, which is the usage of the definite/indefinite form of the noun depending on context [5]. Some Accusative prepositions(like *la* - to, *de pe* - on/from) require the indefinite form of a noun phrase, in case it consists of a noun, which is not followed by an adjective or a determiner. For dealing with this case `indForm` stores the suitable form of the noun, to be used if preceded by such a preposition. For example:

  de pe deal - on the hill (Definite form conflict because of preposition *de pe*)
  de pe un deal - on a hill (Indefinite form for noun)
  *de pe deal*ul *mare* - on the big hill (Definite form for noun + adjective)

The intermediate category between nouns (N) and noun phrases (NP), sometimes called “N bar”, is in the GF resource grammar library called common noun, CN. It consists of a noun, possibly with adjectives, adverbs, relative clauses, appositional attributes, and complements for relational nouns.

Noun phrases can be further formed from common nouns followed by determiners, which give the number of the noun phrase and also select the definite/indefinite form of the common noun.

Proper nouns (PN) have different inflection forms and behavior towards clitic doubling, depending on their animacy. The representation of proper nouns is

```
PN = {s: NCase => Str ; g: Gender ;
        n: Number; a: Animacy};
```

Proper nouns thus inflect for case and have inherent gender, number, and animacy. Smart paradigms for proper nouns can infer these properties, setting animacy to the animate by default.

2.2. Adjectives

Adjectives (A) are represented in the resource grammar as

```
A = {s: AForm => Str};
```

where `AForm = AF Gender Number Species ACase | AA`.

Considering the agreement of adjectives with nouns, we just consider the two genders Feminine and Masculine. For neuter nouns, we choose the Masculine or Feminine form, depending on the number, on syntactical level. The constructor `AF` builds the representation table of an adjective, consisting of 24 forms, while `AA` maps an adjective to its corresponding adverb, which in most cases has the same form as the adjective for Masculine Singular. The complete representation of an adjective thus consists of 25 forms, but smart paradigms need at most 5 forms to infer them all. Adverbs (Adv) are inflectionally invariant in Romanian, as in most languages in the resource library.

One of the main difficulties of giving complete inflection rules for adjectives was the presence of phonetical mutations. They are not predictable from the lexical structure, being rather dependent on the etymology and age of the word. Neological words do not usually develop phonetical mutations. Compared to nouns, adjectives need more forms of the word, so it is important to be able to determine the effects of phonetical mutations in a systematic way.

When building the forms for all genders and numbers of the adjective, two main mutations can occur: o$\rightarrow$oa (masculine singular $\rightarrow$ feminine singular and plural) and e$\rightarrow$ea (masculine singular $\rightarrow$ feminine singular). For example, for o$\rightarrow$oa : *frumos*(Masc Sg), *frumoasă*(Fem Sg), *frumoase*(Fem Pl, "beautiful") and for e$\rightarrow$ea : *drept*(Masc Sg), *dreaptă*(Fem Sg, "right"). These changes affect the second or third last letter in the stem.

The default behavior of the adjectives does not feature the phonetical mutations, for which special functions are provided. In the given lexicon, 80% of the 54 adjectives have default behavior. Also, for 60% of them, just the Masculine Singular Indefinite Nominative form is needed in order to build the whole representation table, using the provided declension rules.

The degrees of comparison are formed on syntactical level, as they do not change the basic form of the adjective.

2.3. Verbs

The category of verbs (V) is by far the most complex one from the Romanian resource grammar. On morphological level the table of a verb is defined as:

```
VForm = Inf
      | Indi Temps Number Person
      | Subjo Number Person
      | Imper Number
      | Ger
      | PPast Gender Number Species ACase;
```

where `Temps = Presn | Imperf | PSimple | PPerfect` represents the tenses for Indicative and Subjunctive that cannot be formed analytically on syntactical level. The past participle behaves like an adjective, as in the other Romance languages.

The representation of a verb on morphological level consists of 62 forms :

- Present, Imperfect, Perfect Simple and Past Perfect: 6 forms for each

- Infinitive: 1 form
- Conjunctive: 1 form, corresponding to the third person singular, as the other forms are identical to the present ones, except for the irregular verb *a fi* (to be) which will be treated separately.
- Imperative: 1 form, corresponding to the 2nd person singular, as the 2nd person plural has the same form as for present.
- Past Participle: 24 forms as for ordinary adjectives.
- Gerund: 1 form

The 6 forms required for the first four tenses are motivated by the fact that verbs have different forms for the Cartesian Product of the 3 persons (1, 2 and 3) and 2 numbers (singular and plural).

There are 4 conjugation groups, based on the last 1–2 letters:

1. verbs ending in *-a* (which do not belong to the 2^{nd} group) are in the 1^{st} group (example *a lucra* - to work)
2. verbs ending in *ea* (where *e* and *a* belong to the same syllable and are not preceded by *h*) belong to the 2^{nd} group (example *a părea* - to seem)
3. verbs ending in *e* belong to the 3^{rd} group (example *a zice* - to say)
4. verbs ending in *i* or *î* belong to the 4^{th} group (example *a iubi* - to love, *a hotărî* - to decide)

Each of these groups is divided into 4–14 subgroups, which use different affixes to form tenses and moods. Most of the verbs from the Romanian vocabulary belong to the 1^{st} and 4^{th} group. Most of the irregular verbs belong to the 2^{nd} group, while the verbs from the 3^{rd} group are most likely to develop phonetical mutations.

We have implemented the most complete taxonomy of Romanian verbs freely available so far [3], which consists of 140 groups of verbs, and also built a smart paradigm that distinguishes the most frequent 10 groups.

The behavior of a verb cannot be inferred from its lexical structure, as it depends on its etymology. For example, *a ara* ("to plough") belongs to Group 1, subgroup 6, while *a nara* ("to narrate"), belongs to Group 1, subgroup 1. Although they look very similar, the first is of Latin origin, while the other is a neological word imported from French.

An interesting feature of Romanian is the absence of auxiliary verbs for building composite tenses. Romance languages require the verb "to have" / "to be" for building the past form (French : *j'ai dormi* "I have slept", *je suis parti* "I have left"). In Romanian, some particles are used for this purpose. They are the same for all the verbs, and they cannot be used independently as verbs. For example, for the past form *am, ai, a, am, aţi, au* that originate in the verb "to have" *am, ai, a, avem, aveţi, au*, but the two are not identical.

Before proceeding with the structure of a verb phrase (VP), a discussion on clitics in Romanian is needed. There are 4 types of clitics that can follow a verb:

- Accusative (direct object)
- Dative (indirect object without preposition)

- Accusative (reflexive verbs)
- Dative (reflexive verbs)

At most two clitics in different cases out of the four can occur in a verb phrase. The representation of clitics in the Romanian resource grammar is: `Clitics = Normal | Composite | Short | Imperative` where each of the parameters represents a different instance of the clitic, as follows:

- `Normal`: the form that the clitic takes when following a verb in a tense/mood that is not composed with an auxiliary beginning with a vowel (Example : *Eu* **te** *întreb* "I ask **you**")
- `Short`: the form corresponding to composed tenses and moods(Example : *Eu* **te**-*am intrebat* "I asked **you**")
- `Composite`: the form that the clitic takes when combined with another clitic, following a verb in a tense that is not composed (Example: **Ţ-l** prezint "I present **him to you**")
- `Imperative`: the form of the clitic that is used for the Imperative form of the verb. (Example: *Intreaba*-**ma! "Ask me!"**)

As shown in [6], the order of the clitics is always Dative–Reflexive–Accusative, and the Reflexive clitic, when present, acts as a Dative or Accusative, according to its case. The `Imperative` clitics are always placed after the verb in Imperative mood, while in the other cases they are placed before the verb in the given order, with the only exception of the `Short` clitic for 3rd Person Singular Feminine, which always occurs after the verb.

When combining two clitics for a non-composite tense/mood, both of them are used in their `Composite` form. For a composite tense, the first one is used with the `Composite` form, while the other one is used with `Short` form. If the second clitic is the 3^{rd} Person Singular Feminine, then the first clitic is used with the `Short` form also. In the current implementation, an extra field is used in order to count the number of clitics in a verb phrase. However, in case that two clitics occur, we need to know the case of the reflexive clitic, in order to use it with the right form for a composite tense/mood.

For efficiency reasons, the clitics are stored in the structure of the noun phrases and are transferred to verb phrases in the complementation process.

Having these preliminaries, we can proceed with the representation of the verb phrase. It results from combining transitive verbs with complements, or from intransitive verbs directly.

```
VP = {s: VForm => Str; isRefl: Agr => RAgr;
      nrClit: VClit; pReflClit: Clitics;
      isFemSg: Bool; neg: Polarity => Str;
      clAcc: RAgr; clDat: RAgr;
      comp : Agr => Str; ext : Polarity => Str};
```

where

- `s` stores the forms of the verb which were built on the morphological level.
- `isRefl` corresponds to the reflexive clitics of the verb phrase.

- `nrClit` counts the number of clitics, while `pReflClit` keeps track of the proper form of the reflexive clitic, when combined with another clitic for a composite tense/mood.

- `clAcc` and `clDat` store the clitics for the Accusative and Dative case.

- `isFemSg` keeps track of whether the verb phrase has an 3^{rd} Person Singular Feminine Accusative clitic.

- `neg` is used to express the polarity.

- `comp` stores the objects of the verb phrase, while the `ext` field stores the secondary phrases, introduced by the verb phrase.

The current implementation of clitics uses the above-mentioned structures and parameters for efficiency reasons, and may look artificial, but the problem of clitics is complex in any language, and requires solutions that are both expressive and efficient.

On syntactical level, a distinguishing feature of Romanian is the lack of infinitives, and the use of verbs in Subjunctive Present instead. For this, case agreement is needed, and the current implementation makes the agreement between the verb and the subject of the phrase or the direct object, depending on the grammatical context. For example:

> **Eu** *vreau să merg* "**I** want to go"
> *Eu* **o** *rog să cumpere* "I ask **her** to buy"

In the first case, the verb agrees with the subject, while in the second case, it agrees with the object.

2.4. Numerals

Numerals in Romanian follow the decimal system, as all the other Romance languages. The cardinals composed with the digits 1 or 2 have different forms for Masculine and Feminine. The ordinals inflect in gender and case, but do not normally have forms for plural. For numerals between 11 and 19, alternative formal and informal forms exist, and the grammar generates both of them. The formal form is however used as default.

A distinguishing feature of numerals is the taxonomy of size: `Size = sg | less20 | pl` . There is a difference between numerals from 2 to 19 and numerals which are greater than 20 on syntactical level: an extra preposition is added when combining a numeral greater than 20 with a noun phrase. For example:

> *zece oameni* "ten people"
> *treizeci* **de** *oameni* "thirty people"

2.5. Sentences

Regarding the formation of clauses and sentences, Romanian is very similar to the other Romance languages. Its structure is SVO where the predicate agrees with the subject in number and person (gender for passive voice or predicates formed by copula + adjective).

The inverse topicalization VOS is used for interrogative sentences introduced by an interrogative pronoun, and for relative clauses. For example:

Ion vede pe cineva “John sees somebody”
Pe cine vede Ion? “Who does John see?”
Casa pe care o vede Ion “The house that John sees”.

The interrogative pronoun *cine*(who), as it is animated, requires the preposition *pe*, when it acts as a direct object, but it does not develop clitic doubling. On the other hand, the noun *casa* (“house”), although not animated, is doubled by the corresponding clitic (*o*) in the relative clause that determines it.

3. Evaluation

The Romanian resource grammar was added to the GF library in September 2009, after almost 4 months of work. It consists of 20 modules that cover morphological and syntactical features of Romanian, which are written in the GF language. The size of the code is 5892 lines, which is above the average of the GF library [2].

Resource grammars can be embedded in programming languages like Haskell and Java. This is achieved by compiling the resource grammar to PGF [9], a portable grammar format, which will imported and processed by the host language. The PGF form of a grammar is also a measure of its complexity, as it reflects the number of rules that the grammar uses and the way they combine. For example, the first implementation of clitics in Romanian, which was the intuitive approach, that just kept the clitics and a boolean parameter, keeping track of whether a given clitic is present or not, made the resource grammar was so complex, that the PGF file could not be generated by the GF compiler, as the number of rules was too big. The current implementation of clitics reduced the number of rules 200 times for the verb category, and 4369 times for the complementation function. This made possible the generation of a PGF file for Romanian, which can be used for parsing, multilingual translations and other related applications. Because of the complexity of the morphology and of the clitics, the Romanian resource grammar has one of the highest number of rules in the library.

There is a trade-off between the expressive power of a resource grammar and its efficiency. Our approach covers, as we showed, many specific features of Romanian, but there are still constructions that the current grammar does not cover. One of them is the presence of clitic doubling in a relative phrase that contains a nested verb phrase sequence. For example: *Maşina pe care mă roagă ei să* **o** *cumpăr* (“The car that they beg me to buy”), where the clitic refers to *maşina* (“the car”), is generated as *maşina pe care mă roagă ei să cumpăr*. This can be understood to have the same meaning, but it is not correct in standard Romanian. The solution to this problem would require an additional field for verb phrases. This would bring about an increase in the number of rules for verbs and functions that involve verbs that would make it impossible, in the current implementation of PGF, to generate the PGF format file for Romanian. We prefered to make the grammar as expressive as possible, in the current context of the GF compiler and resources, but still keeping it reasonably efficient, so that it can be used in GF-related applications.

4. Future Work

An obvious direction for future work is improving the efficiency of the grammar, making it possible to add features that are not currently covered because of complexity issues.

A big step towards a more expressive grammar would be adding a bigger lexicon, perhaps by import from other open source projects.

Another main direction is to derive an application grammar for Romanian for the projects that use GF, like WebALT, TALK or KeY.

5. Related projects

The number of open-source projects that attempt to give a formal characterization of the Romanian language is relatively small, and they deal mostly with the morphological features of the language.

Roric-Ling[2], describes paradigms for inflectional morphology of nouns, adjectives and verbs. The rules cover a small lexicon (almost 100 entries), but there are many other cases of inflection which are not treated. For verb conjugation, around 30 forms of the verb are needed as input. Our approach has a wider coverage of the morphology, also featuring smart paradigms, which require considerably smaller inputs.

Another significant project that deals with Romanian morphology is the spell checker from Open Office[3]. It features a comparable set of rules for inflection of noun and adjectives, and a large database.

The EGLU project [8] features the most comprehensive implementation of the Romanian morphology, a large database, but it does not have such a wide coverage for the syntax part, and, to our knowledge, no treatment of clitics. It also has the possibility of performing automated POS tagging and morphological analysis.

Liviu Ciortuz described and implemented a HPSG kernel for Romanian in his PhD thesis [9], elaborating on NPs, VPs and some aspects about clitics. Our work features more aspects of the grammar and is a part of a large multilingual framework.

The LinGO Matrix [10] and Pargram [11] projects, are similar to the GF project and they both feature a computational grammar for Romanian, but they are still under construction, and were not available for a more detailed comparison.

Regarding the theoretical study of Romanian clitics, we mention the work of P. Monachesi [12].

Other computational linguistic resources for Romanian are related to the areas like machine-learning, aquisition of corpora, POS taggers and lemmatisers, word sense disambiguators and others [13].

6. Conclusions

The current resource grammar integrates Romanian in the GF setting, expressing the main features of the language. However it is not complete, as it cannot parse arbitrary sentences, or generate all the possible constructions. The morphology is complete, in the sense that it covers the main categories and their possible declensions/conjugations, and can always be applied to a bigger lexicon, or used in application grammars for any new domain.

Romanian was not integrated in the Romance module, because of some significant differences from the other languages in the family. Some of these are the enclitical definite article, different forms of nouns and adjectives for case triggered declension, and the animacy hierarchy for nouns. Another key feature of the grammar is the problem

[2]http://phobos.cs.unibuc.ro/roric/morpho/demo.html

[3]http://extensions.services.openoffice.org/node/1392

of clitics and clitic doubling, which is considerably different from the languages that were already present in the GF resource library.

The Romanian resource grammar in GF provides substantial coverage of both morphological and syntactical aspects of the language, and is so far the most comprehensive computational grammar for Romanian.

Bibliography

[1] Ranta, A.: Grammatical Framework: A Type-Theoretical Grammar Formalism. The Journal of Functional Programming **14(2)** (2004) 145–189

[2] Ranta, A.: The GF Resource Grammar Library. Linguistic Issues in Language Technology **2** (2009)

[3] Barbu, A.M.: Conjugarea Verbelor Româneşti. Editura Coresi, Bucureşti (2007)

[4] Perkowski, J.L., Vrabie, E.: Covert Semantic and Morphophonemic Categories in the Romanian Gender System. Slavic and East European Journal **30** (1986)

[5] Chiriacescu, S., von Heusinger, K.: Pe-marking and Referential Persistence in Romanian. In: SinSpeC - Working Papers of the SFB 732 "Incremental Specification in Context". (2009)

[6] Klein, U.: The syntax of Romanian clitic pronouns (2007) Manuscript, University of Bielefeld.

[7] Angelov, K., Bringert, B., Ranta, A.: PGF: A Portable Run-Time Format for Type-Theoretical Grammars. Journal of Logic, Language and Information (2009) To appear.

[8] Tufiş, D., Barbu, A.M.: A Reversible and Reusable Morpho-Lexical Description of Romanian. In Tufiş, D., Andersen, P., eds.: Recent Advances in Romanian Language Technology. Editura Academiei Române, Bucureşti (1997) 83–93

[9] Ciortuz, L.V.: DF—A Feature Constraint Concurrent System with application to Natural Language Processing. PhD thesis, University of Lille (1996)

[10] Bender, E.M., Flickinger, D., Oepen, S.: The grammar matrix: an open-source starter-kit for the rapid development of cross-linguistically consistent broad-coverage precision grammars. In: COLING-02 on Grammar engineering and evaluation, Morristown, NJ, USA, Association for Computational Linguistics (2002) 1–7

[11] Butt, M., Dyvik, H., King, T.H., Masuichi, H., Rohrer, C.: The parallel grammar project. In: COLING-02 on Grammar engineering and evaluation, Morristown, NJ, USA, Association for Computational Linguistics (2002) 1–7

[12] Monachesi, P.: Clitic placement in the romanian verbal complex. In: Clitics in Phonology, Morphology and Syntax, John Benjamins (2000)

[13] Cristea, D., Forascu, C.: Linguistic resources and technologies for romanian language. The Computer Science Journal of Moldova **14**(1) (2006) 34–73

2 A Type-Theoretical Wide-Coverage Computational Grammar for Swedish

Malin Ahlberg Ramona Enache

Abstract: The work describes a wide-coverage computational grammar for Swedish. It is developed using GF (Grammatical Framework), a functional language specialized for grammar programming. We trained and evaluated the grammar by using Talbanken, one of the largest treebanks for Swedish. As a result 65% of the Talbanken trees were translated into the GF format in the training stage and 76% of the noun phrases were parsed during the evaluation. Moreover, we obtained a language model for Swedish which we use for disambiguation.

1. Introduction

Two main approaches have divided the research in computational linguistics. The first and most prominent one is the wide-coverage method, mostly based on statistical methods, which compensates its usually shallow analysis with its coverage. The second one is the opposite - provides a deep analysis, but has a limited coverage. Our work on a computational grammar from Swedish falls into the second category, but aims at getting the best of both worlds by enlarging the coverage in order to fit free text, without affecting the quality of the model.

2. Background

The computational grammar was developed within **GF** (Grammatical Framework) [1]. GF is a dependently-typed grammar formalism, based on Martin-Löf type theory, which is mainly used for multilingual natural language applications.

GF grammars are composed of an abstract syntax, the interlingua of the grammar which describes the semantics on a language-independent level and a number of concrete syntaxes, usually corresponding to natural languages. These describe how the concepts from the abstract syntax are described in the language. By defining such a grammar, one also obtains a parser for the language fragment that is described by a concrete grammar and a natural language generation tool for constructions from the same language fragment. In this way, due to the division of the GF grammars into abstract and concrete, it is possible to achieve semantics-preserving translation between any language pair.

However, because the grammar is strict, we can only deal with constructions that the grammar defines and nothing more. One could classify the limitations of the natural language grammar into two categories — missing lexical items and missing syntactic structures. For this reason, GF was mainly used so far for controlled languages [2], dialogues systems [3] and interactive systems where the user is guided to stay within the bounds of the grammar by a predictive parser, such as a multilingual tourist phrasebook [4]. However, there is recent work on parsing free text — a grammar for patent claims from the biomedical domain [5]. This work highlighted a number of caveats of real-world text, but in the same time gave valuable insights for future work on wide-coverage GF grammars.

The current work aims at overcoming the second limitation of GF grammars — adding support for the most common syntactic constructions that could appear in Swedish texts. For some of these we use dependent types to encode the syntactic phenomena in an elegant way. The lexicon limitation is not a problem, as we use an existing large Swedish lexicon [6], which is imported to GF format from an electronic dictionary and which contains more than 100,000 entries.

Swedish is a North-Germanic language, sharing most of its grammatical structure with Norwegian and Danish. It spoken by approximately nine million people, in Sweden and parts of Finland. Although Swedish syntax is often similar to English, the morphology is richer and the word order more intricate. It is a verb-second language: the second constituent of a declarative main clause must consist of a verb. The normal word order is subject-verb-object, but fronting other constituents (topicalisation) is very common, especially for temporal and locative adverbial phrases. Fronting the finite verb marks questions. Special reflexive pronouns and reflexive possessive pronouns for the third person exist, distinct from the normal third person forms.

For testing and evaluation of the grammar and lexicon, we needed a reliable source for comparison. We have used **Talbanken** [7], a freely available, manually annotated, large-scale treebank. The section used for our work contains more than 6,000 sentences of professionally written Swedish gathered from newspapers, brochures and textbooks. It is also redistributed in an updated version, Talbanken05 [8].

3. Related Work

There exist a number of parsers for Swedish already, such as the data-driven Malt parser [9], also trained on Talbanken, the cascaded finite state parser CassSwe [10], The Swedish Constraint Grammar [11] and the Swedish FDG, which uses the Functional Dependency Grammar [12]. Among computational grammars for Swedish, we mention the Swedish version of the Core Language Engine [13], which provides a comprehensive description of syntax and semantics, as well as a translation to English. Unfortunately, the resource is not available for comparison. Other grammars are BiTSE [14], a Swedish grammar that uses the HPSG format [15] and developed within the LinGO Matrix library [16].

Besides the usages mentioned before, GF is currently the leading technology in the European project MOLTO[4], which aims at developing tools for translating between 15 languages in real-time and with high quality. Previous examples of larger type-theoretical GF grammars that use dependent-types are SUMO-GF [17], a GF representation of SUMO[5], the largest open-source ontology and a natural language generation grammar via Montague semantics [18].

4. Grammar

The work focuses on the syntactical dimension and the following section illustrates the work on describing grammatical constructions in the GF formalism.

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

[5]http://www.ontologyportal.org/

Resource grammar The GF package provides an useful resource library [19], covering the fundamental morphology and syntax of more than 20 languages. There is also a small test lexicon included, containing a few hundred common words. The grammars describe how to construct phrases and sentences and how to decline words. They cover the word order, agreement, tense, basic conjunction, etc. Due to the syntactic similarities between the Scandinavian languages, much of the implementation for Swedish is shared with Norwegian and Danish. The modules that concern the lexical aspects are separate, while 85% of the syntax description is shared. There are about 80 functions, which describe the rules for building phrases and clauses, such as functions for predication and complementation:

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

This is the function types – `PredVP` returns a clause when given its input arguments: a noun phrase and a verb phrase. In addition to the core resource grammars, which is shared with all other languages implemented in the library, there is also an extra module where language specific constructions may be added.

Lexicon As mentioned before, our main lexical resource is SALDO[6] from which a GF lexicon has been extracted [6]. Valency information, which is a key feature to good parsing using GF, is extracted from the lexicon Lexin[7]. This gives us a dictionary with more than 100 000 entries, covering all but 500 words from Talbanken (excluding names, compounds and numerical expressions).

4.1. New Features

Earlier it has been hard to identify missing constructions of the Swedish implementation since there was no large resource available to evaluate it on. Our evaluations are based on Talbanken and when first conducting tests we found much room for improvement. The items listed below are examples of constructions implemented during this work.

The s-passive Swedish has two ways of forming passive verb phrases: the periphrastic passive, formed by using the modal auxiliary verb *bli* (“become") and the s-passive which is formed by adding an *s* to the verb. Passive voice is often used, especially the s-passive. It is however not so common in the other Scandinavian languages, where not all words have passive forms for all tenses. The resource grammar for Scandinavian therefore only implemented the periphrastic passive. During this project, an implementation of the s-passive was added. A ditransitive verb – *ge* (“give") (1a) – gives rise to two passives, (1b) and (1c), both covered by our grammar.

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

b. **Passive use, first place**
Hon erbjöds jobbet
*she offered+**s** the job*
(She was offered the job)

c. **Passive use, second place**
Jobbet erbjöds henne
*the job offered+**s** her*
(The job was offered to her)

[6]http://spraakbanken.gu.se/resurs/saldo
[7]http://spraakbanken.gu.se/lexin/

Impersonal constructions Formal subjects are often used in Swedish.

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

Restrictions on both the verb and the noun phrase are covered by the grammar: the determiner of the noun phrase must be of such type that it requires both the noun and its modifiers to appear in indefinite form. The verb phrase must consist of an intransitive verb or be in passive form.

Formalizing the rules for reflexive pronouns by using dependent types An important area in a Swedish grammar is the treatment of the reflexive pronouns and the reflexive possessive pronouns. The reflexives require an antecedent with which they agree in number and person. They must not be used in subject noun phrases of finite sentences, as shown by the ungrammatical examples (3b) and (4b). Still our grammar should accept the sentences (3a) and (4a) :

(3) a. Sina vantar hade han tappat. (SELF'S gloves, he had lost.)

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

(4) a. Han är längre än sin kompis. (He is taller than SELF'S friend.)

b. *Han och sin kompis leker. (He and SELF'S friend are playing.)

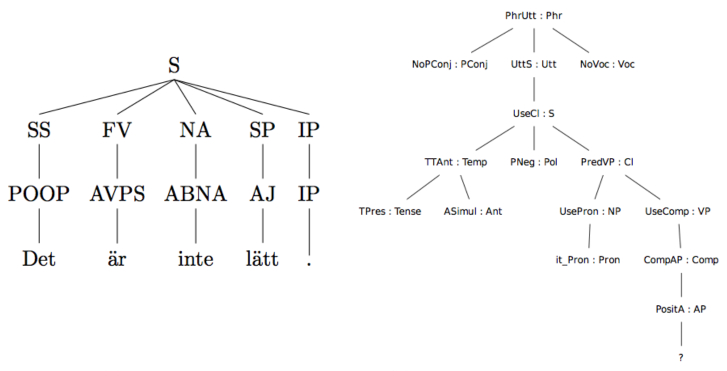
In the standard GF analysis, which is performed bottom-up starting from the POS-tags, information about syntactic roles are given by the functions, not by the categories. For example, the first argument of the function `PredVP` always acts as the subject, but the noun phrase itself does not carry information about its syntactic role. As noun phrases containing reflexive pronouns may be used as ordinary noun phrases – apart from the restrictions stated above – we do not want to differentiate them from other `NP`s on the type level since this would require code duplication. Still, the type system should prevent noun phrases containing reflexive pronouns from being used as subjects. This does not only concern noun phrases, but also adverbial and adjectival phrases (4a).

Our solution introduces the use of dependent types. We make a difference between subjects and objects by letting the type `NP` depend on an argument, which may either be `Subject` or `Object`.

```
PredVP     : NP Subject -> VP -> Cl ;      - predication
ComplSlash :  VPSlash -> NP Object -> VP ; - complementation
PrepNP : (a:  NPType) -> Prep -> NP a -> Adv a ; - adv.  phrase
```

We hence combine the-part-of-speech driven analysis normally performed by GF with a part-of-sentence analysis, where the dependent types give the information we were missing. This is the first example in GF making use of the dependent types for describing the syntox of a natural language. The approach could be extended to describing similar relations in other languages.

Overgeneration Another aspect of the grammar implementation was the avoidance of overgeneration. As the Swedish resource grammar had not been used for larger projects, many examples of overgeneration were present that did not cause problems when working with small lexicons and controlled language. However, when inspecting the test output from Talbanken, unexpected compositions of functions were identified and there after fixed.



(a) Talbanken tree (b) Translation to GF abstract tree

Figure 2.1: The sentence *Det är inte lätt* (“It is not easy").

5. A GF Treebank and Model-Based Disambiguation

Talbanken contains very valuable information about phrase structures and word usage. One part of this project has focused on translating trees from Talbanken to GF by constructing a mapping, which automatically transforms trees in the Talbanken format to GF abstract syntax trees. We hence get a comparison between the two annotations and at the same time we extract a GF treebank. Nodes that fail to be translated are represented by a meta variable, annotated `?`. In figure 2.1, the word *lätt* (“easy") was unknown and therefore represented by a meta variable in the translated tree. Meta variables are also used for connecting sister nodes that can be translated on their own, but not joined into one tree.

Talbanken05 uses three kinds of tags: categories, edge labels and POS-tags. While the POS-tags are reserved for words, the categories give grammatical information: `S`, `NP` and `VP`. The edge labels give the part of sentence: `SS` for subject, `OO` for object, etc.

The mapping could in some cases be performed tag-by-tag, but annotational differences complicated the translation. One example is valency information, which is given implicitly by the complements of a word in Talbanken. If a verb is followed by an object, `OO` containing an `S`, we can conclude that this is a sentence-complement verb. In GF, the valency information is given explicitly for each entry in the lexicon. A sentence-complement verb has the type `VS` and the function `ComplVS` must be used for combining the verb and a sentence into a complete verb phrase.

The translation of Talbanken to GF gives us a GF treebank consisting of more than 6000 sentences. Moreover, the treebank could be mapped to another language from the resource library provided that one defines transfer rules for lexical items and implements the extra syntactic constructions.

The trees give valuable information about which lexical entries and functions are needed to parse the sentence. GF has built-in support for probabilities, and when feeding the output from the mapping to our grammar, we get a model of the language. After the parsing is completed, all trees are ranked, measuring the probability of their constituents. As the data is extracted from manually annotated, real-world sentences, it constitutes a reliable source for disambiguation.

6. Chunk Parsing

The grammar, which is hand written, cannot be expected to cover all possible expressions of the language. Hence, we combine it with statistical means to get an even better coverage. The parser should give as much output as possible even when encountering unknown words and grammatical constructions, idioms and ellipses. Our approach uses of the rich annotation in Talbanken – we perform chunk parsing relying on the tags given in the treebank. Consequentially, whenever a sentence is not recognized by the grammar, it is parsed chunk by chunk. Each chunk is analysed by the grammar-driven parser, and resulting trees are returned. Noun, adverbial and adjectival phrases are considered. Due to the differences in the annotation between Talbanken and GF mentioned is Section 4, verbs are not considered as parsable chunks. Therefore, our focus is on whole sentences and noun phrases. At the first stage, we try to parse the basic sentence structure, and therefore allow the parser to treat complicated chunks as dummy words. This way, we don't lose the high-level sentence analysis given by GF. Subsequently, we try to parse the chunks separately as far as possible.

7. Evaluation

For the extraction of a GF treebank from Talbanken (Section 4), the evaluation measures the numbers of meta variables in each translated tree. Overall we get a number of 65%, and if we limit our input to simple sentences with no more than 10 words, all known to our lexicon, we reach 85%.

The evaluation of the parser is performed by testing it on Talbanken sentences and is still being carried out. So far, our evaluation results cover 600 sentences – about 10% of the treebank. For noun phrases, we cover 76%. By using the chunk parsing described in Section 4, we identify the structure on 65% of the sentences. That is, the parser can identify the verbs and how they relate to the noun phrases, prepositions and particles. However, if a verb is unknown to the grammar, or if it is used with another valency, other prepositions or particles than the lexicon has assigned it, the sentence structure cannot be identified. In that case the parser returns chunks, showing the parse trees of identifiable subphrases.

8. Future Work

Evaluation A more thorough evaluation is still to be done. This should be carried out by measuring the agreement between all parsed chunks and their respective translation from Talbanken. Evaluation should also be done manually. For this, we rely on an expert in Swedish.

Grammar The grammar should cover the most prominent Swedish features. While statistical methods compensate for constructions not covered by the grammar, we still aim for an even broader grammar coverage. As examples of constructions to be added, we mention pronominal object shift, bare indefinites and distinguish between object and subject control verbs.

9. Conclusion

Our work has resulted in a wide-coverage grammar for Swedish which – combined with statistical resources – can be used for parsing open-domain text. The grammar covers a large number of syntactic phenomena and uses an extensive morphological lexicon. To our knowledge, it is the most comprehensive computational grammar for Swedish and the most complex GF grammar to date. Besides parsing, the grammar may well be used for language generation. All parts of the project are open-source and the grammar and other parts are modular and could be reused and further developed.

Bibliography

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

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

[3] Ljunglöf, P., Bringert, B., Cooper, R., Forslund, A.C., Hjelm, D., Jonson, R., Ranta, A.: The talk grammar library: an integration of gf with trindikit. (2005)

[4] Ranta, A., Enache, R., Détrez, G.: Controlled Language for Everyday Use: the MOLTO Phrasebook. Proceeding of the 2nd Workshop on Controlled Natural Languages (CNL 2010) (2011)

[5] España-Bonet, C., Enache, R., Slaski, A., Ranta, A., Marquez, L., Gonzalez, M.: Patent translation within the MOLTO project. In: Proceedings of the 4th Workshop on Patent Translation, MT Summit XIII. (2011) 70–78

[6] Ahlberg, M., Enache, R.: Combining Language Resources Into A Grammar-Driven Swedish Parser. In: Proceedings of LREC. (2012)

[7] Einarsson, J.: Talbankens skriftspråkskonkordans. Lund University: Department of Scandinavian Languages. (1976)

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

[9] Hall, J., Nivre, J., Nilsson, J.: A Hybrid Constituency-Dependency Parser for Swedish. In: In Proceedings of NODALIDA–2007. (2007) 284–287

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

[11] Birn, J.: Swedish Constraint Grammar. Technical report, Lingsoft Inc. (1998)

[12] Tapanainen, P., Järvinen, T.: A non-projective dependency parser. In: In Proceedings of the 5th Conference on Applied Natural Language Processing. (1997)

[13] Gambäck, B.: Processing Swedish Sentences: A Unification-Based Grammar and Some Applications. PhD thesis, The Royal Institute of Technology and Stockholm University, Dept. of Computer and Systems Sciences (1997)

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

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

[16] Bender, E.M., Flickinger, D., Oepen, S.: The Grammar Matrix: An Open-Source Starter-Kit for the Rapid Development of Cross-Linguistically Consistent Broad-Coverage Precision Grammars. In: Proceedings of the Workshop on Grammar Engineering and Evaluation at the 19th International Conference on Computational Linguistics. (2002) 8–14

[17] Enache, R., Angelov, K.: Typeful Ontologies with Direct Multilingual Verbalization. LNCS Post-Proceedings of the Controlled Natural Languages Workshop (CNL 2010) , Marettimo, Italy (2011)

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

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

Chapter 3

Grammars Describing Structured Models

1 Typeful Ontologies with Direct Multilingual Verbalization

Krasimir Angelov   Ramona Enache

Abstract: We have developed a methodology for representation of ontologies in a strictly typed language with dependent types. The methodology is supported by an experiment where we translated SUMO (Suggested Upper-Merged Ontology) to GF (Grammatical Framework). The representation of SUMO in GF preserves the expressivity of the original ontology, adding to this the advantages of a type system and built-in support for natural language generation. SUMO is the largest open-source ontology describing over 10,000 concepts and the relations between them, along with a number of first-order axioms, which are further on used in performing automated reasoning on the ontology. GF is a type-theoretical grammar formalism mainly used for natural language applications. Through the logical framework that it incorporates, GF allows a consistent ontology representation, and thanks to its grammatical features the ontology is directly verbalized in a number of controlled natural languages.

1. Introduction

The constantly growing amount of formal knowledge has brought about the necessity of a coherent and unambiguous representation of ontologies which can further be processed automatically. As a consequence, a number of ontology description languages like KIF [1], OWL [2], CycL [3] and Gellish [4] has emerged. However, the focus in all these languages is on the knowledge representation and consequently, they are mainly descriptive, leaving tasks such as consistency checking or natural language generation to external tools. Moreover, most languages are based on some kind of untyped first-order logic with predicates which occasionally allows higher-order constructions. They aim to maximize the expressivity with the cost of allowing set theoretical paradoxes to be expressed (Section 1). Also, because of the lack of a type system, one can easily extend such ontologies with axioms which are not well-formed. Although type information in these languages is often provided in the form of logical assertions, the validation of correctness is left to a reasoner which may or may not be able to find all problems. Even with a complete and decidable reasoner, if the ontological language has the open-world assumption, a potential problem might be left undiscovered, if it is not stated explicitly that certain classes in the ontology are disjoint. This is a problem when dealing with large coverage ontologies. Or, for example, a predicate could be applied to an argument of the wrong type, or a small change in the signature of a function could lead to the update of all its occurrences. If all these checks are manual, then this is a resource-consuming and error-prone process. In contrast, database systems are equipped with rigid database schemas which ensure that the information is always kept consistent. The programming languages community was also dealing with that from the very beginning of computer science and has developed many different type systems.

We have developed a methodology for encoding of ontologies in strictly typed language based on type theory with dependent types. As a proof-of-concept, an experiment with SUMO [5], the largest open-source ontology available today. The implementation language of choice is GF [6]. The result is a controlled language which could be used to formulate new axioms in SUMO or to render existing axioms in natural language. Further on, we analyzed the difference of expressivity compared to the original ontology and also the other benefits that one can get from encoding an ontology in GF.

SUMO consists of 2 upper-level ontologies (*Merge*, *Mid-level-ontology*) describing general concepts, and 29 domain-specific ontologies for finances (*FinancialOntology*), geographical concepts (*Geography*), and others. The ontology is written in a dialect of KIF (Knowledge Interchange Format [1]), called SUO-KIF, which permits the declaration of concepts in a human-readable form, featuring support for expressing first-order predicate calculus constructions. However, due to the modelling of the hierarchy in SUMO, which treats functions and relations as ordinary concepts, it is possible to express second-order logic constructions in SUO-KIF, such as quantification over functions and relations.

The SUMO ontology has natural language translations for the *Merge* module in 12 languages. The translations are based on a set of string templates, which are combined by concatenation. They are hand-written and cover the ontology partially. However, the templates are not expressive enough to handle various natural language phenomena such as case and gender agreement or phonetic mutations. We will show how these problems were solved by using GF.

GF is a type-theoretical grammar formalism which distinguishes between abstract and concrete syntax. The abstract syntax is a logical framework based on Martin-Löf's type theory [7], in which the application domain can be described in an abstract language-neutral manner. The concrete syntax is a mapping of the abstract syntax into some controlled natural language. Since it is possible to have multiple concrete syntaxes, linked to the same abstract syntax, the abstract syntax acts as a semantic interlingua which allows simultaneous translation into multiple controlled languages.

We consider the abstract syntax of GF as a kind of ontology description language and translate some of the axioms from SUMO to statements in the abstract syntax of a grammar. Other axioms, those related to the natural language generation from SUMO, are used to generate the concrete syntax. The rest of the axioms are just converted to abstract syntax trees and used in the automated theorem prover for reasoning.

The development of a grammar for a new controlled natural language from scratch would involve an ad hoc implementation for low-level linguistic details such as word order, agreement, etc. This is simplified by using the resource grammar library [8] developed in GF. The library provides an abstract syntax for common general-purpose natural language constructions and concrete syntaxes corresponding to 16 languages. The usage of the library ensures that the rendering is always syntactically correct and reduces the development effort for new application grammars. The resource grammar library was used for the generation of the concrete syntax of SUMO.

Another advantage of GF is the portability of the grammars, via PGF [9] – a runtime binary format, which can be used by applications written in Haskell, Java, JavaScript, C and Python - through the GF runtime system. In this way, the GF grammars can be embedded in user applications. GF has been used in various large-scale projects such as the dialogue system research project TALK [10], the educational project WebALT [11], the verification tool KeY [12], and the project in multilingual translation MOLTO [13].

Regarding the automated reasoning and the checking for consistency [14], SUMO was mapped to TPTP-FOF [15], a standard untyped first-order logic language, which is accepted by most theorem provers. There is an annual competition held during the premier conference in automated deduction, CADE[1], which awards prizes for finding inconsistencies in one of the two upper ontologies from SUMO, based on these mappings[2]. A similar translation from SUMO-GF to TPTP is provided. The translated ontology is checked for consistency and is used for making inferences on the abstract syntax trees or natural language, with the aid of an automated theorem prover (Section 1).

SUMO is also associated with a knowledge engineering environment – Sigma [16], which can be used for intelligent browsing of the ontology, optimized natural language generation and automated reasoning [17]. An alternative system with similar capabilities is the KSMSA browser[3]. The web user interface of GF also evolved in the direction of ontology browsing. Since his interface is still under development, we will give an overview of it in Section 1.

From the total number of ontologies that SUMO provides, 17 were translated into GF. These are: *Merge* and *Mid-level-ontology* – the upper ontologies and 12 domain ontologies. The remaining ontologies can also be ported to GF using the same techniques, in a semi-automatic way.

The advantages of representing the SUMO ontology in GF are the possibility to type-check the axioms and the definitions at an early stage and also to generate natural language of a higher syntactical quality. The translation to GF, is also an in-depth analysis of SUMO and the benefits that a type system in general, and GF in particular, could bring to ontology development.

2. The Abstract Syntax of SUMO-GF

The two languages SUO-KIF and GF have been created for different purposes and have evolved in different ways. It is not surprising that the translation of SUMO from SUO-KIF to an abstract syntax in GF is not trivial. Still we will show that the different ontological concepts - from classes and taxonomical relations to complex logical axioms have natural representations in GF.

2.1. The Taxonomy

The most central component of every ontology is the taxonomy of classes, and this is the starting point from where we begin the ontology modelling in GF.

Knowledge representation languages like OWL, KIF and CycL do not set a sharp border between classes and instances. In fact, the classes are just instances of one special class which is the class of all classes. In SUMO the special class is called *Class* and there is a predicate *subclass* which is used to assert the taxonomical relations. For example, the axiom:

$$(subclass\ Human\ Hominid) \tag{3.1}$$

asserts that the class *Human* is a subclass of *Hominid*. Furthermore, there is an axiom stating that everything that is a subclass of *Entity* is also an instance of *Class* and

[1]http://www.cadeinc.org/
[2]http://www.cs.miami.edu/ tptp/Challenges/SUMOChallenge/
[3]http://virtual.cvut.cz/ksmsaWeb/main

viceversa:

$$\begin{aligned}&(<=> (\mathit{instance}\ ?\mathit{CLASS}\ \mathit{Class})\\&\quad (\mathit{subclass}\ ?\mathit{CLASS}\ \mathit{Entity}))\end{aligned}$$

Since the *subclass* relation is transitive and *Entity* is the most general class, from the axiom:

$$(\mathit{subclass}\ \mathit{Class}\ \mathit{SetOrClass})$$

it follows that *Class* is itself an instance of *Class*:

$$(\mathit{instance}\ \mathit{Class}\ \mathit{Class})$$

This kind of cyclic relations were proven to be inconsistent because they lead to different kinds of paradoxes (Section 1). The other two popular languages OWL and CycL are not exceptions and similar examples could be constructed in them as well. This seems to be a common mistake because the first version of Martin-Löf's [18] type theory suffered from the same inconsistency which was first demonstrated with Girard's paradox [19]. The problem was resolved in the later versions of the theory [7] by introducing the concepts of small and big types. In the context of SUMO, this would be translated as a restriction which states that *Class* cannot be an instance of *Class* because it is too big to fit as an instance of itself. The abstract syntax of GF is a logical framework consistent with the modern type theory, so if we want to model ontologies like SUMO in GF we have to resolve the conflict.

GF distinguishes between values and types. Every value belongs to some type but none of the types could be a value as well, so it is not possible for a type to belong to another type. The solution for the cyclic relation in SUMO is to declare that *Class* is a type:

$$\textbf{cat}\ \mathit{Class};$$

Now the classes will be values of type *Class*. For instance:

$$\begin{aligned}\textbf{fun}\ &\mathit{Entity} : \mathit{Class};\\&\mathit{Hominid} : \mathit{Class};\\&\mathit{Human} : \mathit{Class};\end{aligned}$$

Essentially, we cut the class *Class* from the common hierarchy and move it to another level (also known as universe in type theory).

Once we have a way to define classes in the abstract syntax we could also define the taxonomy. In SUMO, the taxonomy is encoded by using the *subclass* predicate. In GF, we can translate *subclass* either as a function or as a type. Since we want to be able to statically check the axioms for well-formedness we choose to represent the predicate as a type:

$$\textbf{cat}\ \mathit{SubClass}\ \mathit{Class}\ \mathit{Class};$$

then the human-hominid relation could be asserted as:

$$\textbf{fun}\ \mathit{Human\_Class} : \mathit{SubClass}\ \mathit{Human}\ \mathit{Hominid}; \tag{3.2}$$

Here, the *SubClass* type is an example of a dependent type. The dependent types are not just simple identifiers, but have in addition indices of some type. In this case, *SubClass* is a type indexed by two values of type *Class*. In the case of *Human_Class* those are *Human* and *Hominid*.

Note that while in the original SUMO axiom (3.1) we had just a logical assertion, in GF we have to assign an unique identifier (*Human_Class*) to it. In type theory this is deeply rooted in Curry—Howard's correspondence, but it is interesting that a similar kind of "labeling" of assertions is now emerging in OWL via the Named Graphs standard [20].

Semantically the *subclass* predicate in SUMO encodes the reflexive transitive closure of the taxonomic relation, while the immediate subclass relation is encoded using the predicate *immediateSubclass*. To take this into account we define one more type:

$$\textbf{cat}\ \mathit{Inherits}\ \mathit{Class}\ \mathit{Class};$$

Strictly speaking the *SubClass* type is the translation of the predicate *immediate-Subclass* and *Inherits* is the translation of *subclass*. However we choose to read simple *subclass* axioms such as (3.1) as assertions for immediate subclassing and thus the conversion tool will generate the *SubClass* type in GF. The reason for this is that this would let us do some reasoning with the ontology by using only the tools that are already available in GF. Our intuition is that this still preserves the principal information from SUMO.

From the atomic *SubClass* axioms we can easily infer the reflexive-transitive closure *Inherits*. All that is needed is to add two inference rules. The inference rules in type theory are nothing else but functions with some specific type signatures:

$$\begin{aligned}\textbf{fun}\ \mathit{inhz} &: (c : \mathit{Class}) \rightarrow \mathit{Inherits}\ c\ c;\\ \mathit{inhs} &: (c_1, c_2, c_3 : \mathit{Class}) \rightarrow \mathit{SubClass}\ c_1\ c_2\\ &\rightarrow \mathit{Inherits}\ c_2\ c_3 \rightarrow \mathit{Inherits}\ c_1\ c_3;\end{aligned}$$

The type of function *inhz* states that every class c inherits itself, i.e. this is the reflexivity axiom. The second function *inhs* expresses the transitivity over *SubClass*, i.e. if c_1 is a subclass of c_2, and c_2 inherits c_3 then c_1 inherits c_3.

The inference rules can be applied using the inference engine built into GF. For example, from the GF shell the user can use the `gt` command to generate an expression of a given type:

```
SUMO> gt -cat="Inherits Human Hominid"
(inhs Human Hominid Hominid Human_Class (inhz Hominid))
```

In type theory the types are seen as logical propositions and the existence of a value of a given type is interpreted as an evidence for the validity of the proposition. The value is also a constructive recipe for building the proof from the axioms in the theory. In Section 1 we will use it to generate explanations in natural language for the proofs.

Some of the classes in SUMO have two or more superclasses. For instance *Human* is both a *CognitiveAgent* and a *Hominid*. In other situations it is necessary to quantify over instances of the union of two or more classes. For that purpose we added two of the primitive operations from description logic – intersection and union of classes:

$$\begin{aligned}\textbf{fun}\ \mathit{both} &: \mathit{Class} \rightarrow \mathit{Class} \rightarrow \mathit{Class}; && \text{– intersection}\\ \mathit{either} &: \mathit{Class} \rightarrow \mathit{Class} \rightarrow \mathit{Class}; && \text{– union}\end{aligned}$$

With the help of these primitives, the full definition of the class *Human* is:

$$\textbf{fun}\ \mathit{Human\_Class} : \mathit{SubClass}\ \mathit{Human}\ (\mathit{both}\ \mathit{CognitiveAgent}\ \mathit{Hominid}); \tag{3.3}$$

The reasoning with these two new primitives can be axiomatized with three new inference rules:

$$\begin{aligned}
\textbf{fun}\ & \mathit{bothL} : (c_1, c_2 : \mathit{Class}) \to \mathit{SubClass}\ (\mathit{both}\ c_1\ c_2)\ c_1;\\
& \mathit{bothR} : (c_1, c_2 : \mathit{Class}) \to \mathit{SubClass}\ (\mathit{both}\ c_1\ c_2)\ c_2;\\
& \mathit{eitherC} : (c_1, c_2, c_3 : \mathit{Class}) \to\\
& \qquad \mathit{SubClass}\ c_1\ c_3 \to \mathit{SubClass}\ c_2\ c_3 \to \mathit{SubClass}\ (\mathit{either}\ c_1\ c_2)\ c_3;
\end{aligned}$$

The first two state that the intersection class of any two classes c_1 and c_2 is a subclass of both c_1 (function *bothL*) and c_2 (function *bothR*). The third function (*eitherC*) states that if two classes c_1 and c_2 are both subclasses of c_3, then their union class is also a subclass of c_3. Now, with the extended definition for *Human* (3.3), the proof that every Human is a kind of Hominid will use the function *bothR*:

```
SUMO> gt -cat="Inherits Human Hominid"
(inhs Human (both CognitiveAgent Hominid) Hominid Human_Class
(inhs (both CognitiveAgent Hominid) Hominid Hominid
(bothR CognitiveAgent Hominid) (inhz Hominid)))
```

At least in some cases, the criterion which distinguishes the members of a given class from the super class is formally specified. In this case the criterion is specified in SUMO as an axiom. In our encoding we found it handy to use an encoding which uses the *KappaFn* function. *KappaFn* is a function in SUMO which takes a logical formula and returns the class of all instances for which the formula is valid. The type of the function in GF is:

$$\textbf{fun}\ \mathit{KappaFn} : (c : \mathit{Class}) \to (\mathit{Var}\ c \to \mathit{Formula}) \to \mathit{Class}; \tag{3.4}$$

It takes as arguments the superclass c and the logical formula and returns the subclass. The type $(\mathit{Var}\ c \to \mathit{Formula})$ indicates that the argument itself is a function which takes a variable of class c and returns a formula. Every instance of c for which the formula is true is also a member of the new subclass. Using *KappaFn* it is trivial to define the class *NegativeRealNumber* as a subclass of *RealNumber*:

$$\begin{aligned}
&\textbf{fun}\ \mathit{NegativeRealNumber} : \mathit{Class};\\
&\textbf{def}\ \mathit{NegativeRealNumber} = \mathit{KappaFn}\ \mathit{RealNumber}\ (\backslash N \to \mathit{lessThan}\ \ldots);
\end{aligned}$$

Again for the inference of the transitive closure to work we need an inference rule:

$$\begin{aligned}
\textbf{fun}\ \mathit{kappa} : (c : \mathit{Class}) \to (p : \mathit{Var}\ c \to \mathit{Formula}) \to&\\
\mathit{SubClass}\ (\mathit{KappaFn}\ c\ p)\ c;&
\end{aligned}$$

which defines the semantics of *KappaFn*, i.e. that the new class is a subclass of the argument of the function.

2.2. Instances

Once we have the taxonomy of the ontology we can proceed with adding some instances. Similarly with the classes we will distinguish between direct instances of a class and generalized instances. The instances will be defined as values of one of the following types:

$$\textbf{cat}\ \mathit{El}\ \mathit{Class};\\ \qquad \mathit{Ind}\ \mathit{Class};$$

The type $\mathit{Ind}\ c$ is assigned to all instances with principal class c, while $\mathit{El}\ c$ is the type of all direct instances of c together with the instances of its subclasses. There is an injection between this two types:

$$\textbf{fun}\ \mathit{el} : (c_1, c_2 : \mathit{Class}) \rightarrow \mathit{Inherits}\ c_1\ c_2 \rightarrow \mathit{Ind}\ c_1 \rightarrow \mathit{El}\ c_2;$$

The function *el* injects an instance with principal class c_1 into the type of the generalized instances of c_2, if there is an evidence that c_1 is a subclass of c_2 (the argument $\mathit{Inherits}\ c_1\ c_2$). For example in the CountriesAndRegions module of SUMO there is an instance for the city of London:

$$\textbf{fun}\ \mathit{LondonUnitedKingdom} : \mathit{Ind}\ \mathit{EuropeanCity};$$

The class *EuropeanCity* is a subclass of *City* so it is possible to do the coercion. The following expression is the injection of *LondonUnitedKingdom* into the generalized instances of *City*:

$$\begin{array}{l} \mathit{el}\ \mathit{EuropeanCity}\ \mathit{City} \\ \quad (\mathit{inhs}\ \mathit{EuropeanCity}\ \mathit{City}\ \mathit{City}\ \mathit{EuropeanCity\_Class}\ (\mathit{inhz}\ \mathit{City})) \\ \quad \mathit{LondonUnitedKingdom} \end{array}$$

2.3. Functions, Predicates and Logical Formulas

In SUMO, all functions and predicates are represented as instances of a descendant of *Relation*, and the expected classes of the arguments and the result are stated as axioms in the ontology. For example the definition of the *AbsoluteValueFn* function is:

$$\begin{array}{l} (\mathit{instance}\ \mathit{AbsoluteValueFn}\ \mathit{UnaryFunction}) \\ (\mathit{domain}\ \mathit{AbsoluteValueFn}\ 1\ \mathit{RealNumber}) \\ (\mathit{range}\ \mathit{AbsoluteValueFn}\ \mathit{NonnegativeRealNumber}) \end{array}$$

Here the predicates *domain* and *range* specify the class of the first argument and the class of the returned value. The class of *AbsoluteValueFn* itself is *UnaryFunction* which encodes the fact that this is a function with only one argument. From this SUMO axioms we generate a type signature in GF:

$$\textbf{fun}\ \mathit{AbsoluteValueFn} : \mathit{El}\ \mathit{RealNumber} \rightarrow \mathit{Ind}\ \mathit{NonnegativeRealNumber};$$

Note that with our implementation we impose the closed world assumption. The argument of *AbsoluteValueFn* is declared of type $\mathit{El}\ \mathit{RealNumber}$, and the only way to construct a value of that type is to combine an instance of some subclass c of *Real-Number* with a proof object of type:

$$\mathit{Inherits}\ c\ \mathit{RealNumber}$$

If this object cannot be constructed from the current state of the knowledge base then the application of *AbsoluteValueFn* is not possible.

The predicates are declared in a way very similar to the functions. The only difference is that while the functions return some instance, the predicates are used to create logical formulas. In the original ontology, there is already a class called *Formula* which represents the class of all well-formed SUO-KIF formulas. In principle the predicates could return *Ind Formula* but there are two reasons for which we choose not to do that. The first reason is that if *Formula* is kept as a class then this would allow quantification over logical formulas which is not supported in first-order logic. The second reason is that when the logical axioms are translated to natural language then *Formula* will correspond syntactically to a sentence while *Ind* corresponds to a noun phrase, and this would make the verbalization of the ontology difficult. Instead we declared *Formula* as a type:

$$\mathbf{cat}\ \mathit{Formula};$$

The last piece that is needed to be able to write logical axioms in GF is to add the standard logical quantifiers and connectives:

$$\begin{aligned}
&\mathbf{cat}\ \mathit{Var}\ \mathit{Class};\\
&\mathbf{fun}\ \mathit{var} : (c_1, c_2 : \mathit{Class}) \to \mathit{Inherits}\ c_1\ c_2 \to \mathit{Var}\ c_1 \to \mathit{El}\ c_2;\\
\\
&\mathbf{fun}\ \mathit{exists} : (c : \mathit{Class}) \to (\mathit{Var}\ c \to \mathit{Formula}) \to \mathit{Formula};\\
&\quad\ \ \mathit{forall} : (c : \mathit{Class}) \to (\mathit{Var}\ c \to \mathit{Formula}) \to \mathit{Formula};\\
&\mathbf{fun}\ \mathit{not} : \mathit{Formula} \to \mathit{Formula};\\
&\quad\ \ \mathit{and}, \mathit{or}, \mathit{impl}, \mathit{equiv} : \mathit{Formula} \to \mathit{Formula} \to \mathit{Formula};
\end{aligned}$$

The only specific thing here is how the variables are introduced by the quantifiers. The first argument of the quantifier (function *exists* or *forall*) is the class over which the function quantifies. The second argument is the formula over which it scopes. The quantified variable itself is a high-order argument of type $\mathit{Var}\ c$. This type plays a role similar to the role of El. While the former denotes some known instance, for Var we neither know the instance, nor its principal class. This is reflected for example in natural language generation where the grammatical gender is deduced from the class of the variable instead of the instance itself. This special treatment of variables allows the generation of more fluent natural language. Still the *var* function allows the coercion from type *Var* to *El*.

With the usage of quantifiers and connectives all axioms from SUMO, which were not already converted to type signatures in GF, can be converted to abstract syntax trees. For example the SUO-KIF formula:

$$\begin{aligned}
&(=>\ (\mathit{instance}\ ?P\ \ \mathit{Wading})\\
&\qquad (\mathit{exists}\ (?W)\ (\mathit{and}\ (\mathit{instance}\ ?W\ \mathit{BodyOfWater})\ (\mathit{located}\ ?P\ ?W))))
\end{aligned}$$

is converted to the following abstract syntax tree in GF:

$$\mathit{forall}\ \ \mathit{Wading}\ (\backslash P \to \mathit{exists}\ \mathit{BodyOfWater}\ (\backslash W \to \ \ \mathit{located}\ (\mathit{var}\ P)\ (\mathit{var}\ \ W)))$$

Note that this is more than just a syntactic conversion because the quantifiers in GF expect explicit class information while in SUMO this is encoded with *instance* predicates.

2.4. Proofs in Natural Language

As it was mentioned in Section 1, the proofs in GF are explicitly represented as abstract syntax trees. Since the abstract syntax trees could also have linearizations in the concrete syntax, it is possible to render the proofs in the same controlled natural language encoding the ontology. For example the following command in the GF shell:

```
SUMO> gt -cat="Inherits Human Primate" | l -lang=SUMOEng
```

will derive a proof for $Inherits\ Human\ Primate$ and will linearize the proof in English. The text contains some HTML tags, so when it is rendered in a web browser it looks like a bullet list:

- *human is a subclass of both cognitive agent and hominid*
- *hominid is a subclass of primate*

The natural language rendering can be used to generate end-user explanations for the inferences in the ontology.

3. Russell's paradox

Russell's paradox [21] was first discovered in naïve set theory. It stems from the assumption that for every logical proposition there is a set of entities which satisfy the proposition. This was shown to be inconsistent with the example of the set of all sets which are not members of themselves. Such a set cannot exist because then it will be simultaneously a member and not a member of itself. The design of SUMO follows naïve set theory and the *KappaFn* function is exactly the way to build sets from propositions. Using the function, the paradox can be expressed as:

$$(instance\ (KappaFn\ "x"\ (not\ (instance\ x\ x)))$$
$$(KappaFn\ "x"\ (not\ (instance\ x\ x))))$$

The reasoning with SUMO is sound only because the *KappaFn* function is not axiomatised and the automated theorem provers cannot make any inferences.

The paradox is principally avoided in the GF translation by first discarding the predicate *instance* and second by making the class *Class* into a type. This results into a completely different signature for KappaFn (3.4) which would make the above statement incorrect even if we still had the predicate *instance*.

4. Verbalization

Apart from the advantages that the GF type system provides, for the natural language generation the benefits of using GF are considerably more substantial. The present work deals with the generation of natural language for the two upper ontologies - *Merge* and *Mid-level-ontology* in 3 languages: English, Romanian and French, as a proof-of-concept for the capabilities of GF to host a controlled language for ontologies.

For English, about 7,000 concepts and relations have been translated to natural language. For Romanian and French, only a small number of examples, that illustrate the advantages of GF over a template-based generation, were built. This is due to the fact that there are no large coverage lexicons for those languages in GF yet.

A typical SUMO template is the predicate `age` expressed in English:

```
(format en age "the &%age of %1 is %n %2")
```

where %n will be replaced with "not" for the negation of the predicate, and with the empty string for the affirmative form. The structure of the templates is rather simple, and works reasonably just for morphologically simple languages, such as English. The templates do not take into account the presence of declension forms for nouns, of the gender agreement with verbs and prepositions or the various moods of a sentence, depending on its usage.

This solution is not compositional and leads to incorrect constructions in languages with a rich morphology such as Romanian. For example the verbalization of "the inverse of the square root of X" in Romanian would require the combination of two templates and would render: *inversa lui rădăcina pătrată a lui X*, which is considerably different from the correct form - *inversa rădăcinii pătrate a lui X*. One reason is that the translation of "square root" should be in Genitive case, whereas the template only has the Nominative one, and in Romanian the two forms are different. The second is the matter of the possessive preposition, which in Romanian needs to agree with its object. The template provides the masculine form as default, but *rădăcina pătrată a lui X* is feminine. For French, although nouns do not have multiple declension forms, there is an agreement in gender and number between nouns and other parts of speech that determines them, which cannot be handled by the SUMO templates. In addition to this, for French there is also the problem of phonetic mutations, such as for the usage of a verb with negative polarity. In case that the verb starts with a vowel, the form of the particles used to express negation changes, and this is a mutation that SUMO doesn't handle, because the templates provide only one value for the particles. It goes without saying that the French and Romanian resource grammars offer solutions for these problems, so that the natural language generation in SUMO-GF is syntactically correct for compositions of patterns also.

Moreover, the feature that shows best the advantage of a typed system in general, and of GF, in particular, over sets of templates is the assignment of a gender to the variables, according to the gender of their type, for languages that have gender agreement [22]. This is a very common feature for Romance and Slavic languages, where there is a gender differentiation. The SUMO templates simply assume that all the variables have masculine gender, while in GF, the wrapper function `var`, that has access to the class of the variable also, would assign a proper gender to the variable. Since variables can only be used after being wrapped with `var`, they will have a correct gender for any usage in a quantified formula. This behaviour shows the importance of separating between variables and instances of a class. If `Var` and `Ind` or `El` would have been unified in the same category, we could not use a wrapper function to change the gender, since we might accidentally change the gender of an ordinary instance.

An example of how the gender variation feature works in the current implementation is the GF axiom:

$$forall\ Animal\ (\backslash A \rightarrow exists\ Animal\ (\backslash B \rightarrow smaller\ (varB)\ (varA)))$$

which would be linearized in French as:

pour chaque animal A il existe un animal B `tel` *que B est plus* `petit` *que A* where animal is of masculine gender in French. For a type of feminine gender, such as house we would have that:

$$\mathit{forall\ House}\ (\backslash A \rightarrow \mathit{exists\ House}\ (\backslash B \rightarrow \mathit{smaller}\ (\mathit{varB})\ (\mathit{varA})))$$

which would be linearized in French as:

pour chaque maison A il existe une maison B `telle` *que B est plus* `petite` *que A*

The axioms are not taken from SUMO, but they are just two examples that illustrate this linguistic feature, and would not probably hold in general, as the set of animals and the set of houses are finite, and hence noetherian.

The examples, although few, show the advantages of GF in developing a set of multilingual aligned syntactically-correct controlled languages for describing ontologies.

Besides axioms, we can also generate natural language for `SubClass`, `Ind` declarations and higher-order functions. For example:

beverage is a subclass of food

blue is an instance of primary color

"x is equal to y" is an equivalence relation

Our work provides natural language generation in English for the two biggest modules *Merge* and *Mid-level-ontology* and two domain specific: *Elements* - featuring chemical substances and *Mondial* - featuring countries and cities of the world. As mentioned before, a total of almost 7 000 objects and 500 relations from SUMO were verbalized. This process is done automatically for objects and semi-automatically for relations, and uses the GF resource grammar.

The automatic process takes advantage of the camel case representation of SUMO concepts. For example, *BodyOfWater* will be rendered as "body of water" and parsed by GF as a noun phrase. Instances are parsed as GF noun phrases, while classes are parsed as GF common nouns, which are similar to noun phrases, only that they have variable number, gender and other morphological features. In this way, the representation of *BodyOfWater* will also contain the plural form "bodies of water", which we can use for generating natural language constructions. For functions and predicates the missing arguments are replaced by some dummy variables and the procedure is semi-automatical, using the original SUMO templates and hand-written verbalizations which are further on parsed as noun phrases for functions and clauses with polarity for predicates. For example, the binary predicate `parent` will be verbalized as "o1 is the parent of o2" and parsed to a GF abstract syntax tree. This method allows generalizations, so the 2 negative forms are "o1 is not the parent of o2" and "o1 isn't the parent of o2" are automatically obtained from this. For the two domain specific ontologies, the information is extracted from the SUMO predicate `name` that gives the English verbalization of the concepts. As a result, our approach renders verbalization of a large number of entries from the ontology, with a high rate of automation, ensuring syntactical correctness of the generated phrases. For example :

For every unique list LIST, every positive integer NUMBER2 and every positive integer NUMBER1, we have that if the element with number NUMBER1 in LIST is equal to the element with number NUMBER2 in LIST, then NUMBER1 is equal to NUMBER2.

For the same axiom, the SUMO templates generate:

For all unique list ?LIST holds for all ?NUMBER1, ?NUMBER2 holds if ?NUMBER1th element of ?LIST is equal to ?NUMBER2th element of ?LIST, then ?NUMBER1 is equal to ?NUMBER2

The optimized natural language generation mechanism from the Sigma system would render the axiom as:

* *If a list is an instance of unique list*

* *then for all a positive integer and positive integer*

∘ *if positive integerth element of list is equal to positive integerth element of list*

∘ *then positive integer is equal to positive integer*

Further optimizing of the code by anaphora generation and a list-like structure of the arguments for better readibility is possible, like in the proof rendering.

5. Evaluation

During the translation of SUMO to GF, we discovered a number of small inconsistencies in the original ontologies like mismatches between instances and classes, usage of undefined objects and usage of functions with a wrong number of arguments. This represents almost 8% of the total number of axioms from SUMO and was determined automatically during the type-checking phase. In addition to this, we left out the higher-order logic constructions such as quantifications on `Formula` or axioms with higher-order functions.

However, there are some types of axioms which could not be ported to SUMO-GF, such as the ones that use quantification over classes, negative type declarations and axioms which use the predicates `subclass`, `range` or `domain`. In addition to this, we mention the class of axioms which feature conditional type declarations. For example:

$$\begin{array}{l}(=> (and\ (instance\ ?DRINK\ Drinking) \\ \qquad\qquad (patient\ ?DRINK\ ?BEV)) \\ \quad (instance\ ?BEV\ ?Beverage))\end{array}$$

The type declaration for *BEV* appears as a consequence of the fact that it is used in the process of *Drinking*. The total number of axioms which are lost in translation is about 23%. Our observations suggests that those axioms could be paraphrased and incorporated in the type system but this would require manual work with every axiom.

6. Automated Reasoning in SUMO-GF

Since SUMO offers a generous amount of information in a first-order logic format, it represents an excellent source for automated reasoning, especially in the context of SUMO-GF where one can perform automated reasoning on natural language.

As mentioned before, the TPTP-FOF translations of the 2 upper SUMO ontologies are used yearly in the ATP competition. We have shown already in Section 1 that a limited kind of ontological reasoning is possible by using GF alone. Unfortunately, the reasoner in GF is not as optimized as current state of the art theorem provers. However, to take advantage of the tools that already exists, we translated the 17 SUMO-GF ontologies to TPTP-FOF, checked them for consistency and used them for solving small inferences.

Since TPTP is an untyped system, whereas GF is strongly typed, the information about types needs to be reformulated, with the aid of an additional predicate `hasType`, that resembles the original `instance` predicate from SUMO.

For subclasses, the translation reflects the possibility of coercing from the subclass to the superclass:

$$\textbf{fun}\ \mathit{Adjective\_Class} : \mathit{SubClass\ Adjective\ Word};$$

and would be translated to TPTP as:

```
fof(axMerge2, axiom, (![X]:
     (hasType(type_Adjective, X)=>hasType(type_Word, X)))).
```

For instance declarations, we have a simpler translation pattern:

$$\textbf{fun}\ \mathit{Flat} : \mathit{Ind\ ShapeAttribute};$$

will be translated to TPTP as:

```
fof(axMerge686, axiom,
      hasType(type_ShapeAttribute, inst_Flat)).
```

A more commonly used approach for expressing typing declarations in first-order logic is to create a predicate for each type, like:

```
type_ShapeAttribute (inst_Flat)
```

We did not choose this method, since the SUMO classes are not just used as types, in typing declarations, but also as arguments for some functions. By using classes as predicates, one couldn't unify the two occurrences in first-order logic.

The functions that manipulate *Formula* objects, such as *not*, *and*, *or*, *impl* and *equiv* have been translated into their corresponding first-order logic operators that are predefined in TPTP: $\sim$, &, $|$ and $\Rightarrow$. For the *both* and *either* constructors, the built-in & and $|$ are used again:

$$\textbf{fun}\ \mathit{Togo} : \mathit{Ind}\ (\mathit{both\ Country\ Nation});$$

will be translated to TPTP as:

```
fof(axmond72, axiom,
      hasType(type_Country, inst_Togo) &
               hasType(type_Nation, inst_Togo)).
```

As for the equality operator *equal*, the situation is more complicated. In SUMO, because of the structure of the concepts, it could basically take any arguments, like classes, and relations and instances. In GF the *equal* function would just take arguments of type *El Entity*, so it would not be possible to test the equality of formulas, functions or classes. In SUMO, *equal* is defined as an *EquivalenceRelation*, with some extra axioms, for the various kinds of arguments that it might take. For instances, the axiom, that verifies a property of equal objects:

$$\begin{array}{l}(=>\ (\mathit{equal}\ ?\mathit{THING1}\ ?\mathit{THING2})\\ \quad (\mathit{forall}\ (?\mathit{CLASS})\\ \qquad\qquad (<=>\ (\mathit{instance}\ ?\mathit{THING1}?\mathit{CLASS})\\ \qquad\qquad\qquad (\mathit{instance}\ ?\mathit{THING2}\ ?\mathit{CLASS}))))\end{array}$$

could not be translated to GF, as it contains a variable type declaration and quantification over a class. Moreover, a more solid interpretation of equality would be using at least a congruence relation, not just an equivalence one. SUMO does not have the concept of congruence, while theorem provers that can process first-order logic with equality, usually have specific mechanisms for dealing with the built-in equality from TPTP [23]. For these reasons, the translation from GF to TPTP, uses the default TPTP equality for the `equal` function.

The existential and universal quantifiers from SUMO and GF, were translated as the built-in quantifiers from TPTP. The type declarations are expressed with the function `hasType` for consistency with the type declarations. For example, the axiom from before was translated to TPTP as:

```
fof(axMid9, axiom, ![Var_P]:
 hasType(type_Wading, Var_P) => ?[Var_W]:
 hasType(type_BodyOfWater, Var_W) & f_located(Var_P,Var_W)).
```

A special case is the translation of higher-order axioms to TPTP. In this case, the function call is replaced by the definition of the function, rendering a construction in first-order logic. In this way the function name is used as a macro for its body. This is the same approach as in [14].

The resulting files have been checked with the first-order theorem prover **E** [24]. **E** is a multiple award-winning theorem prover which is freely available and is based on an equational superposition calculus. It provides support for first-order logic with equality. **E** has been used to check the consistency of the largest ontology currently available - ResearchCyC [25]. The TPTP translations of the GF files were tested for consistency with **E**, and no contradiction was found, given the time limit of 1 hour per file, which was exceeded for the upper-ontologies, due to the increased complexity of the axioms they contain.

Regarding typical inferences that could be solved on the existing data, we used the problems from the SUMO webpage [4].

The category of axioms that the SUMO to TPTP translation can express, but the SUMO-GF to TPTP cannot are mainly the ones that got lost in the SUMO to SUMO-GF translation. In addition to this, there are the nested predicates, quantifications over `Formula` and the class-forming function `KappaFn`. They are used in SUMO-GF only for language generation. The loss is almost 23% of the total number of the axioms. The coverage of the SUMO to GF to TPTP translation is comparable to the direct SUMO to TPTP translation. It is worth mentioning that the first translation yields to a slightly slower system because of the additional type declarations that need to be checked by the theorem prover. However, it is worth investigating if the results could be better, if one chooses the typed version of TPTP. [5].

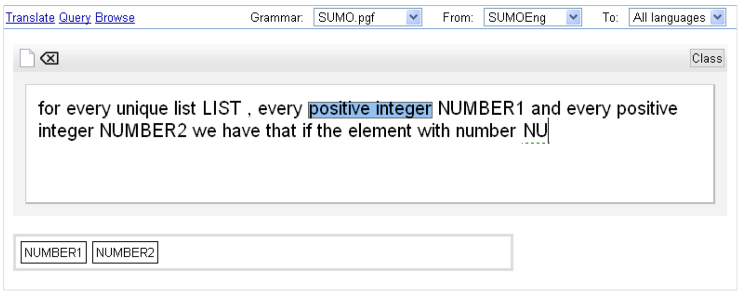
7. End-user Interface

An important component of the GF distribution is the front-end user interface. While the grammarians are supposed to use the GF shell plus some development environment for writing grammars, end users should have the option to use some more comfortable interface. GF comes with a generic web-based interface [26] which could be specialized further for particular applications. In relation with SUMO, the interface was extended with features which make the relation of the ontology with the concrete syntax more transparent.

[4]http://sigmakee.cvs.sourceforge.net/viewvc/sigmakee/KBs/tests/

[5]http://www.cs.miami.edu/ tptp/TPTP/Proposals/TypedFOF.html

Figure 3.1: Text editor for authoring SUMO axioms using controlled natural language



for every unique list LIST , every positive integer NUMBER1 and every positive integer NUMBER2 we have that if the element with number NU

While in Sigma the knowledge engineers are supposed to write the axioms in KIF, in GF they could do it in a controlled natural language. The problem with all controlled languages is that the user has to learn how to write content which is in the scope of the grammar. The successful use of predictive editors in helping the users build constructions within the bounds of the controlled language was investigated in [27]. In GF there is a similar predictive editor (fig. 3.1) which guides the authoring by generation of suggestions. In the example shown the user has just started adding a reference to a variable and the editor suggests the list of all variables in the current scope which start with “NU”. The same kinds of suggestions are offered for every word in the sentence. Furthermore, the user could at any time select a phrase (the highlighted phrase on the picture) and see in the upper-right corner the corresponding ontological type of the phrase (*Class* in this case). If the axiom is not well-formed, i.e. contains a type error for example, then the error is immediately reported and the corresponding phrase is underlined. This very much resembles an IDE for programming languages, except for the fact that the input is a kind of natural language.

For nontrivial ontologies of the scale of SUMO it is often helpful to have an overall view of the ontology. The browsing functionalities of Sigma very much fulfil the requirements. A similar browser (fig. 3.2) for the abstract syntax of the grammars was developed for GF. In the case of SUMO-GF, this corresponds exactly to the taxonomy plus the signatures of all functions and predicates. The user sees the class hierarchy on the left-hand side and can start with the exploration of any class, or could use the search box in the upper-left corner to find a class or function by name.

8. Related Work

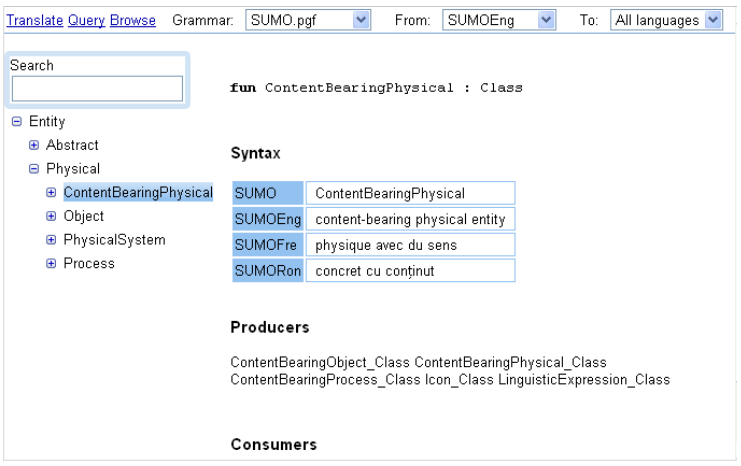
At the moment there exists a large number of applications dealing with ontologies and building various applications on top of them. Regarding the languages that are used to encode ontologies, as mentioned before, the most popular ones do not have a type system.

A first exception is the programming language prototype Zhi#[6], which is a novel language for encoding ontologies. It has a static type-system and it is compiled to C#. It benefits from using the C# built-in types and functions, but also the syntax looks very much like C# and it is not very intuitive for most users.

A more notable example is CASL (Common Algebraic Specification Language) [28]. It introduces the notions of strongly-typed and structured ontologies and provides a strong formal structure for representing them. However, it deals only with the

[6]http://www.alexpaar.de/zhimantic/ZhiSharp.pdf

Figure 3.2: Browser for combined ontology and syntax exploration



```
fun ContentBearingPhysical : Class
```

Syntax

SUMO	ContentBearingPhysical
SUMOEng	content-bearing physical entity
SUMOFre	physique avec du sens
SUMORon	concret cu conținut

Producers

ContentBearingObject_Class ContentBearingPhysical_Class ContentBearingProcess_Class Icon_Class LinguisticExpression_Class

Consumers

algebraic side of the specifications, whereas GF has a built-in natural language generation component, in addition to the robust type system.

Compared to these languages, GF is the only system which combines a strongly typed framework for ontology description with a direct multilingual verbalization. To our knowledge, the current work is the first representation of an ontology in type theory with dependent types. The benefits of dependent types are visible when expressing the concepts and relations from SUMO in GF, as they provide better control on their semantics. robustness to the representation.

Regarding natural language generation, there are many notable applications that verbalize ontologies. Most of them however, have only English as target. A notable exception is the KPML project [29], which provides natural language generation for 10 languages. Another interesting case is the Gellish ontology which provides direct verbalization for English, German and Dutch. However, there is considerably less progress for Slavic and Romance languages, due to their complexity. The GF approach has built-in mechanisms for verbalization via the resource grammars, which provide syntactically correct translations. Moreover, GF also has support for multilingual translation.

Regarding automatic reasoning, there has been work for checking the consistency of all the well-known ontologies. A notable example is the use of the E theorem prover for the ResearchCyC ontology [25]. However, SUMO is the most well-known case of an ontology which is checked for consistency every year, as part of a CADE competition. Compared to the official SUMO translation to TPTP, our approach has a comparable expressivity, rejecting the ill-typed axioms at an earlier stage.

The project OntoNat [30] provides automated reasoning for the SUMO ontology with KRHyper [31]. KRHyper is a theorem prover for first-order logic that implements hyper tableaux, and a version of it that deals with equality is also available [7]. The tool can answer questions posed in normal English, by using the wordnet mappings and a simple parser, in order to infer the SUMO expression that should be checked.

[7]http://www.uni-koblenz.de/ bpelzer/ekrhyper/

9. Future Work

The current work explores aspects of data modelling, compiling from an untyped system to a typed one and from a typed system to first-order logic, type inference, natural language generation, and automated reasoning. These directions can be extended in a more comprehensive manner and lead to stand-alone projects.

One interesting possibility would be to generate higher-quality natural language, following the idea[8] of truncating the hierarchy even more, separating `attributes` and `processes`. Instances of `attribute` and its subclasses can be linearized as adjective phrases, while instances and subclasses of `process` are to be linearized as verb phrases. In this way a predicate like:

$$(attribute\ ?X\ NonFullyFormed)$$

would not be linearized as *non fully formed is an attribute of X* but as *X is not fully formed*. For a predicate like:

$$(agent\ Reasoning\ ?A)$$

we would obtain *A reasons* instead of *A is an agent of reasoning*.

Another interesting application would be to build a user interface, where users could ask questions and get answers from the theorem prover via the GF to TPTP translation. If the prover provides a trace of the proof search, this could be converted back to a GF tree and used for generation of proof explanations in natural language. When dealing with more complex proofs, more work is needed for rendering readable natural language. A comprehensive reference for natural language generation from proofs is [32].

10. Conclusion

The work investigates the representation of upper ontologies in the type-theoretical functional language GF, which provides mechanisms for direct verbalization as a controlled language, having the SUMO ontology as a use case. The results obtained show a consistent improvement from the multilingual natural language generation point of view, in terms of effort, scalability and syntactical correctness of the obtained text. Moreover, the type system, while still preserving a comparable coverage, prevents type errors that could lead to inconsistencies in the ontology. Also the editor makes it easier for users to interact with the ontology by adding content in natural language. From a knowledge engineering point of view, GF offers obvious advantages for encoding ontologies, as the framework defined and applied for SUMO is general enough to fit a large range of ontologies.

[8] http://www.ontologyportal.org/student.html

Bibliography

[1] Ganesereth, M.R., Fikes, R.E.: Knowledge interchange format. Technical Report Logic-92-1, Stanford University (June 1992)

[2] Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F., Stein, L.A.: OWL web ontology language reference. (2009)

[3] Cycorp: The syntax of CycL (2002)

[4] Van Renssen, A.: Gellish: A Generic Extensible Ontological Language. PhD thesis, Delft University (2005) PhD thesis.

[5] Niles, I., Pease, A.: Towards a standard upper ontology. In: FOIS '01: Proceedings of the international conference on Formal Ontology in Information Systems, New York, NY, USA, ACM (2001) 2–9

[6] Ranta, A.: Grammatical Framework: A Type-Theoretical Grammar Formalism. The Journal of Functional Programming **14(2)** (2004) 145–189

[7] Martin-Löf, P.: Constructive mathematics and computer programming. In Cohen, Los, Pfeiffer, Podewski, eds.: Logic, Methodology and Philosophy of Science VI. North-Holland, Amsterdam (1982) 153–175

[8] Ranta, A.: The GF resource grammar library. Linguistic Issues in Language Technology **2**(2) (2009)

[9] Angelov, K., Bringert, B., Ranta, A.: PGF: A Portable Run-Time Format for Type-Theoretical Grammars. Journal of Logic, Language and Information (2009)

[10] Ljunglöf, P., Amores, G., Cooper, R., Hjelm, D., Lemon, O., Manchin, P., Perez, G., Ranta, A.: Multimodal Grammar Library (2006) TALK. Talk and Look: Tools for Ambient Linguistic Knowledge. IST-507802. Deliverable 1.2b.

[11] Caprotti, O.: WebALT! Deliver Mathematics Everywhere. In: Proceedings of SITE 2006. Orlando March 20-24. (2006)

[12] Ahrendt, W., Baar, T., Beckert, B., Bubel, R., Giese, M., Hähnle, R., Menzel, W., Mostowski, W., Roth, A., Schlager, S., Schmitt, P.H.: The key tool. Technical report in computing science no. 2003-5, Department of Computing Science, Chalmers University and Göteborg University, Göteborg, Sweden (2003)

[13] : MOLTO - Multilingual Online Translation. European Project (2010–2012)

[14] Pease, A., Sutcliffe, G.: First order reasoning on a large ontology. Proceedings of the CADE-21 workshop on Empirically Successful Automated Reasoning on Large Theories (ESARLT) (2007)

[15] Sutcliffe, G.: The TPTP problem library and associated infrastructure. Journal of Automated Reasoning **43** (2009) 337–362 10.1007/s10817-009-9143-8.

[16] Pease, A.: The sigma ontology development environment. Working Notes of the IJCAI-2003 Workshop on Ontology and Distributed Systems **71** (2003)

[17] Trac, S., Sutcliffe, G., Pease, A.: Integration of the tptpworld into sigmakee. Proceedings of IJCAR '08 Workshop on Practical Aspects of Automated Reasoning (PAAR-2008) **373** (2009)

[18] Martin-Löf, P.: A theory of types (1971) unpublished.

[19] Girard, J.Y.: Interpretation fonctionnelle et elimination des coupures de l'arithmetique d'ordre superieur, Paris (1972)

[20] W3C: Named graphs (2004)

[21] Russell, B.: Principles of Mathematics. Cambridge University Press, Cambridge (2011)

[22] Ranta, A.: Structures grammaticales dans le français mathématique. Mathématiques, informatique et Sciences Humaines (138, 139) (1997) 5–56, 5–36

[23] Slagle, J.R.: Automatic theorem proving with built-in theories including equality, partial ordering, and sets. J. ACM **19** (1972) 120–135

[24] Schulz, S.: E - a brainiac theorem prover (2002)

[25] Ramachandran, D., Reagan, P., Goolsbey, K.: First-orderized researchcyc: Expressivity and efficiency in a common-sense ontology. In: In Papers from the AAAI Workshop on Contexts and Ontologies: Theory, Practice and Applications. (2005)

[26] Bringert, B., Angelov, K., Ranta, A.: Grammatical framework web service. In: EACL (Demos). (2009) 9–12

[27] Schwitter, R., Ljungberg, A., Hood, D.: ECOLE — a look-ahead editor for a controlled language. In: Proceedings of EAMT-CLAW03. (2003) 141–150

[28] Lüttich, K.: Development of Structured Ontologies in CASL. PhD thesis, University of Bremen (2007) PhD thesis.

[29] Bateman, J.A.: Enabling technology for multilingual natural language generation: the kpml development environment. Nat. Lang. Eng. **3**(1) (1997) 15–55

[30] Baumgartner, P., Suchanek, F.M.: Automated reasoning support for sumo/kif. (2005) Manuscript, Max-Planck Institute for Computer Science.

[31] Wernhard, C.: System Description: KRHyper. Fachberichte Informatik 14–2003, Universität Koblenz-Landau (2003)

[32] Fiedler, A.: Natural language proof explanation. In Hutter, D., Stephan, W., eds.: Mechanizing Mathematical Reasoning. Volume 2605 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (2005) 342–363

2 Multilingual Verbalization of Modular Ontologies using GF and lemon

Brian Davis Ramona Enache Jeroen van Grondelle
Laurette Pretorius

Abstract: This paper presents an approach to multilingual ontology verbalisation of controlled language based on the Grammatical Framework (GF) and the *lemon* model. It addresses specific challenges that arise when classes are used to create a consensus-based conceptual framework, in which many parties individually contribute instances. The approach is presented alongside a concrete case, in which ontologies are used to capture business processes by linguistically untrained stakeholders across business disciplines. GF is used to create multilingual grammars that enable transparent multilingual verbalisation. Capturing the instance labels in *lemon* lexicons reduces the need for GF engineering to the class level: The *lemon* lexicons with the labels of the instances are converted into GF grammars based on a mapping described in this paper. The grammars are modularised in accordance with the ontology modularisation and can deal with the different styles of label choosing that occur in practice.

1. Introduction

As the adoption of ontologies into enterprise application environments grows, new audiences have to deal with ontologies beyond the traditional disciplines that have done so in the past, such as knowledge engineers and ontologists. These audiences range from business users, who need to take ownership of the ontologies, to end users, such as customers or citizens, who are presented with the services based on these ontologies. As the formalisms themselves are often inaccessible to these new audiences, appropriate visualisations are important. Our experience in practice is that business users often overcome their perception of graph oriented visualisations being too technical when gaining experience, but they remain a challenge for incidental reviewers and end users. Therefore, verbalisation of ontologies into natural language is one of the approaches that is crucial to make ontologies accessible to new audiences.

Being able to provide verbalisation in a multilingual manner is important: Governments and enterprise often offer their products and services in international contexts or to customers of different languages. For instance, Dutch Immigrations offers many of its services based on ontologies [1], and it typically needs to interface with people that do not speak Dutch. Also, governments have to deal with numerous international aspects in legislation when drafting their national laws. Specifically in Europe, large parts of national legislation is either heavily influenced by or originates in European legislation. Sharing ontologies capturing such international legislation and being able to refer to them from local ontologies offers important benefits in areas of productivity and traceability across local practices.

In this paper, we present an approach for ontology verbalisation based on the Grammatical Framework (GF) and the Lexicon Model for Ontologies (*lemon*) that is essentially multilingual and addresses the particular challenges when classes are chosen up front based on consensus, while multiple parties contribute often changing instances individually.

We use the Be Informed business process platform[9] as an example throughout this paper. The verbalisation examples described in [2] are built upon and we show how they can be generated across languages transparently. Finally, we illustrate how the challenges in this example generalise across other examples, including the generic case of verbalising Linked Open Data.

2. Related Work

Ontology Lexicalisation

The *lemon* model builds on previous research for designing lexica for interfacing with ontologies, in particular that of the LexInfo [3] and LIR [4] models, as well as existing work on lexicon (meta-)models, in particular the Lexical Markup Framework (ISO-24613:2008) [5]. In addition, it builds on efforts to link resources via the Web to the ISOcat meta-data registry [6]. *lemon* seeks to scale the modelling of lexical and linguistic information related to concepts in ontologies from very simple to quite complex lexical entries. *lemon* is closely related the SKOS project[10], which attempts to model simple knowledge organisation systems such as thesauri, classification schemes and taxonomies on the Semantic Web.

Ontology Verbalisation and CNLs

The application of CNLs for ontology authoring and instance population is an active research area. *Attempto Controlled English*[11] (ACE) [7], is a popular CNL for ontology authoring and generation. It is a subset of standard English designed for knowledge representation and technical specifications, and is constrained to be unambiguously machine-readable into DRS - Discourse Representation Structure, a form of first-order logic. WYSIWYM (*What you see is what you meant*)[8], involves direct knowledge editing with natural language directed feedback. A domain expert can edit a knowledge base reliably by interacting with natural language menu choices and the subsequently generated feedback, which can then be extended or re-edited using the menu options. With respect to GF, there have been a number of GF grammars using ontologies/databases, such as the grammar representing the structure of SUMO, the largest open-source ontology [12] as a type-theoretical grammar and verbalise it in 3 languages [9]. Moreover, an ontology describing paintings and a database describing over 8,000 artifacts from the Gothenburg City Museum were used for generating descriptions of the paintings in 5 languages [10].

3. Challenges in Verbalising Modular Ontologies

3.1. New audiences for ontologies

An example where ontologies are used in business applications is Be Informed's business process platform. It allows representatives of many disciplines and domain experts

[9]`http://www.beinformed.com/`
[10]`http://www.w3.org/2004/02/skos/`
[11]`http://www.ifi.unizh.ch/attempto/`
[12]`http://www.ontologyportal.org/`

to capture the definitions and constraints that the business process must meet and have engines infer executable business processes and decision services from them automatically. This provides business users with a large degree of control and agility, as the constraints are easily changed and the process is updated accordingly. The resulting processes are highly contextual and deal well with exceptions: They allow professionals to influence their own work, and are resilient to the effects of overrides as is described in [2].

When ontologies are used to infer business processes in this way, its stakeholders get to interact with the underlying ontologies on a number of levels. The first is obviously understanding the content of the ontology and the consequences for the inferred business process. The ontology becomes part of the system documentation and is used for validation and reviewing at specification time or for reference when using and maintaining the services and its specification/documentation. Also, the ontology driven services have user interfaces that provide dialog to its end users and the ontology typically is the source of many of the textual elements in that dialog, such as the questions asked, the captions in forms, etc. Finally, the end results of the ontology driven services are also based on the underlying ontology. In applications, these results might be represented by generated documents or letters sent to customers. Specifically when decisions are taken based on models, explaining the end result and the underlying argument have strong ties with the concepts and their textual representations in the ontology.

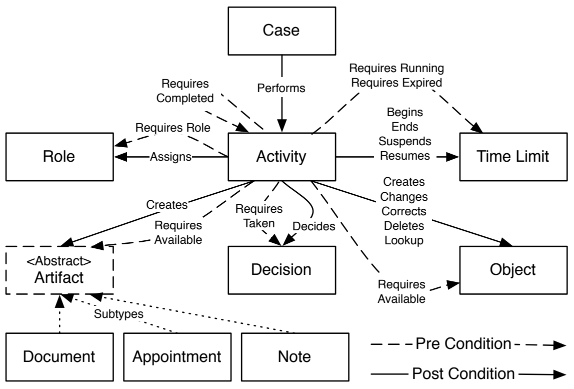


Figure 3.3: Summary of the classes used to capture business processes

Be Informed's business processes are inferred from an ontology capturing all relevant activities, artifacts, involved roles etc. and the conditional relations between these. The meta model, as shown in Figure 3.3, contains the conceptualisation of the business process domain, that instead of using flow semantics to capture who does what and when, is based on pre- and post-conditions. Each activity may have pre-conditions that have to be met before it may be performed, and post-conditions that capture what conditions have to be met in order for the activity to be complete. Figure 3.4 contains an of example how this meta model is used to capture the business process of applying for a grant.

In [2], this example is discussed in more depth and the model from Figure 3.4 is verbalised using a pattern sentence approach. The result when verbalising the example

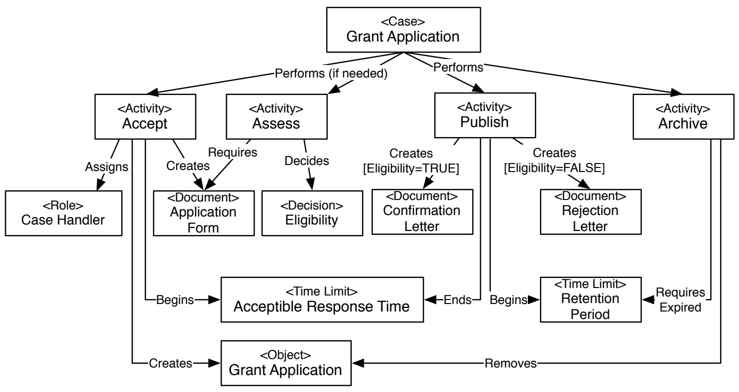


Figure 3.4: An example of an grant application process

used throughout the paper with the pattern sentence approach is the following.

1. The activity PUBLISHING THE RESULT may be performed if
   (a) a document of type DOCUMENT WITH DETAILS is available.
2. The activity PUBLISHING THE RESULT is completed if
   (a) a document of type SUBMISSION FORM has been created.

Although the sentences generated have proven useful in practice, some additional requirements exist in areas such as grammatical correctness and fluency. For instance, the first sentence part is grammatically incorrect, as the label inserted is a verb phrase in itself. Also, the pattern sentences approach proves not to scale in terms of number of supported languages. Every grammar translation needs to deal with all lexical aspects (and their peculiarities) of their specific language.

3.2. Differences between classes and instances

Within the field of knowledge representation, both classes and instances are typically treated as equal means of expression when creating an ontology and both can be used by the knowledge engineer as he sees fit to best capture the conceptualisation of his choice.

In the use case presented in this paper however, class and instance information are typically in separate, but linked ontologies and these ontologies differ greatly in characteristics related to ownership and rate of change:

- Classes are chosen based on consensus across multiple parties, while the instances are provided by individual parties without requiring consensus on the data;
- The classes are often determined once and are fixed, whereas the instances of these classes are introduced over time and may change often;
- Classes may use an ontology formalism as its native/original representation, where instances often have existing databases or other information stores as primary sources, and the ontology is used exclusively for sharing the data;

- The classes are typically created by knowledge engineers or other information professionals, while instances are created by a wide range of people, who enter data in the original databases, and might not deal with ontologies at all.

In the Be Informed use case provided throughout this paper, the product contains default meta models consisting of the classes that are used in modelling. Although these might be adjusted or extended in individual implementations, this is typically done early in the design phase. An example of such a meta model is shown in Figure 3.3. When adopting the product, the majority of effort is spent on creating detailed models based on the classes in the meta model. Meta models and their extensions are typically created by knowledge engineers, while the models themselves are built by both knowledge engineers and domain experts. Moreover, it is considered crucial that business users, who adopt the complete model for validation, can make changes and updates.

The choices made in the lexicalisation/verbalisation of classes and instances respectively, follow a similar pattern as the classes and instances themselves:

- In general the labels and lexical representations for the classes are created alongside the ontology that contains the classes and may require guidelines to be met. The labels of the instances are often chosen by people less or unskilled in knowledge representation or may even originate from existing databases. This compromises the coherent adherence to guidelines for the lexical properties of instance labels;

- Classes may have complex lexical and grammatical consequences, and also introduce sentence patterns and planning. Instances have only labels with limited complexity at a lexical level.

The separation between reaching consensus on the types of things considered and identifying the individual things generalises well across other scenario's of both the use of ontologies and their verbalisation. For instance, ontologies used in Linked Open Data typically expose these characteristics. An example of this is the verbalisation of structured knowledge about 8000 artifacts from the Gothenburg City Museum into natural language descriptions [10]: The conceptualisation of painters, paintings and materials is codified in an ontology with classes, the facts about the individual paintings are exported out of a database into ontologies with instances.

3.3. Practices in choosing labels

In practice, different styles of choosing labels are found when modellers are, for instance, modelling and naming concepts within a business process model.

In practice, there is a difference between concepts that are commonly referred to by a proper name (term) and concepts that do not have a term associated with them and are referred to by description. For example, when modelling business processes, the ontology typically contains activities. Some activities may have names (terms), but more often a label describes what is done in the activity. Examples are "Publishing the result", "Publish the result" or "The result is published" (when the activity is completed). This phenomenon also occurs in multilingual contexts when a name for the concept exists in the source language of a model but not in the target language, in which case some form of description is necessary.

Another source of label choosing practices can, especially in business use of ontologies, be found in the diverse background of the people involved in modelling. Typically, the domain experts may have experience in other structured conceptualisation approaches and the naming conventions or practices that come with it. For instance, many information professionals have backgrounds in systems development disciplines and are familiar with techniques like UML and Entity Relational modelling, influencing their naming practices.

Providing guidelines and practices for systematically choosing good labels contributes to reducing the number of label variants used in modelling. However, for industry adoptability and robustness in practice, ontology verbalisation techniques should be able to deal systematically with these challenges and practices, both in terms of the number of constraints required for accurate modelling and also the implications of language variation and multilingualism for any given ontology concept.

4. Ontology Verbalisation using GF and *lemon*

We present a multilingual ontology verbalisation approach that addresses the challenges discussed in Section 3. It is based on the Grammatical Framework, currently developed in the Molto Project[13], and *lemon*, the Lexicon Model for Ontologies, currently developed in the Monnet Project[14].

Both projects have a strong focus on ontologies and multilingualism, and complement each other well in our approach: GF provides sophisticated multilingual grammar support while *lemon* provides state of the art ontology lexicalisation techniques.

4.1. Introduction to *lemon* and GF

Lexicon **M**odel for **On**tologies, or *lemon* is a model for representing lexical information about words and terms relative to an ontology on the Web. *lemon* is what we term an *ontology-lexicon* in that it expresses how the elements of the ontology, i.e. classes, properties, and individuals, are realised linguistically. The model follows a principle called *semantics by reference* whereby it is assumed that the (lexical) meaning of the entries in the lexicon are expressed exclusively in the ontology. Hence the lexicon merely points to the appropriate concepts. *lemon* is designed to be a basic model supporting the exchange of ontology-lexica on the Semantic Web. The core of *lemon* contains the basic elements required to define lexical entries and their association to their lexical forms and, moreover, to concepts in the ontology representing their meaning. It consists of:

- **Lexicon:** This object represents the lexicon as a whole. It must be marked with a language, and all objects in the lexicon belong to this language.
- **Lexical Entry:** An entry for a given lexicon is a container for one or several forms. It also contains one or more meanings of a lexeme. All forms of an entry must have the same part of speech. An entry may have multiple meanings.
- **Lexical Form:** This is the inflectional form of an entry. It must have one canonical form, but may have any number of other forms. Stems and other partial morphological units can also be modeled as abstract forms.

[13]`http://www.molto-project.eu/`
[14]`http://www.monnet-project.eu/`

- **Representation:** A lexical form may have several representations, ranging from different orthographies, to phonetic representation as well as standard written representation.
- **Lexical Sense:** This links a lexical entry to the **reference** in the ontology i.e. a concept, property or instance.
- **Component:** A lexical entry may also be broken up into a number of components.

The following example gives a simple lexicon with a single lexical entry:

```
@prefix lemon: <http://www.monnet-project.eu/lemon#>.
@prefix bpo: <http://www.beinformed.com/resource/>.

:lexicon lemon:entry:adult_applicant;
  lemon:language "en".

:adult_applicant
  lemon:canonicalForm [lemon:writtenRep "Applicant is Adult"@en];
  lemon:sense [lemon:reference bpo:adult_applicant].
```

This simple English lexicon has a single entry, with canonical form "Applicant is Adult", and a sense that refers to the entry in Be Informed Business Process ontology.

The **G**rammatical **F**ramework (GF) [11] is a formalism for describing multilingual grammars. A GF grammar consists of an abstract syntax, acting as a semantic interlingua and a number of concrete grammars that verbalise the abstract syntax in multiple languages. In this way the semantic level is the central part of the grammar, connecting any language pair of concrete syntaxes.

Most GF grammars are used to describe fragments of natural language. The largest and most general such grammar is the resource grammar library, which contains *resource grammars* for 24 languages, and implements the most common syntactic constructions (such as predication, complementation, etc.) for these languages.

The resource grammar library supports the development of so-called *application grammars* for more restricted domains by providing standard language-specific technicalities so that they do not need to be described again in the new (application) grammar. This makes it easier to develop application grammars by assembling the primitive constructs from the resource grammar and obtain syntactically-correct text in languages from the library.

In the work reported on in this paper GF is used to develop an application grammar for the Be Informed use case, discussed in section 3.

4.2. Modular GF grammars based on decoupling of classes and instances

The ontology verbalisation approach exploits the complementary strengths of GF and *lemon*. GF is used to capture ontological information as well as the required sentence structure while *lemon* is the source of concrete label information. The approach is explained by using as example the Be Informed business process ontology and the pre- and post-condition sentence patterns of [2]. The aim of the business processes ontology in Figures 3.3 and 3.4 is to specify business processes in terms of pre- and post-conditions, which in turn requires the verbalisation of such conditions.

A challenge resulting from the strict separation of (ownership of) TBox and ABox is that the lexical information required in verbalisation of the complete ontology also needs modularisation along these boundaries and that restrictions with respect to availability/feasibility of knowledge engineering to the different ontology parts also apply to the lexical information associated with those parts.

The grammar modularisation follows the structure of the meta model in Figure 3.3 (a similar approach was followed in [10]). The verbalisation proceeds according to the sentence patterns described in [2]. In particular, each activity may have pre-conditions and post-conditions verbalised as *conditional statements* ("A if B"), where A and B are simple *propositional statements* with modalities, as appropriate. All *concept labels* are to be verbalised as propositional statements in accordance with specifications, as explicated in [2].

The modular layered approach to verbalisation is illustrated in Figure 3.5 by means of the pre-condition triple

```
(Activity,Requires_Available,Artifact subtyped as Document).
```

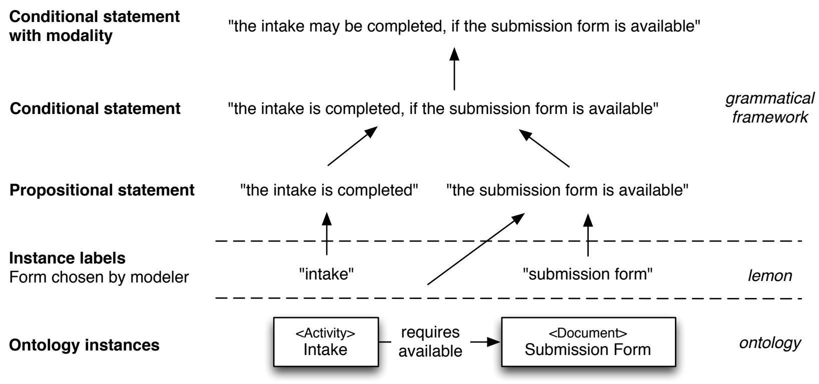

Conditional statement with modality
"the intake may be completed, if the submission form is available"
Conditional statement
"the intake is completed, if the submission form is available"
grammatical framework
Propositional statement
"the intake is completed"
"the submission form is available"
Instance labels
Form chosen by modeler
"intake"
"submission form"
lemon
Ontology instances
<Activity> Intake
requires available
<Document> Submission Form
ontology

Figure 3.5: Layered verbalisation procedure

The TBox abstract syntax caters for the pattern sentence structure (the conditional and propositional statements), the modalities, the concept classes and the relations between them. The polarities and modalities are introduced as separate functions, allowing for the addition of more modalities by merely creating two additional functions per modality (positive and negative form), using the primitive operation already defined. The (meta model) classes are modelled as GF categories and the relations become GF functions with a return type used for verbalisation, as shown in the code segment taken from the TBox abstract syntax module:

```
 cat
  Activity;  -- meta model type
  Document;  -- meta model type
  Artifact;  -- meta model type

  Fragment;  -- type for proposition
  BIText;    -- type for language generation

fun
  requires_available : Activity -> Artifact -> Fragment;
  subDocArtifact : Document -> Artifact;
  FCan : Fragment -> BIText;
```

The conceptual modularisation of the grammar is completed by the ABox abstract syntax, which uses the TBox abstract syntax, and contains the label/instance definitions (ABox information) as GF function declarations with category types, for example,

```
AIntake : Activity;
SubmissionForm : Document;
```

The verbalisation of the information captured by the ontology, as well the pattern sentences in which they may occur, is addressed in the concrete syntaxes for English and Dutch. The linearisation categories for the (abstract syntax) categories provide detailed linguistic structure in terms of parts of speech etc, for rendering correct verbalisations. Complex (GF record) types are used to facilitate label variants. The predicates that are built with transitive verbs (creates, deletes, corrects, changes) are rendered in the passive voice, as required by the Be Informed application. The TBox concrete syntax also provides primitives for creating the main categories, for example `mkActivity` (see next section) for `Activity` from basic parts of speech from the resource library, and hides the implementation details from the users, thereby ensuring that the optimisations are as seamless as possible.

The TBox concrete grammar (code segment below) provides the linearisation of both the propositional and the conditional statements. In particular, the overloaded function `mkFragm` implements both the simple propositional statement (as complete sentence pattern) (`hasExt = False`) and the conditional statement (`hasExt = true`) by means of pattern matching on the number and types of the arguments. The overloaded function `mkBIText` completes the verbalisation by finally adding modality and polarity, as necessary. We show `mkFragm` and a part of `mkBIText` by way of illustration.

`Utt` and `S` are GF resource grammar library categories for sentences, questions, etc. and declarative sentences, respectively, while `Fragm` is a user defined complex (GF record) type. `VV` is the the resource grammar library category for verb-phrase-complement verbs.

```
lincat
  Activity  = {noun : NP; subj : NP; vp : VP; hasVerb : Bool};
  Document, Artifact = NP;

lin
  requires_available ac ar = mkFragm ac.subj ac.vp (mkS (mkCl ar
  (mkVP available_A)));
  subDocArtifact d = d;
  FCan frag = mkBIText frag positivePol may_VV;

oper
  Fragm = {subj : NP; pred : VP; ext : {s : S; hasExt : Bool}};

  ifExt : {s : S; hasExt : Bool} -> S -> S = \ext,s -> case
  ext.hasExt of {
    True  => Sentence.SSubjS s if_Subj ext.s;
    False => s
    };

  mkFragm = overload {
  mkFragm : NP -> VP -> Fragm =
    \np, vp ->
```

```
      {subj = np;
      pred = vp;
      ext = {s = dontCareS; hasExt = False}};

  mkFragm : NP -> VP -> S -> Fragm =
    \np, vp, sub ->
      {subj = np;
      pred = vp;
      ext = {s = sub; hasExt = True}};
  };

  mkBIText = overload {
  mkBIText : Fragm -> Pol -> Utt =
    \frag, pol ->
   mkUtt (ifExt frag.ext (mkS pol (mkCl frag.subj frag.pred)));
.......

  mkBIText : Fragm -> Pol -> VV -> Utt =
    \frag, pol, vv ->
      mkUtt (ifExt frag.ext (mkS pol (mkCl frag.subj
      mkVP  vv frag.pred))));
  };
```

In the ABox concrete syntax the labels are defined according to the required types using either provided primitives or basic parts of speech, for example,

```
 AIntake = mkActivity (mkNP the_Quant intake_N);
 DSubmissionForm = mkNP the_Quant
                     (mkCN submission_N (mkNP form_N));
```

where the functions `mkN`, `mkCN`, `mkNP`, `mkVP`, `mkCl`, `mkS`, `mkUtt`, etc., and many more, are available in the GF resource grammar library for supporting and facilitating application grammar development.

The function `mkActivity`, used in linearising the label `AIntake`, converts the label "intake" to a propositional statement, where the verb to be used with activities expressed as nouns or noun phrases is always "to complete', i.e. "the intake is completed" as prescribed by the meta model in Figure 1. We return to the `mkActivity` function in Section 4.3 where the handling of label variants is discussed.

The final (customary) component in the suite of modules that constitute a GF application grammar is a dictionary of application specific lexical items that do not occur in the resource grammar lexicon and additional linguistic constructs that are language specific and/or are not included in the resource grammar, for example, verb nominalisation. Examples (showing the abstract as well as the concrete syntax) are as follows:

```
 oper
  intake_N:N;
  submission_N:N;
  form_N:N;
```

and

```
 oper
  intake_N = mkN "intake";
  submission_N = mkN "submission";
  form_N = mkN "form";
```

An example of a pre-condition triple and its verbalisation in English and Dutch by means of the GF grammar is follows:
(`Activity`, `Requires_Available`, `Artifact`, subtyped as `Document`), instantiated with the labels “intake” and “submission form”, is rendered as the pre-condition, containing the propositional statements “the intake may be completed” and “the submission form is available”, with modality added:

```
The intake may be completed , if the submission form
is available.
De inname kan worden afgerond , als het aanvraagformulier
beschikbaar is.
```

Other typical examples of linearisations are as follows:

Pre-conditions:

```
44b may be completed , if the document with details
is available.
44b kan worden afgerond , als het document met details
beschikbaar is.
```

Post-conditions:

```
If 44b has been completed ,
          the submission form has been created.
Als 44b afgerond is , is het aanvraagformulier aangemaakt.
```

Propositional statements (intermediate layer):

```
44b has been completed.
44b is afgerond.
```

In summary, the modularisation of the grammar was illustrated by showing how concepts and relations between them are introduced as categories and functions in the TBox abstract syntax and reused in the in the ABox abstract syntax to instantiate abstract instances. The concrete syntaxes were shown to provide linearisations, using different variants, for the instance labels, the propositional, and the conditional statements that consitute the pre- and post-conditions in the ontology.

4.3. Dealing with different variations of labels

In the previous section the focus was on modularisation as a mechanism to support efficient and effective ontology development. By separating the TBox and ABox information in different grammar modules the modeller is allowed to concentrate on application dependent information while generic business process modelling support is provided by the grammar. In this section we discuss the extent to which the BI grammar may allow for label variants and what kinds of variation is accommodated at present.

Basically, the grammar allows two broad kinds of labels. Firstly labels in the form of nouns or compound nouns (names or terms) are permitted, for example, “Intake” and “Equality principle”. Secondly, labels may take the form of a proposition such as “The Result is published” or other verb oriented style labels such as “Publishing the result” or “Publish the result”.

While allowing the modeller some freedom of choice in label selection, the verbalisation of label variants that refer to the same concept or relation in the ontology should be unique.

The following code fragment from the TBox concrete syntax shows how some of these design choices may be implemented:

```
 mkActivity = overload {
 mkActivity: N -> V2 -> NP -> Activity = \n,v,o -> lin Activity {
    noun = mkNP the_Quant (mkCN n (mkAdv of_Prep o));
    subj = o;
    vp = passiveVP v;
    hasVerb = True
 };
 mkActivity: V2 -> NP -> Activity = \v,o -> lin Activity {
    noun = nominalize (mkVPSlash v) o;
    subj = o;
    vp = passiveVP v;
    hasVerb = True
 };
 mkActivity: NP -> Activity = \o -> lin Activity {
    noun = o;
    subj = o;
    hasVerb = False
 }
};
```

In this way, the ABox concrete syntax may be used to create an object of type `Activity` by, for example, either providing an `NP` for simple cases like "Intake" or a `V2` or an object `NP` for labels such as "Publishing the result" and "Publish the result".

There are a number of other fields that are used in the English TBox concrete syntax that ensure an optimized natural language generation such as "The Results are published" or "Intake is completed if the results are published", but these details are just handled in the grammar, and the users of the ABox syntax do not need to be aware of them.

It should be noted that these choices of implementation are, to some extent, language specific since in labels of the latter kind the English gerund is used while in Dutch the infinitive form of the verb plus "van" is customary.

In the following example the label variant "publish the result" is rendered as in the sentences below:

Post-condition triple:

```
(Activity, Creates_Artifact, Document)
```

Linearisation:

```
If the result has been published , the submission form
is available.
Als het resultaat gepubliceerd is , is het
aanvraagformulier beschikbaar.
```

4.4. Multilingual aspects

In the GF grammar the multilingual correspondence of the bilingual system is via the ontological labels. For instance, "Publishing the result" in English corresponds to “Publiceren van het resultaat” in Dutch because they both map to the instance `APublishingOfResult`. However, word-wise there is no one-to-one correspondence, since this would not scale up when adding more languages and extending the ontology with more instances. From this perspective it resembles the *lemon* notion of multilingualism and supports the automated mapping from *lemon* to GF, as discussed in a subsequent section.

Moreover, since the TBox syntax mainly uses the resource library and aligneable primitives from there, it can be abstracted in a functor, so that the new languages can inherit it, at least partially. In this way, adding a new language would be a simpler process, since it would mainly require the writing of a new lexicon and translating the instances from the ABox concrete syntax. An investigation into this aspect forms part of future work.

4.5. Generating the instance grammars from *lemon*

We reiterate that the exploratory work presented in this paper focuses on three aspects of modular ontology verbalisation, as also illustrated in Figure 3.5. In section 2 we addressed the modularisation of the GF grammars, used for the verbalisation, in accordance with the modular ontologies. It was noted that the labour intensive component of the product is the creation of the ABox grammars (that contain instances and their labels) and that the flexibility in specifying labels is of strategic importance. Section 2, therefore, focused on label variants currently allowed by the GF grammar. In this section we now briefly turn our attention to the third aspect viz. an automated way of populating the ABox grammars by making use of *lemon*. The approach is based on the observation that the *lemon* entries contain all the essential information required to construct GF lexicon entries and linearisation rules for the instance labels in the ABox grammars.

We illustrate this by means of an example in which essential parts of the linguistic information, necessary to create an instance label, are shown. We consider the entry for the instance label “Sketch of the situation”, which is represented as a *lemon* `decomposition`:

```
 #Sketch of the situation
lemon:decomposition ( :sketch_component :prep_component
                :det_component :sit_component );

lemon:phrase_Root
[lemon:constituent :NP;
 lemon:edge [ lemon:constituent :NN; lemon:leaf:sketch_component];
 lemon:edge [ lemon:constituent :PP;
 lemon:edge [ lemon:constituent :P; lemon:leaf:prep_component];
 lemon:edge [ lemon:constituent NP;
 lemon:edge [ lemon:constituent :DET; lemon:leaf:det_component];
 lemon:edge [ lemon:constituent :NN; lemon:leaf:sit_component
         ]]]];

:sketch_component lemon:element sketch.
:sketch a lemon:Word .
:sketch isocat:partOfSpeech isocat:noun .

:prep_component lemon:element of.
:of a lemon:Word .
```

```
:of isocat:partOfSpeech isocat:preposition .

:det_component lemon:element the.
:the a lemon:Word .
:the isocat:partOfSpeech isocat:determiner.

:sit_component lemon:element situation.
:situation a lemon:Word .
:situation isocat:partOfSpeech isocat:noun .
```

It consists of a `phrase_Root` entry which describes the syntactic decomposition of the entry. The *lemon* entry is decomposed into a noun phrase, in turn decomposed into a noun, a preposition, a determiner and another noun. Each component in the *lemon* decomposition is comprised of `elements`, for example the preposition `of` a the noun `situation`.

In GF the same information is encoded as follows, making use of functions from the resource grammar library:
ABox abstract syntax:
`DSketchOfSituation : Document` where `Document` is a class in the ontology and a category in GF.
ABox concrete syntax:
`DSketchOfSituation = mkNP a_Quant (mkCN sketch_N (PrepNP of_Prep (mkNP the_Det situation_N)))` where `mkNP`, `mkCN` and `PrepNP` are functions from the resource grammar library.
Dictionary abstract syntax: `sketch_N : N` and `situation_N : N`
Dictionary concrete syntax: `sketch_N = mkN "sketch"` and `situation_N = mkN "situation"`.

The constituent structure (syntax tree) of the label “Sketch of the situation” is given in the first part of the *lemon* entry and by the GF ABox concrete syntax entry, while the lexical information is available in the second part of the lemon entry and the GF dictionary entry. An appropriate generalisation of this correspondence (mapping) will cover all possible *lemon* entry syntax trees and their GF equivalents. This will form the basis for an automated procedure to generate ABox concrete syntax entries for instance labels from *lemon* and is part of our future work. Furthermore, this approach is well suited to multilingual entries in *lemon* and their representation and verbalisation in GF, the investigation of which also forms part of future work.

5. Discussion and Future Work

In this paper we demonstrated how GF and *lemon* can be combined to provide multilingual verbalisation in a specific class of problems, where ontologies are modularised into ontologies containing consensus based classes and (multiple) instance ontologies. In these scenarios, the instances are typically not created by knowledge engineers and are often based on information stores other than ontologies.

We showed an approach to grammar development that leads to grammars being modular along the same boundaries as the ontologies, with the grammar modules also having the same dependency structure as the ontology parts.

Additionally, we introduced mechanisms in the TBox grammar that deal automatically with the different styles of label choosing that are encountered in practice. This is particularly relevant in cases where different disciplines are involved in creating the instances and in multilingual cases where concepts may have a name in one language and can only be referenced by describing them in another language.

By generating the instance grammars from the ontology labels encoded in *lemon*, the need for GF engineering at the instance level is reduced. This is crucial for the adoption of this mechanism, as the instance data either already exists in databases or is created by people who may not be expected to engineer GF representations of the labels they choose.

In terms of multilingualism this approach supports and suggests two ways of verbalisation. In cases where translations of the labels are available inside *lemon* lexica, multiple ABox syntaxes can be generated for the different languages. This approach has the benefit that labels may differ quite significantly across languages, but obviously requires translation of the ontology. Alternatively, translation could be performed at the GF grammar level if the label styles align across languages. In cases where aligned dictionary grammars are available, this approach could prove particularly efficient.

Future work also includes the extension of this approach to include different levels of paraphrasing and advanced sentence planning in order to achieve improved fluency in and across sentences. Finally, we envisage investigating label choosing practices as encountered amongst professionals in the different fields that may benefit from a framework as described in this paper. The importance of robustness under lexical variance as found in real world applications suggests an in depth study of label style variation and its impact on ontology verbalisation.

Acknowledgements

The authors wish to gratefully acknowledge the support for this work by the European Commission (EC). This work was partially funded by the EC within the EU FP7 Multilingual Ontologies for Networked Knowledge (MONNET) Project under Grant Agreement No. 248458 and the MOLTO Project under Grant Agreement No. 247914.

Bibliography

[1] Heller, R., Teeseling, F.: Knowledge applications for life events: How the dutch government informs the public about rights and duties in The Netherlands. In: Proceedings of the 6th European Semantic Web Conference on The Semantic Web: Research and Applications. ESWC 2009 Heraklion, Berlin, Heidelberg, Springer-Verlag (2009) 846–850

[2] Van Grondelle, J.C., Gülpers, M.: Specifying flexible business processes using pre and post conditions. In: PoEM. Volume 92 of Lecture Notes in Business Information Processing., Springer (2011) 38–51

[3] Buitelaar, P., Cimiano, P., Haase, P., Sintek, M.: Towards linguistically grounded ontologies. In: The Semantic Web: Research and Applications. (2009) 111–125

[4] Montiel-Ponsoda, E., de Cea, G., Gómez-Pérez, A., Peters, W.: Modelling multilinguality in ontologies. In: Proceedings of the 21st International Conference on Computational Linguistics (COLING). (2008)

[5] Francopoulo, G., George, M., Calzolari, N., Monachini, M., Bel, N., Pet, M., Soria, C.: Lexical markup framework (LMF). In: Proceedings of the 2006 International Conference on Language Resource and Evaluation (LREC). (2006)

[6] Kemps-Snijders, M., Windhouwer, M., Wittenburg, P., Wright, S.: ISOcat: Corralling data categories in the wild. In: Proceedings of the 2008 International Conference on Language Resource and Evaluation (LREC). (2008)

[7] Fuchs, N.E., Kaljurand, K., Kuhn, T.: Attempto Controlled English for Knowledge Representation. In Baroglio, C., Bonatti, P.A., Małuszyński, J., Marchiori, M., Polleres, A., Schaffert, S., eds.: Reasoning Web, Fourth International Summer School 2008. Number 5224 in Lecture Notes in Computer Science, Springer (2008) 104–124

[8] Power, R., Scott, D., Evans, R.: What you see is what you meant: direct knowledge editings with natural language feedback. In Prade, H., ed.: 13th European Conference on Artificial Intelligence (ECAI'98). John Wiley and Sons, Chichester, England (1998) 677–681

[9] Angelov, K.A., Enache, R.: Typeful ontologies with direct multilingual verbalization. In: Controlled Natural Languages Workshop (CNL 2010), Marettimo, Italy (2011)

[10] Dannélls, D., Enache, R., Damova, M., Chechev, M.: Multilingual online generation from semantic web ontologies. In: WWW2012. EU projects track, Lyon, France (04/2012 2012)

[11] Ranta, A.: Grammatical Framework: Programming with Multilingual Grammars. CSLI Publications, Stanford (2011) ISBN-10: 1-57586-626-9 (Paper), 1-57586-627-7 (Cloth).

3 Multilingual Grammar for Museum Object Descriptions

Dana Dannélls   Aarne Ranta   Ramona Enache

1. Introduction

During the last decade, there has been a shift from developing natural language systems to developing domain independent applications that are capable of producing natural language descriptions directly from Web ontologies ([1, 2, 3]). Many of the existing systems employ verbalisation methods to present the content of the ontology structure to particular subset of users, almost exclusively in English language. The problem with the existing verbalization methods is that they assume each ontology statement is mappable to natural language, which is not always the case, in particular for languages other than English. Moreover, the task of producing adequate natural language descriptions by employing verbalization methods is very difficult if not impossible.

We have developed a grammar application in GF that applies natural language generation techniques to generate multilingual descriptions about museum artefacts, starting from the CIDOC-CRM ontology. We opted for a layered representation of the natural language generation system, where the ontology represents the first layer, which provides the semantic structure and the instances that we will verbalize. A second layer is the natural language generation grammar, which defines the way in which we combine ontology data in order to create text. Due to the multilingual context, the generation grammar aims to be general enough to allow the same assertions to be expressed in all the languages.

At our knowledge, this is the first attempt to develop a prototype that exploits natural language generation techniques such as applying discourse strategies to generate multilingual descriptions in at least five languages from semantic web content, in particular from OWL ontology standards such as CIDOC-CRM.

1.1. The purpose of this document

Work package 8 foresees several tasks:

- integrate data from the Gothenburg City Museum (GCM) with the Conceptual Reference Model CIDOC-CRM ontology standard
- build a prototype of a cross-language retrieval and representation system to be tested with objects in the museum
- implement a multilingual domain application grammar that is capable of generating well-formed object descriptions from CIDOC-CRM

The purpose of this document is to describe the developed domain specific application grammar we have been implementing. The grammar presented here allows to generate well-formed multilingual natural language descriptions about museum artefacts with the aim of empowering users who wish to access cultural heritage information through different computing devices.

1.2. Use cases

Some of the the key benefits of the grammar application that is being developed in this project are:

- acceptability and usability by other grammarians
- acceptability and usability by the industrial environment
- possibilities of reuse by other applications

2. The grammar development workflow

To accomplish the goals of this workpackge we found it necessary to develop a specific ontology model to store and present detailed information about specific artefacts that are available in the Gothenburg City Museum database. The focus was on painting objects, as described in [4]. The main model for the painting ontology development was the CIDOC–CRM, which provided a detailed conceptual reference as a starting point for the ontology design. Using the painting ontology as a point of departure we were able to develop two different generations modules: one that applies direct verbalization (section 3) and another that exploits discourse pattern generation (section 3).

The painting ontology containing data from the museum has been integrated into the Reason-able View of Linked Data for Cultural Heritage, [5] which is also part of workpackage 4, see [6]. Basically, we integrated the paintings and the other objects in one single repository, providing an interoperable framework that is to used retrieve information about museum objects, [7].[15] From this Web repository it is possible to retrieve a set of RDF triples that provides a formal description about the museum object, including the name of the object, the painter who created it, the year of painting, the material that was used to execute the object, the current location of the object, how it was acquired, its value and other semantic information that is given both in the form of Id's and canned text, such as about the content or historical knowledge about an object.

The retrieved information forms the input to the domain grammar application from which we are able to generate multilingual descriptions as described below (section 3).

3. The grammar

3.1. Ontology verbalization

An important component in the natural language generation system is the layer that connects the ontology with the generation grammar. On one step we get it by importing the Painting ontology in GF, as it contains the instances of most fields that describe the paintings. However, the painting names and painters need to be imported directly from the database.

Unlike previous experiments with representing SUMO (Suggested Upper-Merged Ontology) in GF ([8, 9]), the representation of the painting ontology doesn't use dependent types, since simple types are enough to describe the concepts and relations and model the hierarchy in a simplified way that suffices the needs of the generation grammar.

For example, the classes `OilPainting` and `Painting`, along with the inheritance relation between them are represented as GF categories:

[15] `http://museum.ontotext.com`

```
cat Painting ;
cat OilPainting ;

fun OilPainting_Painting : OilPainting -> Painting ;
```

where the inheritance relation is modeled as a coercion function between the two types. In the concrete syntaxes, all coercion functions will be linearized as the identity, since they shouldn't be visible in verbalization.

The instances are represented as GF instances of the GF mappings of their ontology classes. For example:

```
fun AerosolPaint : Material ;
```

The advantage is that when porting the Painting grammar to a new language, one could linearize certain categories to different parts of speech, depending on how the concepts are expressed in the language.

Developing a concrete syntax for the ontology grammar is quite straightforward for English, but could be a challenge for other languages. The examples from the current grammar were translated manually, because the number of paintings that we described was very small.

However, in the future, we plan to build the lexicon multilingual lexicon automatically in 2 steps:

- port the painting ontology (at least the classes and instances) in another language by using the lexical translation tools from WP3, which would ensure a semantics-preserving mapping of the lexical units by using multilingual resources, such as DBPedia.
- as we assume that the painter name and painting name are the only attributes that might not be found in the Painting Ontology, we add them on the fly, for the painting that we want to verbalize and not for the whole database. The reason is that they will be represented as proper names, whereas the other features could be mapped to different parts of speech depending on the language. Moreover, the names of the painting and its creator can be translated using the same resources as in the previous step, before added to the concrete syntax.

The automation of lexicon acquisition will be possible as soon as there will be an integration of the WP3 tools to the grammar development ones.

3.2. Discourse pattern generation

In the first deliverable of this workpackage, e.g. [10] we presented a number of features and discourse patterns that we learned by analyzing a large set of well-formed object descriptions. Below we summarize some of the discourse patterns the analysis reveled.

- painting paintingtype painter
- painting painter year
- painting museum painter size
- painting painter represented museum
- painting material year painter

- painting painter year museum colour size

The initial idea was to follow these features and patterns when generating multilingual descriptions.

We isolated a number of attributes of paintings that we decided to focus on in the prototype development. We agreed that each description should convey at least 3 main features of a painting. This assumption enable us to define a default representation in GF and thereby always produce a description about an object. These three features are:

1. the name of the painting — `Painting`
2. the name of the painter — `Painter`
3. the type of painting (for example, *oil painting*) — `PaintingType`

and 5 optional ones that allow us to generate more detailed descriptions:

1. the colours used in the painting — `Colour`
2. the size of the painting — `Size`
3. the material of the painting — `Material`
4. the year when the painting was created — `Year`
5. the museum where the painting is currently displayed — `Museum`

The difference between the 2 categories of features, is that we don't expect to find all the optional categories in all the painting descriptions from the database, but we want to have only one representation for the instances of paintings from the ontology which we will verbalize and only one verbalization function in the grammar, which would be easier to port to new languages.

The solution is to wrap the categories representing optional features in a number of categories inspired by the `Maybe` type from Haskell, which retains the presence/absence of the feature and in case the feature is present, its value.

Hence, we can represent the text generation as one function taking all the features as arguments:

```
MkGenText :
   Painting -> Painter -> PaintingType ->
   OptColour -> OptSize -> OptMaterial ->
   OptYear -> OptMuseum -> GenText ;
```

where `OptColour`, `OptSize`, `OptMaterial`, `OptYear` and `OptMuseum` are the wrapper categories. The concrete representations of `MkGenText` opt for different text patterns, depending on the presence of the optional parameters.

For the cases, where the name of the painter or the painting is missing, we created a number of instances that indicate the absence of the features:

- `NoPainting`
- `NoPainter`

In this way, we get the most detailed description that one can form with the available features. This differentiates the current approach from the previous one, described in [11], where each text-generation pattern is represented separately, so that the user can choose the sort of descriptions that she wants.

The reason why the current grammar only retains the most informative description is that the implementation of the patterns contains redundancies and entails more effort from the grammar writer. On the other hand, the usage of the patterns provides more options for natural language generation, because we can control the amount of information that we describe.

The grammar structures is ported to 5 languages: English, French, Italian, Finnish and Swedish.

3.3. General design issues

The current approach reduces the use of dependent types to a minimum, in order to keep the grammar simpler and make it easier for users to port it to new languages. The only use of dependent types in the current grammar is for representing the painting structure. This is necessary for enforcing a predefined structure on the generated text. For example the definition (def) below enforces the generated text (`MkGenText`) to bear the eight features.

```
fun
  vtext2gtext : VerifiedText -> GenText ;
def
  vtext2gtext (MkVerifiedText pg pr pt cr se ml yr mm _) =
MkGenText pg pr pt cr se ml yr mm ;
```

This makes it possible to control the natural language generation; keeping the descriptions consistent with the ontology. For example, the painting `GSM940042Obj` is represented as following:

```
GSM940042ObjPainting : CompletePainting
 GSM940042Obj MiniaturePortrait  JKFViertel (MkYear (YInt 1814))
 (MkMuseum GoteborgsCityMuseum)  (MkColour Grey)
   (MkSize (SIntInt 349 776)) (MkMaterial Wood) ;
```

Thus, when a description is generated, we get all the information that is associated to it in the Painting ontology.

This is however just an optional feature, because one can preserve the semantic consistency by adding another layer, exterior to the grammar, that calls the text-generation function with the proper arguments. But the possibility of having it inside the grammar is more attractive, since it shows the power of GF.

In any case, the dependent types don't bring about any change when porting the grammar to new languages, as they are just a way to group together features that used for generation.

The previous version of the grammar featured a more extensive use of dependent types including the type used to represent a painting and all its attributes that is preserved in the current implementation. Moreover, the previous grammar used semantic definitions for functional programming-inspired pattern-matching on the relevant features that each pattern needs to use. This entails that the case analysis is done just once – in the abstract syntax and doesn't need to be repeated for each language. The current implementation implements it in the concrete syntax, which could lead to code repetition across the languages.

In both cases, the dependent types don't need to be implemented in the concrete syntax, and the less-experienced user won't need to manipulate them in order to port the grammar to a new language. The current grammar is even more user-friendly, as the dependent types are almost seamless, and the users don't need to use them, if the grammar is used within a runtime system that doesn't provide support for them, such as Java or C.

When adapting the grammar to a new language, the only thing the grammarian needs to create about is one function, `MkGenText` and the lexicon.

```
MkGenText painting painter paintingtype colour
                              size material year museum =
   let
    s1 : Text = mkText (mkS pastTense
     (mkCl painting
         (mkVP (mkVP (mkVP (passiveVP paint_V2) material.s)
     (SyntaxEng.mkAdv by8agent_Prep (mkNP painter))) year.s))) ;

     sizeS : S = mkS (mkCl it_NP size.s) ;
     colourS : S = mkS (mkCl it_NP
                        (mkVP (passiveVP paint_V2) colour.s)) ;

     s2 : Text = case <size.isGiven, colour.isGiven> of {
       <True,True>  => mkText (mkS and_Conj sizeS colourS) ;
       <True,False> => mkText sizeS ;
       <False,True> => mkText colourS ;
       _            => emptyText
      } ;

     s3 : Text = case museum.isGiven of {
       True => mkText (mkS
                (mkCl (mkNP this_Det paintingtype)
                   (mkVP (passiveVP display_V2) museum.s))) ;
       _ => emptyText
      } ;
   in
      mkText s1 (mkText s2 s3) ;
```

3.4. Generation Results

The application outputs consist of short, well-formed natural language descriptions in five languages. On the syntactic level, sentence structures contain passive constructions, aggregations and generation of referring expressions. Some examples are given below.

```
Painting: MkGenText GSM940051Obj BrynolfWennerberg
PortraitPainting NoColour NoSize (MkMaterial Wood)
(MkYear (YInt 1889)) (MkMuseum GoteborgsCityMuseum)
```

- PaintingEng: *Hisingen was painted on wood by Brynolf Wennerberg in 1889. This portrait is displayed at the City Museum of Gothenburg.*
- PaintingFin: *Maalauksen Hisingen on maalannut Brynolf Wennerberg puulle vuonna 1889. Tämä muotokuva on esillä Göteborgin kaupunginmuseossa.*

- PaintingFre: *Le tableau Hisingen a été peint sur bois par Brynolf Wennerberg en 1889. Ce portrait est exposé dans le musée municipal de Göteborg.*
- PaintingIta: *Il quadro Hisingen è stato dipinto su legno da Brynolf Wennerberg nel 1889. Questo ritratto è esposto nel museo municipale di Goteburgo.*
- PaintingSwe: *Hisingen målades på trä av Brynolf Wennerberg år 1889. Den här porträttmålningen är utställd på Göteborgs stadsmuseum.*

```
Painting: MkGenText GSM980019Obj AnnaLindskog OilPainting
(MkColour Black) (MkSize (SIntInt 435 365)) (MkMaterial Canvas)
(MkYear (YInt 1885)) (MkMuseum GoteborgsCityMuseum)
```

- PaintingEng: *The girl was painted on canvas by Anna Lindskog in 1885. It is of size 435 by 365 and it is painted in black. This oil painting is displayed at the City Museum of Gothenburg.*
- PaintingFin: *Maalauksen Flickan on maalannut Anna Lindskog kankaalle vuonna 1885. Se on kokoa 435 kertaa 365 ja se on maalattu mustalla. Tämä öljymaalaus on esillä Göteborgin kaupunginmuseossa.*
- PaintingFre: *Le tableau Flickan a été peint sur toile par Anna Lindskog en 1885. Il est de taille 435 sur 365 et il est peint en noir. Cette peinture à l' huile est exposée dans le musée municipal de Göteborg.*
- PaintingIta: *Il quadro Flickan è stato dipinto su tela da Anna Lindskog nel 1885. Misura 435 per 365 ed è dipinto in nero. Questo dipinto ad olio è esposto nel museo municipale di Goteburgo.*
- PaintingSwe: *Flickan målades på duk av Anna Lindskog år 1885. Den är av storlek 435 gånger 365 och den är målad i svart. Den här oljemålningen är utställd på Göteborgs stadsmuseum.*

4. Conclusion and future work

The presented grammar consists of one discourse patten that contains a small amount of features a painting object description should convey. From this pattern we are able to generate different descriptions depending on the information that is available about this object. The main advantage of the grammar is that it can be ported the other languages very easily, by only modifying one pattern and changing the lexical entities.

One of the functionalities the current grammar does not cover is the ability to combine different features across different sentences. However, the simplicity of the grammar makes the working effort of adding new patterns for distributing features differently across sentences minor. Moreover, with only small modifications, such as selecting other types of referring expressions, we are able to increase the fluency of the output results depending on the language.

In the nearest future we intend evaluate the generation results and port the grammar to 10 additional languages. We plan to increase the coverage of grammar and the lexicon for at least 5 languages.

It will be interesting to test how the grammar performs with different objects and on other domains. Another possible future direction is to generate texts in different formats that can be adaptable to different user needs, for example by modifying the style of the generated texts in terms of syntactic variations.

Bibliography

[1] Schwitter, R., Tilbrook, M.: Controlled Natural Language meets the Semantic Web. In: Proceedings of the Australasian Language Technology Workshop, Macquarie University (2004) 55–62

[2] Fuchs, N.E., Kaljurand, K., Kuhn, T.: Attempto Controlled English for Knowledge Representation. In Baroglio, C., Bonatti, P.A., Małuszyński, J., Marchiori, M., Polleres, A., Schaffert, S., eds.: Reasoning Web, Fourth International Summer School 2008. Number 5224 in Lecture Notes in Computer Science, Springer (2008) 104–124

[3] Williams, S., Third, A., Power, R.: Levels of organisation in ontology verbalisation. In: Proceedings of the 13th European Workshop on Natural Language Generation (ENLG), Nancy, France (September 2011) 158–163

[4] Dannélls, D.: The painting ontology. Journal of applied ontologies (2011) Submitted.

[5] Damova, M., Dannélls, D.: Reason-able view of linked data for cultural heritage. In: Proceedings of the third International Conference on Software, Services and Semantic Technologies (S3T). (2011)

[6] Damova, M.: Data Models and Alignment. (May 2011) Deliverable 4.2. MOLTO FP7-ICT-247914.

[7] Dannélls, D., Damova, M., Enache, R., Chechev, M.: A Framework for Improved Access to Museum Databases in the Semantic Web. In: Language Technologies for Digital Humanities and Cultural Heritage. (2011)

[8] Enache, R.: Reasoning and language generation in the sumo ontology. Master's thesis, Chalmers University of Technology (2009)

[9] Enache, R., Angelov, K.: Typeful ontologies with direct multilingual verbalization. Workshop on Controlled Natural Languages (CNL) 2010 (2010)

[10] Dannélls, D.: D.8.1 Ontology and corpus study of the cultural heritage domain. (2011) Deliverable of EU Project MOLTO Multilingual Online Translation.

[11] Dannélls, D., Damova, M., Enache, R., Chechev, M.: Multilingual online generation from semantic web ontologies. In: Proceedings of the World Wide Web Conference (WWW2012), Lyon, France (2012)

Chapter 4

Bootstrapping Grammars from External Sources

1 Controlled Language for Everyday Use: the MOLTO Phrasebook

Aarne Ranta   Ramona Enache   Grégoire Détrez

Abstract: Controlled languages are usually targeted for technical domains and designed to be unambiguous. This paper presents a controlled language whose domain is touristic phrases, aimed to be usable by anyone without prior training. Despite its informal nature, the language of phrases has a firm notion of semantics, defining the correctness of translations. However, this semantics is formulated in terms of context and situation rather than by logical formulas. Moreover, the language is often ambiguous, and the translation may depend on resolving the ambiguity. This paper shows how to formalize a semantics for tourist phrases and implement it in 15 languages, how to deal with the ambiguities, and how to make the system available for layman users on the web and on mobile phones. While a useful application as such, the Phrasebook also paves the way for an extended notion of controlled language, and the techniques are aimed to be general enough to support many such extensions.

1. Introduction

Controlled languages are typically designed for use on technical domains. Their users are experts such as aircraft engineers [1], medical doctors [2], and topographers [3]. The language is typically a natural-language image of a formal system, such as predicate logic in [4] or OWL (Web Ontology Language) [5] in [6]. The purpose of these controlled languages is to support knowledge representation, reasoning, and mechanical checking of correctness; the main point of using a natural language fragment rather than a formalism is to have a notation that is readable without special training. When there is no underlying formalism, as in [1], the purpose is to eliminate the ambiguity, vagueness, and unclarity of uncontrolled natural language.

However, the notion of controlled language can be given a wider interpretation: it can be just *any* fragment of natural language specified with a formal set of rules. Actually it can be seen as the technological counterpart of Wittgenstein's philosophical notion of **language games** [7], which are systems of rules specifying how language is used for performing different tasks. In the tradition represented by Wittgenstein and his followers, language games are the very essence of language: they should *not* be seen as mere fragments of an underlying total system, but as the building blocks that actually constitute the thing called language. Very little can be rigorously said about natural language as a whole, whereas these limited fragments are units that (at least in many cases) permit a formal description and—consequently—a computer implementation.

When we start looking at language from the language game point of view, we suddenly begin to see "formal systems" everywhere. One of the most basic ones is the social game of greetings and politeness phrases. For instance, when I ask for something, I attach the word *please*. When you hand it over to me, you say *here we are*, to which I should say *thank you*, and you can conclude by replying *you're welcome*.

These four phrases get their precise meanings in the context of this game. Actually, each of them could be used in some other context and mean something different. This is seen clearly when we look at their translations. Here is a simple dialogue in three languages:

English	**Swedish**	**German**
A beer please.	*En öl tack.*	*Ein Bier bitte.*
Here we are.	*Var så god.*	*Bitte.*
Thank you.	*Tack.*	*Danke.*
You're welcome.	*Var så god.*	*Bitte.*

English makes most distinctions here, by using a different phrase for each of the four moves of the game. Swedish uses *tack* for both asking and thanking. German uses different phrases for these two, but the word *bitte* (literally, "I request") is otherwise used for everything! Nevertheless, there's no problem in translating Swedish and German phrases to English, *as soon as we know what move they express in the language game*.

Of course, it is just a coincidence that English has unambiguous phrases for all language game moves here. English, and all other languages, are full of ambiguities, if we look only at the syntax without context. This is not just a feature of everyday language but even of mathematics, as convincingly shown in [8]. But the ambiguities are almost always easily resolved by looking at the context of use.

An ambiguity specific to English is generated from the word *you*. It has two translations in Swedish (the familiar singular *du*, the plural or formal singular *ni*), three in German (the familiar singular *du*, the familiar plural *ihr*, and the formal *Sie*), and up to eight in languages like Spanish (singular/plural, familiar/formal, masculine/feminine). For instance, the English phrase *are you German* has eight translations in Spanish. The translation is determined by the context of use—basically, by the addressee.

The "language game" of social phrases is not only a philosophical experiment, but also a lucrative business. Phrasebooks like Berlitz and Lonely Planet are still sold in millions of copies, although electronic phrasebooks running on mobile phones are taking more and more of the market share. A typical electronic phrasebook is just a digital version of the printed book: a collection of phrases that can be looked up either by typing search strings or by browsing in hierarchic menus. A particularly smart example is the Chinese iPhone application YoChina[1], which puts each phrase into a context and also shows a set of responses from which the interlocutor can choose.

Even the most sophisticated commercial phrasebooks are still just collections of **canned phrases**: fixed strings, which, even though there might be thousands of them, don't cover all possible combinations of the concepts involved. A different approach can be taken by using **machine translation**; thus Google Translate [2] is available as a mobile phone application that actually translates each individual phrase separately. While this is the most natural and powerful approach to the problem, it still has open issues. The first issue is quality: even though Google Translate often does a good job, it can just as easily produce something totally wrong, and this can lead to embarrassing situations if used in a social context of communication (mostly resolved by a good laugh, of course). In particular, Google Translate is based on a generic, statistical language model which cannot make distinctions like the ones needed for the different uses of German *bitte*. The second issue with Google Translate as used by a traveller far

[1]http://www.yocoy.com

[2]http://translate.google.com

away from home is the cost of mobile data transfer. It may just be too expensive to use the service.

In this paper, we will introduce a controlled language translator approach to tourist phrasebooks. We will show a **formal semantic model**, which unambiguously specifies an infinite class of phrases. Then we will show how the semantic model is translated to phrases in 15 natural languages. The translations are reversible, which means that the phrasebook can both generate natural language from the formal semantics and interpret it in the formal semantics. The combination of generation and interpretation is translation; our phrasebook is able to translate equally well with all of the 14*15=210 language pairs. The translator runs as an off-line application on Android mobile phones and can be downloaded free of charge from Android Market[3]. The phrasebook is also available as a web application[4].

Figure 1 (left) shows the web interface to the phrasebook. The user has constructed the English sentence *how far is the Russian restaurant*, which the system has translated to the other 14 languages. The construction is carried out by a **predictive parser** [9], which predicts the set of possible next words at each point. The input can be made by typing text (in the white slot on the right) or by clicking at one of the rectangles showing a word. The possible continuations here are *?* (to terminate the phrase), *by* (as in *by tram*), and *from* (as in *from the hotel*). As soon as there is enough input to translate, the translations are shown. Figure 1 (right) shows the Android mobile application. For size reasons, the application shows only one target language at a time. As a bonus, it has speech synthesizer output for some languages.

Touristic phrases are a rich domain, and one could easily spend a lifetime on building, refining, and extending an electronic phrasebook. What we want to show in this paper is a technology that gives maximal support to this work. The technology is based on GF (Grammatical Framework, [10]), which is a grammar formalism designed for supporting multilingual grammars of controlled languages. In addition to a programming language, GF provides RGL (Resource Grammar Library, [11]), which encapsulates the low-level linguistic knowledge of morphology and syntax that is needed when building high-quality translation systems.

In addition to the grammar engineering tools, GF has a set of tools supporting runtime applications. These include libraries for web servers and clients [12] and, most importantly for the current purpose, a Java-based run-time system for Android phones.

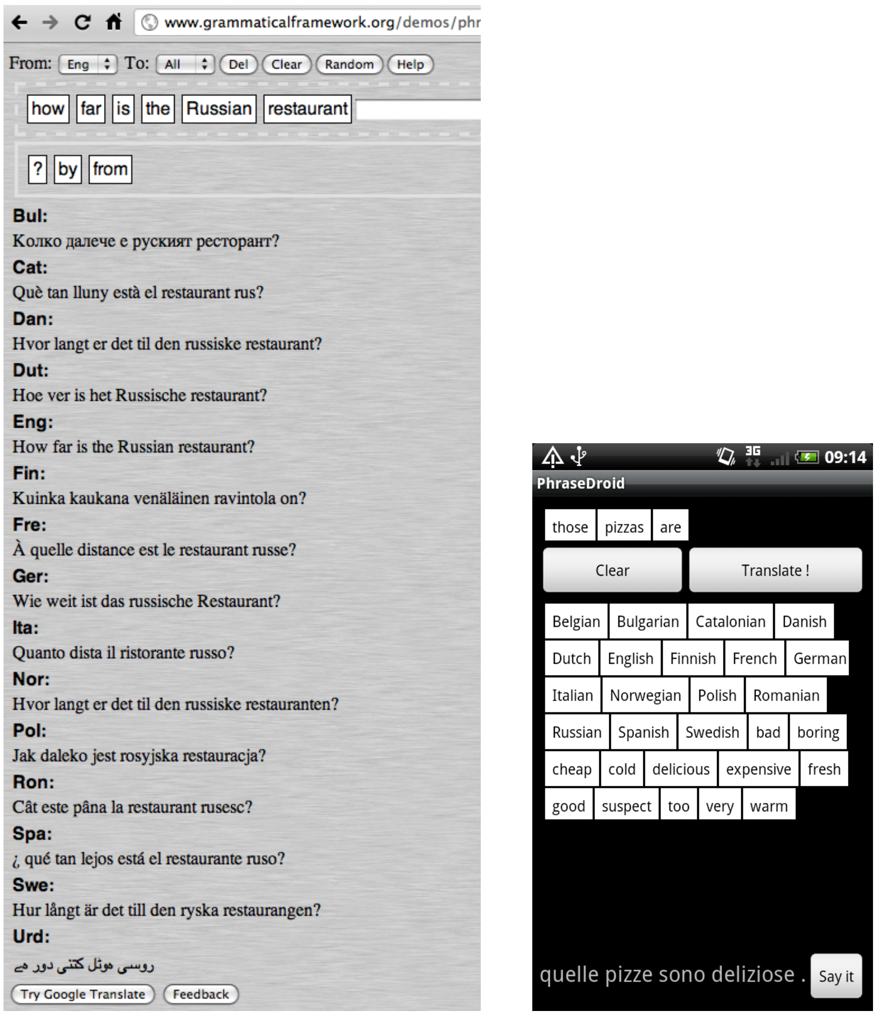
Thus, at the same time as the phrasebook is a practical help for tourists, it is a showcase for a powerful general technology. This technology is being developed in the European MOLTO project [5]). In addition to using GF, MOLTO explores ways to use statistical translation models to help the construction and improve the coverage of grammar-based systems. The MOLTO Phrasebook is a first experiment of this: some of the languages involved were implemented by programmers not knowing the language at all, but using a statistical model to bootstrap the grammar and a native-speaker informant to evaluate it. This was developed into a general method that will be usable for any further project of building multilingual controlled language systems.

The structure of the paper is as follows: Section 2 specifies the coverage of the MOLTO phrasebook by giving an overview of its semantic model. Section 3 shows examples of how the different languages are implemented by using GF and RGL. Section 4 shows how ambiguities are displayed to users by means of disambiguation grammars.

[3]https://market.android.com/details?id=org.grammaticalframework.android.apps.phrasedroid

[4]http://www.grammaticalframework.org/demos/phrasebook/

[5]Multilingual On-Line Translation, http://www.molto-project.eu


www.grammaticalframework.org/demos/phr
From: Eng To: All Del Clear Random Help
how far is the Russian restaurant
? by from

Bul:
Колко далече е руският ресторант?
Cat:
Què tan lluny està el restaurant rus?
Dan:
Hvor langt er det til den russiske restaurant?
Dut:
Hoe ver is het Russische restaurant?
Eng:
How far is the Russian restaurant?
Fin:
Kuinka kaukana venäläinen ravintola on?
Fre:
À quelle distance est le restaurant russe?
Ger:
Wie weit ist das russische Restaurant?
Ita:
Quanto dista il ristorante russo?
Nor:
Hvor langt er det til den russiske restauranten?
Pol:
Jak daleko jest rosyjska restauracja?
Ron:
Cât este până la restaurant rusesc?
Spa:
¿ qué tan lejos está el restaurante ruso?
Swe:
Hur långt är det till den ryska restaurangen?
Urd:
روسی ہوٹل کتنی دور ہے

Try Google Translate
Feedback
09:14
PhraseDroid
those pizzas are
Clear
Translate !
Belgian Bulgarian Catalonian Danish
Dutch English Finnish French German
Italian Norwegian Polish Romanian
Russian Spanish Swedish bad boring
cheap cold delicious expensive fresh
good suspect too very warm
Say it

quelle pizze sono deliziose .

Figure 4.1: The MOLTO Phrasebook as a web application (left) and as an Android mobile application (right).

Section 5 introduces the method of example-based grammar writing using statistical models and human informants. Section 6 explains the Java run-time system of GF and the architecture of the mobile Android application. Section 7 presents some results from evaluation, and Section 8 concludes.

2. The Semantic Model

In GF, a semantic model is called an **abstract syntax**. It is defined by giving a set of **categories** (keyword `cat`) and a set of **functions** (keyword `fun`), which together define the notion of **well-typed trees**. For instance, the phrases in the beer-ordering dialogue above can be given the following abstract syntax:

```
cat
  Phrase ; Item
fun
  GivePlease    : Item -> Phrase
  HereWeAre     : Phrase
  ThankYou      : Phrase
  YouAreWelcome : Phrase
  ABeer         : Item
```

The model could be made more precise by specifying that these phrases must appear in a certain order to constitute a valid dialogue. But for the purposes of a phrasebook, it is enough to specify uniquely each type of phrase by giving it a function name. All functions in this simple model are actually **constants**, i.e. they take no arguments—except `GivePlease`, which takes an `Item` as its argument.

The linguistic realizations of the semantic model are specified by a **concrete syntax**, which tells how trees formed in abstract syntax are **linearized** into strings in different languages. We will return to the details of linearization in Section 3; just to give an example, the following linearization rules (`lin`) could be given for German:

```
lin
  GivePlease item = item ++ "bitte"
  HereWeAre       = "bitte"
  ThankYou        = "Danke"
  YouAreWelcome   = "bitte"
  ABeer           = "ein Bier"
```

All linearization rules in GF can be also used for **parsing**, that is, converting strings to trees. This tiny example clearly shows that parsing can be **ambiguous**, that is, return more than one tree. The everyday counterpart of parsing ambiguity is shown by the situation where someone asks: “What is *bitte* in English?” The correct answer is that it depends on context: it may mean *please* or *here we are* or *you are welcome*.

In the full MOLTO Phrasebook, none of the 15 languages is unambiguous. What we need is an abstract syntax that formalizes all possible distinctions, so that each abstract syntax tree has a unique linearization in every language. Now, capturing all relevant distinctions in 15 languages might sound like a hopeless task, but in fact the semantic model scaled up quite well when the grammar was extended language by language. After a careful initial design (with awareness of what typically happens in languages), almost no changes were needed in the abstract syntax when new languages were added.

category	**explanation**	**example**
Phrase	complete phrase, unit of translation	*Where are you?*
Greeting	idiomatic greeting	*hello*
Sentence	declarative sentence	*I am in the bar*
Question	question, either yes/no or wh	*where are you*
Proposition	can be used as sentence or question	*this pizza is good*
Object	object of wanting, ordering, etc	*two pizzas and a beer*
Item	a single entity	*this pizza*
Kind	a type of an item	*pizza*
Quality	qualification of an item	*very good*
Place	location	*the bar*
PlaceKind	type of location	*bar*
Person	agent wanting or doing something	*you*
Action	proposition about a Person	*you are here*
Nationality	complex of language, property, country	*Swedish, Sweden*
Language	language (can be without nationality)	*Flemish*
Citizenship	property (can be without language)	*Belgian*
Country	country (can be without language)	*Belgium*
Currency	currency	*Swedish crown*
Number	number expression in words	*two hundred and five*
Price	price (number + currency)	*sixty-five dollars*

Table 4.1: Some of the 42 categories of the Phrasebook.

Printed phrasebooks have canned, static phrases, whereas a digital grammar-based phrasebook has rules for forming phrases from smaller expressions. The MOLTO Phrasebook has 42 categories and 290 functions. Of the functions, 130 take arguments and 160 are constants, which means that they are either lexical items or canned phrases. What is a lexical item in one language can be a multiword phrase in another language, as shown for instance by *bitte* vs. *here we are*. The number of phrases is infinite because of recursion, but on the reasonable level of tree depth 3, the Phrasebook has 484,938 abstract syntax trees of phrases.

The full code of the phrasebook, with some documentation, can be found on-line[6]. We will here show a sample of the coverage, and then focus on a few interesting problems created by some of the constructions. Table 1 gives some of the categories, and Table 2 some of the combination functions.

For a detailed sample, let us focus on the category `Action`, and the ways of asking persons for information about themselves and what they do. The complete `Phrase` corresponding to the question

Are you Swedish?

has the tree (in GF's LISP-like notation)

```
PQuestion (QProp (PropAction
  (ACitizen YouFamMale (CitiNat Swedish))))
```

This tree is formed by the functions

[6]http://www.grammaticalframework.org/examples/phrasebook/doc-phrasebook.html

arguments	**value**	**examples**
Number, Kind	Object	*five pizzas*
Quality, Kind	Kind	*Italian pizza*
Kind	Item	*this pizza*, *the pizzas*
PlaceKind	Place	*the bar*, *a bar*
Proposition	Sentence	*the bar is open*, *the bar isn't open*
Proposition	Question	*is the bar open*
Action	Proposition	*I speak Polish*
Person, Object	Action	*you have beer*, *you have no beer*
Person, Citizenship	Action	*you are German*
Person, Place	Action	*you are in the bar*
Person, Sentence	Action	*you know that I am in the bar*
Person, Person	Action	*you know my wife*
Person, Question	Action	*you know how far the bar is*
Person, Number	Action	*I am seventy years old*
Person, Number	Action	*I have six children*
Person, Name	Action	*my name is Bond*
Person	Action	*I am hungry*
Person, Item	Action	*I like this pizza*
Person, Country	Action	*I live in Sweden*
Person, Language	Action	*I speak Polish*
Person, Currency	Action	*I have Swedish crowns*
Person, Object	Action	*I want two apples*
Person, Place	Action	*I want to go to the hospital*
Person	Question	*how old are you*
Item	Question	*how much does the pizza cost*
Item, Price	Proposition	*the pizza costs five euros*
Place	Proposition	*the museum is open*
Place, Date	Proposition	*the museum is open today*
Place, Day	Proposition	*the museum is open on Mondays*
Place, Date	Greeting	*see you in the bar on Monday*
Person	Person	*my wife, your husband*
Number, Currency	Proposition	*five euros*
Place	Question	*how far is the zoo*
Place, Place	Question	*how far is the centre from the hotel*
Transport, Place	Question	*which bus goes to the hotel*

Table 4.2: Some of the 130 combination rules of the Phrasebook.

```
PQuestion  : Question -> Phrase
QProp      : Proposition -> Question
PropAction : Action -> Proposition
ACitizen   : Person -> Citizenship -> Action
YouFamMale : Person
CitiNat    : Nationality -> Citizenship
Swedish    : Nationality
```

But thinking in terms of reliable translations, there are many more trees, resulting from the semantic ambiguity of English *you*. Of these, the Phrasebook deals with dimensions of gender and politeness; plural *you* is not covered by the current version (mostly because it is not so frequently needed). Thus *you* corresponds to four constants of type `Person`,

```
YouFamMale, YouFamFemale, YouPolMale, YouPolFemale
```

Varying this constant in the above tree gives four French linearizations:

YouFamMale:	*Est-ce que tu es suédois ?*
YouFamFemale:	*Est-ce que tu es suédoise ?*
YouPolMale:	*Est-ce que vous êtes suédois ?*
YouPolFemale:	*Est-ce que vous êtes suédoise ?*

Although German also has gender, it makes no difference in this example. Thus we obtain

YouFamMale, YouFamFemale:	*Bist du schwedisch?*
YouPolMale, YouPolFemale:	*Sind Sie schwedisch?*

One challenge in the Phrasebook is to communicate the ambiguities to the end user: when she types in *Are you Swedish?*, she should get a list of the alternatives in the desired target language, with explanations that enable her to decide which alternative to choose in her situation of use. We will return to this question in Section 4.

As `Action` is a subcategory of `Proposition`, it can be used for both questions and assertions, both positive and negated. Thus the functions involved in the question can be reused for sentences like *I am not Swedish*, which has two French translations. In general, the design of the abstract syntax follows two principles, which can be explained via geometrical metaphors:

Convexity: for any two phrases contained, also all phrases "between" them (i.e. combining their concepts in different ways) are contained. This principle guarantees that the users can easily learn what to expect from the phrasebook, and their expectations will be fulfilled.

Orthogonality: phrases are built from the least number of independent components.
While convexity is a great help for the user of the phrasebook, orthogonality helps the developer by giving her the minimum of concepts to implement for each language. A user who knows that the Phrasebook contains the property *Swedish* and the country *France* will, by convexity, expect it also to contain the property *French* and the country *Sweden*. The category `Nationality` is used to guarantee this, as it collects triples of language, property, and country. These triples can often be formed by a systematic word formation mechanism (e.g. *Swedish, Swedish, Sweden*), which helps the developer.

As a downside, abstract concepts like `Nationality` may be more complex to implement than more specific concepts like `Language`, `Citizenship`, and `Country`. [7] Often there is no regular word formation mechanism, and there are countries and languages that do not fit into the "national state" concept. For instance, the languages spoken in Belgium are Flemish and French. Thus in the Phrasebook, *Belgium* is a country without a lexically associated language, whereas *Flemish* is a language without a lexically associated country.

The set of combination rules in the Phrasebook is quite useful as it is, but the set of lexical items is still small and a little random. Therefore an obvious next step in developing the Phrasebook is to add words for drinks, food, nationalities, places, and so on. Keeping all this in synchrony for 15 simultaneous languages is not trivial.

3. Concrete Syntax

Constant phrases, such as *thank you* and *please*, are easy to define for all languages. Combination rules are more tricky: even in the small fragment covered by the Phrasebook, linguistic problems such as inflection, agreement, and word order arise, and require expertise in the grammar of each of the target languages. Fortunately for the Phrasebook, this expertise was readily available in the GF Resource Grammar Library (RGL). This is of course not just a lucky coincidence—it is more proper to say that the Phrasebook was built as a showcase of GF in general and of the RGL in particular.

A concrete syntax has two components. One is linearization rules (`lin`) as shown above, telling how abstract trees are mapped into strings. The other one is the **linearization types** of categories (`lincat`). These types are linguistic categories such as sentences, noun phrases, and adjectives. In the `lin` rule examples so far, only one linearization type was used: the type `Str` of strings. But this is usually not enough. For instance, to account for all combinations of a German noun, we need the type

```
{s : Number => Case => Str ; g : Gender}
```

that is, a record with a string depending on number and case (the component `s`), and a gender (component `g`). In other languages, nouns can have other linearization types, and the features number, case and gender can get other values than in German. But in RGL, all this complexity is defined internally, and the user only needs to know that the type `CN` covers common nouns in all RGL languages.[8]

To give a sample of linearization types used in the Phrasebook, let us consider the categories needed in the example *Are you Swedish?*:

category	**linearization type**	**explanation**
Phrase	`Text`	text
Question	`QS`	question
Proposition	`Cl`	clause
Person	`NP`	noun phrase
Action	`Cl`	clause
Citizenship	`A`	adjective
Nationality	`{l : NP ; p : A ; c : NP}`	NP, adjective, NP

[7]The terminological choice between "Nationality" and "Citizenship" is of course arbitrary, and only an implementation detail not visible to the end user.

[8]See http://grammaticalframework.org/lib/doc/synopsis.html for RGL categories and functions.

All these types are standard linguistic categories of RGL, except the one of `Nationality`, which uses a record consisting of a language noun phrase `l`, a property adjective `p`, and a country noun phrase `C`. This record is, so to say, the linguistic representation of the complex concept of a nationality, thus representing a **lexical family**.

The linearization rules are specified by RGL functions, most of which have the name `mk`*C* for the value category *C*. Thus we have

```
PQuestion q = mkText q
QProp p = mkQS (mkQCl p)
PropAction a = a
ACitizen p c = mkCl p c
YouFamMale = youSg_Pron
CitiNat n = n.p
Swedish = mkNationality "Sweden" "Swedish"
```

The last rule uses the operation `mkNationality`, which takes a string for a noun and for an adjective, to form the country name from the noun (*Sweden*) whereas both the property and the language use the adjective (*Swedish*). This is the only English-specific rule in this set. Other languages have different ways of defining this lexical family. Finnish, for instance, uses the country name as the language name, just spelled with a small initial.

Another example of a lexical family is types of locations. They are defined

```
PlaceKind = {name : CN ; at : Prep ; to : Prep}
```

Thus places have associated prepositions, used for expressing location and direction. For instance, in English we have *in the bar*, *at the station* for the location and *to* expressing the direction for both. In Finnish, prepositions are expressed by cases, so that "bar" uses so-called internal cases (*baarissa* "in the bar" inessive, *baariin* "to the bar" illative) whereas "station" uses external cases (*asemalla* "at the station" adessive, *asemalle* "to the station" allative). Sometimes even more fine-grained distinctions are needed; for instance, in Swedish "to the toilet" is expressed as *på toaletten* in phrases relating to the function ("I want to go to the toilet"), whereas phrases expressing pure direction say *till toaletten*.

The prepositions are thus stored in the record as lexical properties of the places; they are idiomatic in each language and highly unpredictable. GF provides ways to express them on a reasonably high level, so that just the minimal information need be given in the lexicon: thus in Finnish, we just need the noun and an identifier `ssa` or `lla` which is conventionally used for indicating the type of local case:

```
Bar = mkPlace (mkN "baari") ssa
Station = mkPlace (mkN "asema") lla
```

To determine this little piece of information—the proper case or preposition for each location—is linguistic knowledge that turned out to be possessed only by native speakers, who made several corrections to the initial grammars.

As the RGL has a common API for the syntax functions of the 18 languages included, combination rules in application grammars can in principle be expressed by code that is common to all languages. This is technically implemented by the use of **functors** ([10], chapter 5), and it is the technique used, for instance, in the GF implementation of Attempto Controlled English [13]. The use of a functor means that the languages use the same syntactic structures to express the meanings. For instance, all languages in the Phrasebook use an equivalent of *you know that I am in the bar* to express this proposition. However, the Phrasebook domain is particularly rich of idioms that the languages express by different syntactic means. This was a challenge we expected, and one of the reasons why the Phrasebook was an interesting case study for multilingual translation in the first place. Thus, of the 130 combination rules, only 96 (74%) are implemented by a functor; usually the percentage is close to 100.

Some typical examples of non-functorial expressions are the following:

I am fifty years old: French *j'ai cinquante ans* ("I have fifty years").

my name is Bond: German *ich heisse Bond* ("I have-name Bond"), French *je m'appelle Bond* ("I call myself Bond").

I am hungry: French *j'ai faim* ("I have hunger"), Finnish *minun on nälkä* ("of-me is hunger").

I like this pizza: Italian *questa pizza mi piace* ("this pizza pleases me").

I am married: Finnish *olen naimisissa* ("I am in-marriage", with a special adverbial).

how old are you: French *combien d'ans as-tu* ("how many years do you have").

how far is the station: French *à quelle distance est la gare* ("at what distance is the station"), Italian *quanto dista la stazione* ("how much does the station distance", with a special verb).

Most of these variations are clustered in systematic ways, so that for instance all Romance languages use the same structure and all Germanic languages (except perhaps English) another structure. The construct *how* with an adjective or adverb does not exist in Romance languages, and is hence not even a part of the RGL API.

4. Ambiguity and Disambiguation

The abstract syntax is by definition unambiguous. Therefore the main way in which a grammar developer can analyse the ambiguity of a string is by inspecting the abstract syntax trees. But this device is of course not appropriate for a tourist phrasebook: it would be awkward and often useless to show the syntax trees to the user.

Fortunately, the technique of multilingual grammars provides a straightforward, declarative way to display ambiguities: one can write for each language a special concrete syntax, which is like the original grammar except that it eliminates its ambiguities by using alternative (although less idiomatic and often longer) expressions—a **disambiguation grammar**. For example, the original English grammar linearizes each of the four abstract variants of *you* as just *you*, but the disambiguation grammar attaches an explanation in parentheses: *you (familiar,male)*, *you(polite,female)*, etc. This idea is inspired by the notion of **feedback texts** of the WYSIWYM system [14].

The implementation of a disambiguation grammar can be written on top of the base grammar by using **restricted inheritance**: it inherits everything from the base grammar except those rules that need disambiguation. Those rules can then be replaced by other rules. The following module is a complete code for a disambiguation grammar for the phrasebook dealing with the four *you*'s. The unambiguous variants are formed from *you* by attaching an adverbial to it.

```
concrete DisambPhrasebookEng of Phrasebook = PhrasebookEng -
  [YouFamMale, YouFamFemale, YouPolMale, YouPolFemale]
  ** open SyntaxEng, ParadigmsEng in {
lin
```

```
    YouFamMale   = mkNP you_NP (mkAdv "(familiar,male)") ;
    YouFamFemale = mkNP you_NP (mkAdv "(familiar,female)") ;
    YouPolMale   = mkNP you_NP (mkAdv "(polite,male)") ;
    YouPolFemale = mkNP you_NP (mkAdv "(polite,female)") ;
  }
```

In the full Phrasebook, the number of ambiguous constructs is between 10 and 20 for each language. An ambiguity shared by all languages is the notion of the currency *crown*, as used for the currency of different Scandinavian countries. In the normal usage, one says *crown* rather than e.g. *Danish crown*, if it is clear from the context that one is speaking about Danish crowns. The implementation of this does not use the disambiguation grammar, because both expressions make sense in the base grammar as well. Thus the base grammar defines crowns by using the **variants** construct of GF (expressed by |):

```
  DanishCrown =
    mkCN (mkA "Danish") (mkN "crown") | mkCN (mkN "crown")
  SwedishCrown =
    mkCN (mkA "Swedish") (mkN "crown") | mkCN (mkN "crown")
```

and similarly in all languages.

Since the abstract syntax encodes all interpretations that are relevant in any of the languages, it can lead to **spurious ambiguities** when applied to any particular language pairs. For instance, the familiar *you* is *sinä* and the polite *you* is *Te* in Finnish, without the gender distinction involved anywhere in the sentence. Hence, when translating from English to Finnish, only two alternatives should be displayed. The same thing may happen in Italian, where the masculine and feminine forms of some adjectives are the same. Thus *are you Swedish* has only two translations (*sei/è svedese*) even though *are you Italian* has four (*sei/è italiano/italiana*).

In the Phrasebook, the user should of course not see spurious ambiguities but only relevant ones. This is guaranteed by the following modification of GF's translation algorithm, which otherwise shows as many translation strings as there are parse results. Each translation is equipped by the set of those disambiguating expressions that give rise to it. The translation algorithm is as follows:

```
parse the source sentence to obtain trees $t_1, \ldots, t_n$
for each target language $L_i$:
for each tree $t_j$: linearize $t_j$ in $L_i$
group trees with the same linearization $s_k$ into the pair $< s_k, \{t \mid t^* = s_k\} >$
return each $s_k$ together with the linearizations of the associated trees in the disambiguation grammar of the target language
```

Here is an example of the algorithm at work:

English input:

Are you Swedish?

French output:

Est-ce que tu es suédois ? (Are you (Familiar,Male) Swedish?)

Est-ce que tu es suédoise ? (Are you (Familiar,Female) Swedish?)

Est-ce que vous êtes suédois ? (Are you (Polite,Male) Swedish?)

Est-ce que vous êtes suédoise ? (Are you (Polite,Female) Swedish?)

Italian output:

Sei svedese? (Are you (Familiar,Male) Swedish? / Are you (Familiar,Female) Swedish?)

È svedese? (Are you (Polite,Male) Swedish? / Are you (Polite,Female) Swedish?)

As a further optimization, the algorithm could compress the alternatives (Familiar,Male) and (Familiar,Female) to just (Familiar). This would be helped by a disambiguation grammar that has more structure than just the unanalysed strings in parentheses. One could also achieve this by some hand-written code in the phrasebook application; however, this would be against the purpose of developing the Phrasebook as a show-case for a general technology.

5. Example-Based Grammar Writing

In previous projects, the typical author of a GF concrete syntax is fluent in the target language and has GF skills which are directly proportional to the complexity of the abstract syntax to implement. However, when dealing with 15 languages and a reasonably rich semantic interlingua, the task of finding such people is a difficult one. When adding the time constraints yielded by the MOLTO deadlines and the time needed to improve a native speaker's GF skills or a GF programmer's knowledge of a language that she had little to no skill in before, the task seemed to be a mission impossible. This was the case for German, Dutch, Danish, Norwegian and Polish. As a solution to this, we devised the example-based grammar learning system, that is meant to automate a significant part of the grammar writing process and ease grammar development. The two main usages of the system are, first, to reduce the amount of GF programming necessary in developing a concrete grammar, and, secondly and more importantly, to make the extraction of certain features of a language automatic for grammar development.

In the last years, the GF community has constantly increased and so has the number of languages in the resource library and the number of application grammars using them. The writer of a concrete application grammar is typically different from the writer of the resource grammar for the same language, has less GF skills and is most likely unaware of the almost 300 constructors that the resource grammars implement for building various syntactical constructions [11]. In order to hide this detail, an API is provided so that the domain grammar writer only needs to know the GF categories and look up how they can be built from each other.

For example, the sentence *my name is John* is parsed to the following abstract syntax tree:

```
PredVP (DetCN (DetQuant (PossPron i_Pron) NumSg)
  (UseN name_N)) (UseComp (CompNP (UsePN john_PN)))
```

If we use the API constructors, the abstract syntax tree is simpler and more intuitive, as the structure is flatter and each function has an easily memorable name:

```
mkCl (mkNP (mkQuant i_Pron) name_N) (mkNP john_PN)
```

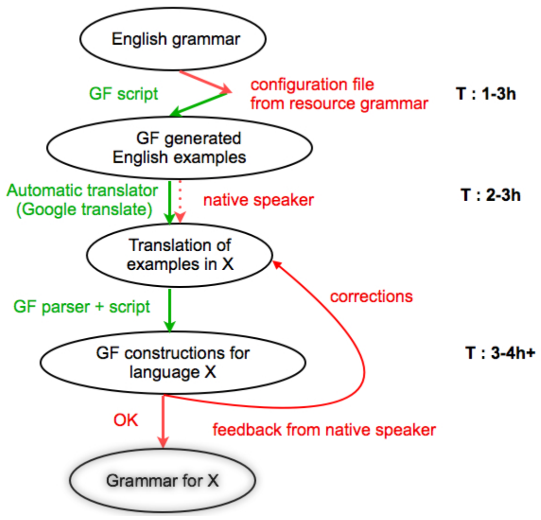


Figure 4.2: The example-based grammar learning schema

The example-based grammar learning system aims to make one step more in this direction and reduce the need for using even the API functions. The key idea is based on parsing, followed by compilation to API. It provides considerable benefits, especially for idiomatic grammars such as the Phrasebook, where the abstract syntax trees are considerably different.

For example, when asking for a person's name in English the question *what is her name* has the syntax trees shown above. On the other hand, in French the question would be translated to *je m'appelle John* (literally, "I call myself John"), which is parsed to:

```
PredVP (UsePron i_Pron)
  (ComplSlash (SlashV2 appeler_V2) (UsePN john_PN))
```

and corresponds to the following API abstract tree:

```
mkCl i_NP appeler_V2 (mkNP john_PN)
```

By replacing `i_NP` and `john_PN` with variables, this tree can be used as the linearization of a two-place predicate:

```
lin HasName x y = mkCl x appeler_V2 (mkNP y)
```

Figure 2 shows the algorithm for example-based grammar writing. It shows the construction steps of the concrete syntax of the Phrasebook grammar for the language *X*, where the developer has basic or no skills in the language. In our experiment *X* was one of Danish, Dutch, German, Norwegian, and Polish. The arrows represent the main steps of the process, whereas the circles represent the initial and final results after each step of the process. For every step, the estimated time is given. This is variable and greatly influenced by the features of the target language and the semantic complexity of the phrases and would only hold for the Phrasebook grammar.

Initial resources:

- English Phrasebook
- resource grammar for X
- script for generating the inflection forms of words and the corresponding linearizations of the lexical entries from the Phrasebook in the language X. For example, in the case of the nationalities, since we are interested in the names of countries, languages and citizenship of people and places, we would generate constructions like "I am English. I come from England. I speak English. I go to an English restaurant" and from the results of the translation we will infer the right form of each feature. In English, in most cases there is an ambiguity between the name of the language and the citizenship of people and places, but in other languages all three could have completely different forms. This is why it is important to make the context clear in the examples, so that the translation will be more likely to succeed. The correct design of the test of examples, is language dependent and assumes analysis of the resource grammar, also. For example, in some languages we need only the singular and the plural form of a noun in order to build its GF representation, whereas in other languages such as German, in the worst case we would need 6 forms which need to be rendered properly from the examples.
- script for generating random test cases that cover all the constructions from the grammar. It is based on the current state of the abstract syntax and it generates for each abstract function some random parameters and shows the linearization of the construction in both English and language *X*, along with the abstract syntax tree that was generated.

Example-based concrete grammar learning algorithm:

- Step 1: **Analysis of the target grammar and lexicon acquisition**

  The first step assumes an analysis of the resource grammar and extracts the information needed by the functions that build new lexical entries. A model is built so that the proper forms of the word can be rendered, and additional information, such as gender, can be inferred. The script applies these rules to each entry that we want to translate into the target language, and one obtains a set of constructions.

- Step 2: **Generation of examples in the target language**

  The generated constructions are given to an external translator tool (Google translate) or to a native speaker for translation. One needs the configuration file even if the translator is human, because formal knowledge of grammar is not assumed.

- Step 3: **Parsing and decoding the examples with GF**

  The translations into the target language are further more processed in order to build the linearizations of the categories first, decoding the information received. Furthermore, having the words in the lexicon, one can parse the translations of functions with the GF parser and generalize from that.

- Step 4: **Evaluation and correction of the resulting grammar**

  The resulting grammar is tested with the aid of the testing script that generates constructions covering all the functions and categories from the grammar, along with some other constructions that proved to be problematic in some language. A native speaker evaluates the results and if corrections are needed, the algorithm runs again with the new examples. The examples validated by the native informant are kept for regression testing of the future results. The algorithm is repeated as long as corrections are needed.

It is worth noting that the time needed for preparing the configuration files for a grammar will not be repeated, since the files are available for future usage. The time for the second step can be saved if automatic tools, like Google translate are used. This is only possible in languages with large corpora available. Good results were obtained for German and Dutch with Google translate, but for languages like Polish, which are both complex and lack enough resources, the results are discouraging. If the statistical oracle works well, the only step where the presence of a human translator is needed is the evaluation and feedback step. An average of 4 hours per round and 2 rounds were needed for the languages for which we performed the experiment. The final results are comparable to a grammar developed by a native speaker GF programmer.

However, one can already remark that the success of this method also depends highly on the lexicon acquisition, which we perform in the first step. What is more is that the lexicon is language-dependent, and is not alignable. Also, without previous knowledge of all the languages, one cannot foresee what words we would need to use, and since they are not used in all languages, it wouldn't make sense to have them all in a multilingual aligned lexicon. For the moment, this task was solved by either guessing the correct part-of-speech based on a similar concrete grammar already developed(for example Danish and Norwegian were bootstrapped from Swedish) or by having the lexicon built and POS-tagged with the aid of native informants.

Among the 5 languages considered, a concrete Phrasebook grammar was successfully built for Danish, Dutch, German and Norwegian, whereas for Polish, it was not possible to get through the first and most difficult step—target grammar analysis and lexicon acquisition, because of the complex morphology of the language and the lack of available resources. In the end the concrete grammar was developed by the writer of the resource grammar.

The experiment involved 7 programmers with basic or advanced GF skills that wrote 10 resource grammars, whereas for the 4 languages mentioned before, the example-based algorithm was used. The approximate development total time is 1 person month for the whole Phrasebook, or 1.5 days per language on the average.

Based on this case study, we roughly estimated the effort used in constructing the necessary sources for each new language and compiled Table 3.

Explanation of the scores

Grammarian's language skills:

- − : no skills
- # : basic skills(general knowledge of the grammar)
- ## : medium skills(fluent)
- ### : advanced skills(native speaker)

Language	**Fluency**	**GF skills**	**Inf. dev.**	**Inf. testing**	**Ext. tools**	**RGL edits**	**Effort**
Bulgarian	###	###	-	-	-	#	##
Catalan	###	###	-	-	-	#	#
Danish	-	###	+	+	+	##	##
Dutch	-	###	+	+	+	#	##
English	##	###	-	+	-	-	#
Finnish	###	###	-	-	-	#	##
French	##	###	-	+	-	#	#
German	#	###	+	+	+	##	###
Italian	###	#	-	-	-	##	##
Norwegian	#	###	+	+	+	#	##
Polish	###	###	+	+	+	#	##
Romanian	###	###	-	-	+	###	###
Spanish	##	#	-	-	-	-	##
Swedish	##	###	-	+	-	-	##

Table 4.3: Development effort for the Phrasebook.

Grammarian's GF skills

- — : no skills
- # : basic skills(simple GF exercises)
- ## : medium skills(more comprehensive GF exercises)
- ### : advanced skills(resource grammar writer/substantial contributor)

Informant needed for development/Informant needed for testing

- —: no
- + : yes

Changes on the resource grammars

- —: no changes
- # : 1-3 minor changes
- ## : 4-10 minor changes, 1-3 medium changes
- ### : >10 changes of any kind

Overall effort

# : less than 8h/person

## : 8-24h/person

### : >24h/person

This experiment is significant because it is a showcase for the ongoing work on example-based concrete grammar learning technology which will make GF grammar writing easier in terms of adding more languages and developing larger grammars, but also because it represents an analysis on the possible interaction of GF with other available translation tools, which will ease the work of both beginners and advanced users of the technology.

6. The Mobile Application

If one wants to build a tool for a controlled language for everyday usage, it seems logical for this tool to be as unobtrusive as possible. Moreover, since our language is targeting tourists, we have to take into account a particular setting where people, when going on vacation, may not have access to a computer and access to Internet can be very limited due to low coverage or prohibitive costs. This are the criteria that we tried to meet when building PhraseDroid, an application that works offline, on smartphone devices and with a simple user interface. Moreover, we wanted to do this by creating a technology that is as general as possible, and in fact applies to any multilingual GF grammar.

PhraseDroid is an Android application, that can be used on handheld devices running the Android operating system. Figure 1 shows a screen shot of the application in its current state. As you can see, the application is using the same "magnet interface" as the web application. This permits the user to compose a sentence while staying in the coverage of the grammar. Moreover, this kind of interaction works well on devices with touch screens because the magnets are large enough to be able to be selected with fingers.

What is more is that the Android platform provides a high-quality speech synthesis for several of the languages covered by the grammar, which can be plugged into our application. This gives clear benefits compared to a traditional (paper) phrasebook.

As mentioned in Section 1, there are more and more phrasebook applications developed for smartphones nowadays. They can be divided in two main categories:

The finite phrasebook. Those are usually made of a list of sentences translated in one, or more, foreign languages. Those phrasebooks are lacking from the point of view of expressivity since it isn't possible to change a sentence as needed, even if a very similar sentence is covered by the phrasebook.

The application providing machine translation through an on-line service. The Google Translate application (and the various applications that are just front-ends to it) is the best example in this category. This kind of applications can obviously be used while traveling, but they require the possibility to connect to the Internet, which is not guaranteed when one is travelling abroad due to technological or economical reasons. In addition, unlike in our application, the translation engine is not tailored toward tourist usage but is generic, which can lead so sub-optimal translation in many cases.

In contrast, our application, once installed, works off-line and features a grammar design specifically for tourist translations. And since the user inputs the sentence to be translated herself, it allows a great deal of variation and fine-tuned translation for a given situation.

The application is based on a Java interpreter for GF's binary grammar format, called PGF [15]. Therefore the application is very modular: adapting the application to a different controlled language requires little more than dropping a new pgf file in the right folder. This means that one can in no time create a translation application for another controlled language given that a GF grammar for this language has been written.

Thus an important part of the process of creating the application was to write a library in Java that provides the functions needed in the application. Its usage is not limited to Android phones, but it can also be plugged in into any Java program, whether on a desktop computer, or a web browser plug-in. In the current state, the Java library supports (predictive) parsing, linearization, and random generation of well-typed trees.

This is less than the features available in the full GF interpreter, written in Haskell, but it is sufficient for machine translation and lots of other uses.

7. Evaluation

7.1. Translation Quality

This is the first criterion of evaluation. It was first assessed by the systematic use of native speaker testers, and later by comments collected from more random users of the web demo and the Android application. The goal has been what might be called *perfect quality*, in terms of meaning-preservation, grammaticality, idiomaticity, and fluency. Hence all errors found in earlier versions were corrected immediately. With the first “official version” (the one also running on Android), few direct errors have been found, but there are some inadequacies that appear in reports:

Some sentences permitted by the abstract syntax are semantically anomalous, e.g. *is there an airplane to the toilet*. This could be fixed by using a more strict type system; however, we consider this to be less important as long as the translations are correct.

The choice of prepositions is not always fine-grained enough; for instance the distinction between *gå till toiletten* and *gå på toaletten* (both “go to the toilet”) in Swedish is not handled (cf. Section 4).

The usage of nationality adjectives for persons is not always optimal, but nouns should be introduced in the lexical family. Thus *I am a Finn* would be better than *I am Finnish*, with corresponding variations in many languages.

7.2. Coverage

There is no end of conceivable extensions if we want to cover everything that a tourist might want to say. The syntactic combination rules are sufficient for many situations, but they should definitely be extended with more vocabulary. For instance,

drinks, food, currencies, countries

time expressions like *half past eight*

free-string input for names of places and persons

7.3. Engineering Effort

One of the main goals of the MOLTO project is to improve the productivity of GF-based translation systems "by an order of magnitude". This means that the development time of translation systems should be shortened to 10% of the original. The development time for the Phrasebook was two working days per language on average. If this is the baseline to be compared with at the end of the project (in 2013), then a new language should be possible to add in a couple of hours. Some of this improvement can be possible to reach by a better use of example-based grammar writing.

However, some parts of the grammar may be inherently difficult, due to idiomatic structures. Another way to interpret the productivity improvement would then be in terms of the concepts covered. If the first Phrasebook built in two days covers hundreds of concepts, a realistic goal could be to cover thousands of concepts in the same time. To this end, methods of automatic lexicon extraction are being developed, with ontologies, terminologies, and statistical translation models as sources.

7.4. Usability

The size of the run-time PGF grammar is 500 kB. It runs smoothly on both web applications and mobile phones. For mobiles, a substantial optimization effort was needed, but it was made on the level of GF and will therefore benefit all future applications.

The web application provides the input method of typing strings, which the mobile doesn't have. This will certainly become an issue when the Phrasebook is extended to contain thousands of concepts. It will also become an issue how to navigate in the large space of words to find exactly the phrase one wants to use. A hierarchical approach similar to syntax editors [16] will probably be introduced as a useful complement to string-based input.

The mobile application has some usability issues reported by users, which will have to be addressed in future releases.

8. Conclusion

We have explained a controlled language approach to a multilingual tourist phrasebook, covering 15 languages. While intending to build a useful application for travellers, we have also seen it as an experiment to extend the notion of controlled language and scale it up in various respects:

extending the notion of semantics from logic to "language games" (Section 2)

porting a controlled language from one language to many (Section 3)

coping with ambiguity, rather than banning it (Section 4)

making it easier to implement controlled languages, in terms of both effort and skill (Section 5)

building applications for laymen rather than specialists, and making them run on light devices (Section 6)

Our conclusion is that there is a lot of potential in controlled languages to become more useful in everyday life, the multilinguality aspect being at least as interesting for laymen as the traditional reasoning aspect is.

Acknowledgements

The MOLTO Phrasebook is a collaborative project. In addition to the authors of this paper, it has involved Krasimir Angelov, Olga Caprotti, Thomas Hallgren, Inari Listenmaa, Jordi Saludes, Adam Slaski, and Shafqat Virk as programmers and Richard Bubel, Rise Eilert, Karin Keijzer, Michał Pałka, Willard Rafnsson, and Nick Smallbone as testers and informants. The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement n:o FP7-ICT-247914.

Bibliography

[1] The Boeing Company: Boeing Simplified English Checker. http://www.boeing.com/assocproducts/sechecker/ (2001)

[2] Shiffman, R.N., Michel, G., Krauthammer, M., Fuchs, N.E., Kaljurand, K., Kuhn, T.: Writing clinical practice guidelines in controlled natural language. In: Proceedings of the 2009 conference on Controlled natural language. CNL'09, Berlin, Heidelberg, Springer-Verlag (2010) 265–280

[3] Hart, G., Johnson, M., Dolbear, C.: Rabbit: Developing a control natural language for authoring ontologies. In: ESWC. (2008) 348–360

[4] Fuchs, N.E., Kaljurand, K., Kuhn, T.: Attempto Controlled English for Knowledge Representation. In Baroglio, C., Bonatti, P.A., Małuszyński, J., Marchiori, M., Polleres, A., Schaffert, S., eds.: Reasoning Web, Fourth International Summer School 2008. Number 5224 in LNCS, Springer (2008) 104–124

[5] Dean, M., Schreiber, G.: OWL Web Ontology Language Reference (2004) http://www.w3.org/TR/owl-ref/.

[6] Gruzitis, N., Barzdins, G.: Towards a More Natural Multilingual Controlled Language Interface to OWL. In: 9th International Conference on Computational Semantics (IWCS). (2011) 335–339 http://www.aclweb.org/anthology/W/W11/W11-0138.pdf.

[7] Wittgenstein, L.: Philosophical Investigations. Basil Blackwell, Oxford (1953)

[8] Ganesalingam, M.: The Language of Mathematics. PhD thesis, Department of Computer Science, University of Cambridge (2010) http://people.pwf.cam.ac.uk/mg262/.

[9] Angelov, K.: Incremental Parsing with Parallel Multiple Context-Free Grammars. In: Proceedings of EACL'09, Athens. (2009)

[10] Ranta, A.: Grammatical Framework: Programming with Multilingual Grammars. CSLI Publications, Stanford (2011) ISBN-10: 1-57586-626-9 (Paper), 1-57586-627-7 (Cloth).

[11] Ranta, A.: The GF Resource Grammar Library. Linguistics in Language Technology **2** (2009) http://elanguage.net/journals/index.php/lilt/article/viewFile/214/158.

[12] Bringert, B., Angelov, K., Ranta, A.: Grammatical Framework Web Service. In: System demo, Proceedings of EACL'09, Athens. (2009)

[13] Ranta, A., Angelov, K.: Implementing Controlled Languages in GF. In: Proceedings of CNL-2009, Athens. Volume 5972 of LNCS. (2010) 82–101

[14] Power, R., Scott, D.: Multilingual authoring using feedback texts. In: COLING-ACL 98, Montreal, Canada (1998)

[15] Angelov, K., Caprotti, O., Enache, R., Hallgren, T., Listenmaa, I., Ranta, A., Saludes, J., Slaski, A.: D10.2 molto web service, first version. (D10.2) (06/2010 2010)

[16] Khegai, J., Nordström, B., Ranta, A.: Multilingual Syntax Editing in GF. In Gelbukh, A., ed.: Intelligent Text Processing and Computational Linguistics (CICLing-2003), Mexico City, February 2003. Volume 2588 of LNCS., Springer-Verlag (2003) 453–464 `http://www.cs.chalmers.se/~aarne/articles/mexico.ps.gz`.

Chapter 5

Grammar-Based Hybrid Systems for Machine Translation

1 Patent Translation within the MOLTO Project

Cristina España-Bonet Ramona Enache
Adam Slaski Aarne Ranta
Lluís Màrquez Meritxell Gonzàlez

Abstract: MOLTO is an FP7 European project whose goal is to translate texts between multiple languages in real time with high quality. Patents translation is a case of study where research is focused on simultaneously obtaining a large coverage without loosing quality in the translation. This is achieved by hybridising between a grammar-based multilingual translation system, GF, and a specialised statistical machine translation system. Moreover, both individual systems by themselves already represent a step forward in the translation of patents in the biomedical domain, for which the systems have been trained.

1. Introduction

MOLTO[1] is an European project within the Seventh Framework Programme. Its main goal is to develop a set of tools for translating texts between multiple languages in real time with high quality.

MOLTO clearly bets for high quality translation, the cost to pay is to limit the coverage to restricted domains which can be covered by a grammar. As its main technique, the project uses domain-specific semantic grammars and ontology-based interlinguas. These components are implemented in GF (Grammatical Framework) [12], which is a grammar formalism where multiple languages are related by a common abstract syntax. Up to now, GF has been applied in several small-to-medium size domains such as dialogue systems[2] or the translation of mathematical exercises[3].

When dealing with real text from a given domain, a grammar fails to cover any ungrammatical construction used. However, empirical machine translation systems in general, and statistical machine translation systems (SMT) in particular, have good coverage on any sort of text. The aim of MOLTO is to get the best of both worlds by building a hybrid GF-SMT system that achieve high-precision and good coverage.

Patents have been chosen for the opening of the system to non-restricted language. This election has two main reasons. First, the language of patents, although having a large amount of vocabulary and richness of grammatical structure, still uses a formal style that can be interpreted by a grammar. And second, there is nowadays a growing interest for patents translation. The high and increasing number of registered patents has created a huge multilingual database of patents distributed all over the world. So,

[1]MOLTO: FP7-ICT-247914, 2010–2013, www.molto-project.eu

[2]TALK project, Tools for Ambient Linguistic Knowledge: IST-507802, 2004–2006, www.talk-project.org

[3]WebALT project, Web Advanced Learning Technologies: EDC-22253, 2005–2007, webalt.math.helsinki.fi

SET	Segments	EN tok	DE tok	FR tok
Training	279,282	7,954,491	7,346,319	8,906,379
Development	993	29,253	26,796	33,825
Test	1,008	31,239	28,225	35,263

Table 5.1: Numbers for the patents aligned corpus in English (EN), German (DE) and French (FR).

there is an actual need for building systems able to access, search and translate patents, in order to make these data available to a large community.

The objective of MOLTO with respect to patents translation is twofold. On one hand, research on hybrid translation systems is being carried out to study the best approach to combine GF and a SMT system. On the other hand, a prototype for machine translation and retrieval within patents will be built. The purpose of this paper is to focus on the first part and depict the current status and prospects for the translation system.

The paper is organised as follows. The next section, Section 1, describes the corpora and linguistic processors used in this work. We detail in Section 1 the two independent systems to translate patents. Afterwards, Section 3 depicts the hybridization prospects for these systems, and finally Section 1 summarises and outlines future work.

2. Patents Domain

A patent is an official document granting a right. Besides the terms of the patent itself, it also contains information about its publication, authorship and classification for example. Being an official document, the structure giving the terms of the patent is quite fixed. Every patent has a title, a description, an abstract with the most relevant information and a series of claims.

A claim is a single (possibly very long) sentence composed mainly of two parts: an introductory phrase and the body of the claim, usually linked by a conjunction. It is in the body of the claim where there is the specific legal description of the exact invention. Therefore, claims are written in a lawyerish style and use a very specific vocabulary of the domain of the patent.

2.1. Corpus

MOLTO works with European patents and the task is restricted to English, French and German. A first domain of application includes biomedical and pharmaceutical patents. We select patents with IPC (International Patent Classification) code A61P, corresponding to “Specific therapeutic activity of chemical compounds or medical preparations”.

A parallel corpus in the three languages has been gathered from the corpus of patents given for the CLEF-IP track in the CLEF 2010 Conference[4]. These data are an extract of the MAREC corpus, containing over 2.6 million patent documents pertaining to 1.3 million patents from the European Patent Office[5] (EPO). Our parallel corpus is a subset with those patents with translated claims and abstracts into the three languages. From this first subset we selected those patents that deal with the appropriate domain.

[4]http://clef2010.org/

[5]http://www.epo.org/

The final corpus built this way covers 56,000 patents out of the 1.3 million. That corresponds to 279,282 aligned parallel fragments as it can be seen in Table 5.1. A fragment is the minimum segment aligned in the three languages, so, it is shorter than a claim and, consequently, shorter than a sentence. Two small sets for development and test purposes have also been selected with the same restrictions: 993 fragments for development and 1008 for test.

2.2. Linguistic processors

The detection and correct tokenisation of chemical compounds has been shown to be crucial in the performance of translators (see Section 1 for the analysis). A regular tokeniser would for example split the *compound* “cis-4-cyano-4-(3-(cylopentyloxy)-4-methoxyphenyl)cyclohexane-1-carboxylic” by the puctuation into 9 tokens and, consequently, each of the tokens would be translated as an independent word. To deal with this peculiarity of the domain, we developed a pipeline to detect, tokenise and translate compounds.

Compound recogniser and tokeniser As a first approximation we devise a recogniser and tokeniser based on affix detection. A list with 150 affixes has been compiled and it is used to select the candidate tokens to be a compound from the corpus. The candidates selected this way are matched against a dictionary and those without a match are considered to be compounds and do not get an internal tokenisation. 103,272 compounds are found with this procedure within the training corpus defined in the previous section.

However, this list of compounds contains some noise. Examples of noise are in this context proper names with the defined affixes (Hôpital), words that do not appear in the dictionary (extracorporeal) or simply typos (comparoate). The amount of noise is considerable, but extra words do not in general imply a wrong tokenisation. So, the method works better as a (non-)tokeniser than as a compound detector and it bets for high recall instead of precision.

Given the power of GF, one can also build a simple grammar for translating compounds. What makes the difference between this rule-based approach and a mere translation of each word in the compound is that in this case the possible reordering of the words is already defined by the grammar. So, functional words like acid, ester or aldehyde swap its position with the radical words whenever necessary.

Part-of-speech tagger, lemmatiser and named entity recogniser Part-of-speech (PoS) tagging and lemmatisation are necessary in the lexicon building of the patents grammar. `GENIA` [14], a linguistic processor prepared specially to process texts from the biomedical domain, is used for both purposes.

Named entities are marked in the text and are not translated by GF, but translated independently and substituted afterwards. In the biomedical domain, a simple heuristic works as well tagging proper names as a state-of-the-art tagger not specifically trained. We consider to be proper names the words starting with a capital letter (after lowercasing the sentences), and the words containing numbers or special characters inside. This simple methodology lead to 100% precision and recall for the first 200 fragments in the training corpus of Section 3, where the proper names were manually annotated and the output was compared to that of the named entity recogniser. In this case 176 proper names were properly classified and replaced with a place holder name.

3. Individual translation systems for patents

The translation of patents can be approached through different methods. In this work we focus on GF and SMT systems, and specialise the two of them into the patents domain.

3.1. Interlingua-based translation, GF

The key concept of GF is the division of a grammar in an abstract syntax part and the concrete syntaxes corresponding to each of the target languages. The largest and most general example of such a grammar is the *resource library* [13], comprising 20 languages, for which the main grammatical constructions are provided. The library can be further used by domain-specific grammars, which can use the grammatical constructions from here alleviating the burden of handling linguistic difficulties and allowing a better focus on the higher-level details.

Even with this easiness, building a rule-based general-purpose translation system is a laborious task. However, we assume that most of the claims can be covered by a limited set of grammatical constructions and extend the GF resource grammar with these constructions.

Grammars like this one with non-trivial coverage usually are ambiguous, the number of the interpretations is the product of the number of parse trees for each subconstruction. On the good side, the grammar covers all possible interpretations, but on the other side, in order to make it usable, statistical based disambiguation needs to be used.

The task of translation is resumed to parsing from the source language to an abstract syntax tree and linearising it in the target language. Still, the system is restricted to the language generated by the grammar. Lexicon building is then an important step since the vocabulary of patent claims is virtually unlimited. The GF library multilingual lexicon contains the most common entries for structural parts-of-speech and it is used as a base to be extended with nouns, adjectives, verbs and adverbs. The abstract syntax for these PoS is created from the claims in one language (English). Once it is built, it is lemmatised and manually corrected from noise and ambiguities. Then, the proper inflection is generated using the implemented GF paradigms and the English dictionary of the GF library. Base forms are translated into the necessary languages and the inflection is generated for each of them.

The following figure shows the basic steps of the full system's behaviour:

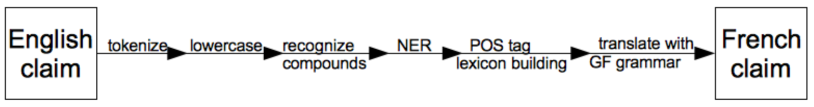


Figure 5.1: GF translation system for patent claims.

Up to now, performance of the grammar aimed to parse full claims is still unsatisfactory. The high level of ambiguities remaining results in slowness, and coverage is up to now a 15% of the working corpus. Hybrid systems can deal with ambiguities, i.e., multiple translation options, and can complete with statistical translations the parts not covered by GF. However, the grammar must be expanded so that the two systems can collaborate on equal terms.

	DE2EN			EN2DE		
METRIC	**Bing**	**Google**	**Domain**	**Bing**	**Google**	**Domain**
1−WER	0.52	0.64	**0.72**	0.42	0.51	**0.69**
1−TER	0.59	0.67	**0.76**	0.45	0.53	**0.71**
BLEU	0.43	0.58	**0.65**	0.33	0.45	**0.58**
NIST	8.25	9.67	**10.12**	6.53	8.05	**9.40**
ROUGE-W	0.40	0.48	**0.52**	0.34	0.41	**0.48**
GTM-2	0.30	0.40	**0.47**	0.25	0.32	**0.43**
METEOR-pa	0.60	0.69	**0.74**	0.36	0.45	**0.57**
ULC	0.09	0.29	**0.41**	0.03	0.19	**0.43**

Table 5.2: Automatic evaluation using a set of lexical metrics of the in-domain SMT system for the English-German language pair. Results of two state-of-the-art systems, Bing and Google, are showed for comparision.

	FR2EN			EN2FR		
METRIC	**Bing**	**Google**	**Domain**	**Bing**	**Google**	**Domain**
1−WER	0.54	0.66	**0.78**	0.57	0.63	**0.73**
1−TER	0.59	0.70	**0.80**	0.60	0.66	**0.74**
BLEU	0.45	0.62	**0.70**	0.43	0.53	**0.62**
NIST	8.52	10.01	**10.86**	8.39	9.21	**9.96**
ROUGE-W	0.41	0.50	**0.54**	0.39	0.45	**0.49**
GTM-2	0.32	0.43	**0.53**	0.31	0.36	**0.45**
METEOR-pa	0.61	0.72	**0.77**	0.57	0.65	**0.71**
ULC	0.07	0.28	**0.44**	0.10	0.23	**0.39**

Table 5.3: As in Table 5.2 for the English-French language pair.

3.2. Statistical translation, SMT

The statistical system is a state-of-the-art phrase-based SMT system trained on the biomedical domain with the corpus described in Section 3. Its development has been done using standard freely available software. A 5-gram language model is estimated using interpolated Kneser-Ney discounting with `SRILM` [6]. Word alignment is done with `GIZA++` [7] and both phrase extraction and decoding are done with the `Moses` package [8, 9]. The optimisation of the weights of the model is trained with MERT [10] against the BLEU [11] evaluation metric.

Table 5.2 shows a first evaluation of this system (Domain) using a variety of lexical metrics. This set of metrics is a subset of the metrics available in the `Asiya` evaluation package [16]. We specifically select this set of metrics because all of them are available for the three languages: English, German and French. Together with our in-domain system we show the same evaluation for two public SMT systems for general translation: Bing[6] and Google[7]. These systems can be considered the state-of-the-art of a SMT open domain translator.

In general, our in-domain trained system performs significantly better than the two general purpose ones mainly because of two reasons. First, it has been trained on the

[6]http://www.microsofttranslator.com/
[7]http://translate.google.com

	DE2FR			FR2DE		
METRIC	**Bing**	**Google**	**Domain**	**Bing**	**Google**	**Domain**
1−WER	0.42	0.52	**0.76**	0.30	0.43	**0.65**
1−TER	0.47	0.56	**0.68**	0.32	0.46	**0.66**
BLEU	0.29	0.43	**0.56**	0.24	0.39	**0.53**
NIST	6.72	8.21	**9.10**	5.35	7.30	**8.88**
ROUGE-W	0.31	0.38	**0.45**	0.29	0.37	**0.44**
GTM-2	0.24	0.30	**0.41**	0.21	0.28	**0.41**
METEOR-pa	0.45	0.56	**0.64**	0.26	0.39	**0.51**
ULC	0.03	0.22	**0.41**	-0.03	0.19	**0.44**

Table 5.4: As in Table 5.2 for the French-German language pair.

specific domain and second, the tokenisation tools have been specifically developed to deal with chemical compounds. The concrete values can be read in Tables 5.2, 5.3 and 5.4 for the language pairs English-German, English-French and French-German respectively.

Even though the Domain system shows a good performance among SMT systems, some of the observed translation errors would not be produced by a rule-based system, which, on the other hand, would probably produce different ones. Table 5.5 displays two translations from German into English where this is made evident. In the first one, systems are not able to capture the different order in the verb position, although the translation is adequate lexically. The second sentence is an example of the importance of the chemical names. Google, for instance, tokenises the compound by the punctuation. Some of the tokens are then translated, but the full compound is not recovered. Bing and Domain do not tokenise the compound, but according to the results, the word does not appear in the training corpus and has not been translated. These kinds of errors can be easily alleviated by the GF grammar and are a motivation to combine GF and SMT for the translation of patents.

4. Hybridisation approaches

Hybrid approaches in MOLTO depart from three key assumptions when facing the combination of paradigms: 1) the quality of a completely translated sentence by a GF-based system will be always better than the translation obtained with SMT; 2) when the GF-based systems fails at producing a complete translation it can probably produce a set of partial translations (phrases) with confidence scores or probabilities; 3) the SMT system is always capable of generating an output translation. Assumption number one implies that our combination setting will be set as a fall-back strategy, i.e., in general SMT will be seen as a back-off for GF-based MT. Assumption number two makes it possible to combine partial outputs from GF with the SMT system in a real hybrid approach. Assumption number three guarantees that a translation will be always output by the combined system.

Keeping these premises in mind we develop combination schemes to integrate grammar-based and statistical MT systems in a hybrid approach. We can divide the schemes in three big groups:

Hard integration: Force fixed GF fragment translations within a SMT system.

DE	Verwendung nach Anspruch 23 , worin das molare Verhältnis von Arginin zu Ibuprofen 0,60 : 1 beträgt .
EN	**The use** of claim 23 , wherein the molar ratio of arginine to ibuprofen **is** 0.60 : 1 .
Domain	The use of claim 23 , wherein the molar ratio of arginine to ibuprofen 0.60 : 1 .
Google	The **method** of claim 23 , wherein the molar ratio of arginine to ibuprofen 0.60 : 1 **is** .
Bing	Use of claim 23 , wherein the molar ratio of arginine to ibuprofen is 0.60 : 1 .

DE	(±)-N-(3-Aminopropyl)-N,N-dimethyl-2,3-bis(syn-9-tetradecenyloxy)-1-propanaminiumbromid
EN	(±)-N-(3-**a**minopropyl)-N,N-dimethyl-2,3-bis(syn-9-tetradeceneyloxy)-1-propanaminium **bromide**
Domain	(±)-N-(3-Aminopropyl)-N,N-dimethyl-2,3-bis(syn-9-tetradecenyloxy)-1-propanaminiumbromid
Google	(±)-N-(3-aminopropyl)-N , N-dimethyl-2 , 3-bis (syn-9-tetradecenyloxy) is 1-propanaminiumbromid
Bing	(±)-N-(3-Aminopropyl)-N,N-dimethyl-2,3-bis(syn-9-tetradecenyloxy)-1-propanaminiumbromid

Table 5.5: Examples of wrong German-to-English translations in SMT systems. This kind of errors are not produced by the GF grammar for translating compounds.

Soft integration led by SMT: Make available GF fragment translations to a SMT system.

Soft integration led by GF: Complement with SMT options the GF translation structure.

Each one of these options involves either the modification of the original systems or the construction of a new architecture. The most important thing in order to combine methodologies is that GF is able to parse general text robustly, it must be able to skip those structures not covered by the grammar and give some general information so that the statistical component of the engine takes care of the fragments. The first work on this task is the robust parser being developed for GF. Current experiments use shallow parsing as a first approximation and efforts are being made to increase the coverage.

Similarly, it is important that systems can share information. In order to make available GF translations to a SMT system one mainly needs to be able to feed an SMT decoder with translation pairs. GF translation pairs can be obtained by using its high quality alignments and extract the phrases in the SMT style. Since GF alignments are reliable, this will add a set of high quality phrases to be combined with those coming from the pure SMT system in the translation table. GF has been adapted for this purpose so that it is able to generate both alignments in the usual format[8] and with a text Giza-like nomenclature.

4.1. Ongoing work: robust GF with an extended lexicon

A first combination of GF and statistical methods is being developed for the English-to-French translation of patents. The kernel of the system is the Interlingua based translation of Section 1. The system uses the patents grammar together with the resource grammar, builds automatically the lexicon from the English text and translates it into French. The GF translation mechanism is then applied on sentences that can be parsed, otherwise a chunker is used to fragment the sentence and only the parts of the sentence that can be handled by the grammar are translated. The other parts can be

[8]Graphviz, an open source graph visualization software (http://www.graphviz.org/).

sent to the SMT system, or alternatively the SMT system can be fed with the phrases translated with GF.

5. Conclusions

One of the goals of the MOLTO project is to build a high-quality and robust translator for patents in at least three European languages: English, German and French. In order to achieve this purpose several systems are being developed. One of them is a multilingual rule-based translation system, and another one is a statistical translation system. Both of them depart from general systems and have been specialised into the patents domain. Besides, these two approaches will be merged to forge hybrid systems, and some ongoing work is devoted to build independent modules of the individual systems that can ease the integration.

The work is still in progress. The GF grammar needs a more thorough evaluation, in order to decide upon future extensions that would improve its coverage. A dedicated chunker in the three languages is also being built to divide claims and allow a separate treatment. And probably the most difficult step is to implement a disambiguation module that deals with the open language found in patents.

On the other hand, the in-domain SMT system already outperforms state-of-the-art general translation systems. The more advanced hybrids will combine the large coverage shown by SMT together with the capabilities of GF in generating grammatically correct translations.

Acknowledgements

This work has been funded by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement number 247914 (MOLTO project, FP7-ICT-2009-4-247914). The work was conducted using the Matrixware Research Collection, provided by IRF `www.ir-facility.org`. Authors are thankful to Aurélien Max and Xavier Auvray for their help with the French processing tools and the preliminary evaluation respectively.

Bibliography

[1] Ranta, A.: Grammatical Framework: Programming with Multilingual Grammars. CSLI Publications, Stanford (2011) ISBN-10: 1-57586-626-9 (Paper), 1-57586-627-7 (Cloth).

[2] Tsuruoka, Y., Tateishi, Y., Kim, J., Ohta, T., McNaught, J., Ananiadou, S., Tsujii, J.: Developing a robust part-of-speech tagger for biomedical text. In Bozanis, P., Houstis, E.N., e., eds.: Advances in Informatics. Volume 3746. Springer Berlin Heidelberg (2005) 382–392

[3] Ranta, A.: The GF resource grammar library. Linguistic Issues in Language Technology **2**(1) (2009)

[4] Stolcke, A.: SRILM – An extensible language modeling toolkit. In: Proc. Intl. Conf. on Spoken Language Processing. (2002)

[5] Och, F.J., Ney, H.: A systematic comparison of various statistical alignment models. Computational Linguistics **29**(1) (2003) 19–51

[6] Koehn, P., Shen, W., Federico, M., Bertoldi, N., Callison-Burch, C., Cowan, B., Dyer, C., Hoang, H., Bojar, O., Zens, R., Constantin, A., Herbst, E., Moran, C.: Open Source Toolkit for Statistical Machine Translation. Technical report, Johns Hopkins University Summer Workshop. http://www.statmt.org/jhuws/ (2006)

[7] Koehn, P., Hoang, H., Mayne, A.B., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., Herbst, E.: Moses: Open source toolkit for statistical machine translation. In: Annual Meeting of the Association for Computation Linguistics (ACL), Demonstration Session. (Jun 2007) 177–180

[8] Och, F.J.: Minimum error rate training in statistical machine translation. In: Proc. of the Association for Computational Linguistics, Sapporo, Japan (July 6-7 2003)

[9] Papineni, K., Roukos, S., Ward, T., Zhu, W.J.: Bleu: a method for automatic evaluation of machine translation. In: Proceedings of the Association of Computational Linguistics. (2002) 311–318

[10] Giménez, J., Màrquez, L.: Asiya: An Open Toolkit for Automatic Machine Translation (Meta-)Evaluation. The Prague Bulletin of Mathematical Linguistics (94) (2010) 77–86

2 A Hybrid System for Patent Translation

Ramona Enache Cristina España-Bonet
Aarne Ranta Lluís Màrquez

Abstract: This work presents a HMT system for patent translation. The system exploits the high coverage of SMT and the high precision of an RBMT system based on GF to deal with specific issues of the language. The translator is specifically developed to translate patents and it is evaluated in the English-French language pair. Although the number of issues tackled by the grammar are not extremely numerous yet, both manual and automatic evaluations consistently show their preference for the hybrid system in front of the two individual translators.

1. Introduction

Hybrid MT (HMT) is an emerging and challenging area of machine translation, which aims at combining the known techniques into systems that retain the best features of their components, and reduce the disadvantages displayed by each of the methods when used individually.

This work presents a hybrid translation system specifically designed to deal with the translation of patents. The language of patents follows a formal style adequate to be analysed with a grammar, but at the same time uses a rich and particular vocabulary adequate to be gathered statistically. We focus on the English-French language pair so that the effects of translating into a morphologically rich language can be studied.

With respect to the engine, a grammar-based translator is developed to assure grammatically correct translations. We extend GF (Grammatical Framework, [12]) and write a new grammar for patent translation. The SMT system that complements the RBMT is based on Moses [9]. This system works on two different levels. First, it is used to build the parallel lexicon of the GF translator on the fly. Second, it is the top level decoder that takes the final decision about which phrases should be used.

In the following Section 2 describes recent work both in patent translation and hybrid systems. Section 2 explains our hybrid system and Section 3 evaluates its performance. Finally, Section 3 summarises the work and outlines possible lines to follow.

2. Related work

This work tackles two topics which are lately attracting the attention of researchers, patent translation and hybrid translators.

The high number of patents being registered and the necessity for these patents to be translated into several languages are the reason so that important efforts are being made in the last years to automate its translation between various language pairs. Different methods have been used for this task, ranging from SMT [1], [2] to hybrid systems [3], [4]. Besides full systems, various components associated to patent translation are being studied separately [7], [8], [9].

Part of this work is being done within the framework of two European projects, PLuTO (Patent Language Translations Online[9]) and MOLTO (Multilingual Online Translation[10]). PLuTO aims at making a substantial contribution to patent translation by using a number of techniques that include hybrid systems combining example-based and hierarchical techniques. On the other hand, one of MOLTO's use cases aims at extending a grammar-based translator with an SMT to gain robustness in the translation of patents. This paper is carried out within MOLTO.

HMT is not only useful in this context but is being applied in different domains and language pairs. Besides system combination strategies, hybrid models are designed so that there is one leading translation system assisted or complemented by other kinds of engines. This way the final translator benefits from the features of all the approaches. A family of models are based on SMT systems enriched with lexical information from RBMT [10], [11]. On the other side there are the models that start from the RBMT analysis and use SMT to complement it [12], [13], [14].

Our work can be classified in the two families. On the one hand, SMT helps on the construction of the RBMT translator but, on the other hand, there is the final decoding step to integrate translations and complete those phrases untranslated by RBMT. We use GF as rule-based system.

GF is a type-theoretical grammar formalism, mainly used for multilingual natural language applications. Grammars in GF are represented as a pair of an *abstract syntax* –an interlingua that captures the semantics of the grammar on a language-independent level, and a number of *concrete syntaxes* –representing target languages. There are also two main operations defined, *parsing* text to an abstract syntax tree and *linearising* trees into raw text. In this way one can translate between two target languages of the same multilingual grammar, by combining parsing and linearization.

The GF resource library [13] is the most comprehensive grammar for dealing with natural languages, as it features an abstract syntax which implements the basic syntactic operations such as predication and complementation, and 20 concrete syntax grammars corresponding to natural languages. This layered representation makes it possible to regard multilingual GF grammars as a RBMT system, where translation is possible between any pair of languages for which a concrete syntax exists. However, the translation system thus defined is first limited by the fixed lexicon defined in the grammar, and secondly by the syntactic constructions that it covers. For this reason, GF grammars have a difficult task in parsing free text. There is some recent work on parsing the Penn Treebank with the GF resource grammar for English [24], whereas the current work on patent translation is the first attempt to use GF for parsing un-annotated free text.

3. HMT system

The patent translator is a hybridisation between rule-based and statistical techniques. So, the final system is not only a combination of two different engines but the subsystems also mix different components. We have developed a GF translator for the specific domain that uses an in-domain SMT system to build the lexicon; an SMT system is on top of it to translate those phrases not covered by the grammar. In the following we describe the individual translators and the data used for their development.

[9]http://www.pluto-patenttranslation.eu/
[10]http://www.molto-project.eu/

3.1. Corpus

A parallel corpus in English and French has been gathered from the corpus of patents given for the CLEF-IP track in the CLEF 2010 Conference[11]. These data are an extract of the MAREC corpus, containing over 2.6 million patent documents pertaining to 1.3 million patents from the European Patent Office[12] (EPO). Our parallel corpus is a subset with those patents with translated claims and abstracts into the two languages. From this first subset we selected those patents that deal with the biomedical domain.

The final corpus built this way covers 56,000 patents out of the 1.3 million. That corresponds to 279,282 aligned parallel fragments extracted from the claims. A fragment is the minimum aligned segment in the two languages, so, it is shorter than a claim and, consequently, shorter than a sentence. The length of the fragments is variable and depends on the aligned units that can be extracted from the xml mark-up within the patent such as paragraph tags for example. Two small sets for development and test purposes have also been selected with the same restrictions: 993 fragments for development and 1008 for test.

3.2. In-domain SMT system

The first component is a standard state-of-the-art phrase-based SMT system trained on the biomedical domain with the corpus described in Section 3. Its development has been done using standard freely available software. A 5-gram language model is estimated using interpolated Kneser-Ney discounting with `SRILM` [6]. Word alignment is done with `GIZA++` [7] and both phrase extraction and decoding are done with the `Moses` package [8, 9]. Our model considers the language model, direct and inverse phrase probabilities, direct and inverse lexical probabilities, phrase and word penalties, and a non-lexicalised reordering. The optimisation of the weights of the model is trained with MERT [10] against the BLEU [11] evaluation metric.

A wider explanation of this system, the pre-process applied to the corpus before training the system and a deep evaluation of the translations can be found in España-Bonet et al. [2].

3.3. GF system

As explained in Section 2, the extension of GF to a new domain implies the construction of a specialised grammar that expands the general resource grammar. Since in our case of application we are far from a close and limited domain, some probabilistic components are also necessary. The general architecture is illustrated by Figure 5.2. A GF grammar-based system alone cannot parse most patent sentences. Consequently, the current translation system aims at using GF for translating patent chunks, and assemble the results in a later phase.

As a pre-process, claims are tagged with part-of-speech (PoS) with Genia [14], a PoS tagger trained on the biomedical domain. From the PoS-tagged words only the ones labelled as nouns, adjectives, verbs and adverbs are kept, since the GF library already has an extensive list of functional parts of speech such as prepositions and conjunctions. We use the extensive GF English lexicon[13] as a lemmatiser for the PoS-

[11]http://clef2010.org/

[12]http://www.epo.org/

[13]The GF English lexicon is based on the Oxford Advanced Learner's Dictionary, and contains around 50,000 English words.

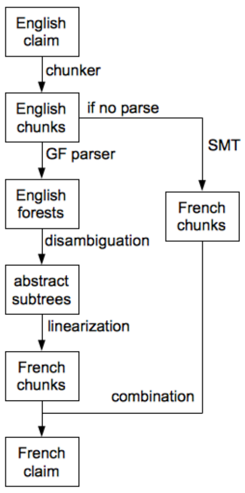


Figure 5.2: Architecture of the GF translation system.

tagged words, so that one can build their correspondent abstract syntax entry. Moreover, all the inflection forms of a given word are obtained from the same resource.

This process is made online. For every sentence to translate, the lexicon is enlarged with the corresponding vocabulary. The French version of the lexicon is built by translating the individual entries from the English lexicon (all inflection forms) with the SMT individual system trained on the patent corpus. The French translations are lemmatised with an extensive GF French lexicon, based on the large morphological lexicon Morphalou [23] in order to get their inflection table. The part-of-speech is assumed to be the same as in the English counterpart.

When this procedure is applied on the test set, the part-of-speech tagger is able to find 2,013 lexicon entries. However, due to part-of-speech mismatching or to the fact that a given word was not found in the SMT lexical table, 43.81% of the entries could not be translated to French.

In order to increase the coverage of the final GF translation, the grammar is adapted to deal with chunks instead of with full sentences. So, the source text is chunked into noun phrases (NP), adjective phrases (AP), adverbial and prepositional phrases (PP), relative pronouns (RP) and verb phrases (VP). Other kinds are ignored.

Some technical details have to be taken into account in order to build the patents grammar for chunks. Whereas NPs can be translated directly, a VP, RP or AP needs to have an NP to agree with, otherwise the GF grammar cannot know which linearisation form to choose. For NP and PP which can be translated independently, a mapping into corresponding GF categories is defined, whereas for VP, RP and AP, their GF mapping requires an NP in order to build their correspondent linearisation. If the required NP is not found, the chunk is sent to the SMT. Also, the VP category from the English and French GF resource grammars is implemented as a discontinuous category, so that it can handle discontinuous constituents in English and clitics in French. The patent grammar uses a category built on top of VP, which represents the flattened version of a VP, with all the constituents combined.

Because the syntactical structure of chunks is important in this case, a post-processing step is needed. This is meant to ensure that the PoS-tagging is consistent and that certain aspects captured in the grammar can be properly reflected in the claims. One can

Word	PoS Genia	Chunk Genia	PoS Final	Chunk Final
the	DT	B-NP	DT	B-NP
use	NN	I-NP	NN	I-NP
of	IN	B-PP	IN	I-NP
claim	NN	B-NP	NN	I-NP
1	CD	I-NP	CD	I-NP
,	,	O	,	O
wherein	IN	B-PP	RP	B-RP
said	V	B-VP	DT	B-NP
use	NN	B-NP	NN	I-NP
is	VBZ	B-VP	VBZ	B-VP
intramuscular	JJ	B-ADJP	JJ	I-VP
.	.	O	.	O

Table 5.6: Chunk detection for the example sentence Ex1.

see the importance of this step with an example.

Ex1 *The use of claim 1 , wherein said use is intramuscular .*

In the previous example, "said", a frequent used word in patent claims, acts as a definite article, whereas Genia tags it as a verb and therefore is it not merged with the following noun into a noun phrase. Moreover, the relative pronoun "wherein" is labelled as an adverb or noun phrase. The post-processing process updates the tags of certain entries and the tag of the following word, when needed.

Table 5.6 shows how the original tagging from Genia is converted into the correct GF parse chunks: *the use* (NP), *of claim 1* (PP), *wherein* (RP), *said use* (NP), *is intramuscular* (VP). As one can notice, chunks are merged when needed, like for the PP *of claim 1*, where the preposition was merged with the NP into a single chunk. The same goes for the VP chunk, as it is aimed to combine two-placed verbs or copulas with their objects before parsing.

GF parses the corresponding English chunks to obtain a forest of abstract syntax trees. In order to disambiguate among the possible options, all of them are linearised, looked up in the French corpus and the most frequent linearisation is kept as the best translation.

The translation sequence is done from left to right, so that the last-occurring NP is retained, and is used to make the agreement with VP, RP or AP. If no such NP can be found, or if the GF grammar is not capable to parse the one indicated by the chunker, the current chunk is passed to the SMT. In the working example, this in not necessary, and GF grammar alone obtains a translation for the full sentence:

1. *the use* $\rightarrow$ "l' utilisation" (NP)
2. *of claim 1* $\rightarrow$ "selon la revendication 1" (PP)
3. *wherein* $\rightarrow$ "dans laquelle" (RP agreeing with *"l' utilisation"*)
4. *said use* $\rightarrow$ "ladite utilisation" (NP)
5. *is intramuscular* $\rightarrow$ "est intramusculaire" (VP agreeing with *"ladite utilisation"*)

Finally, chunks are combined together with the punctuation marks, other non-included elements and untranslated chunks in the same order as in the source language.

	GF	**SMT**
NP	2,366 (14.9%)	2,199 (13.8%)
VP	275 (1.7%)	1,302 (8.2%)
AP	1,960 (12.3%)	1,935 (12.2%)
RP	648 (4.1%)	86 (0.5%)
Other	–	5,099 (32.0%)
Total	*5,301 (33.3%)*	*10,621 (66.7%)*

Table 5.7: Number and percentage of individual chunks translated by the HI system.

3.4. Top SMT layer

The grammar-based translator already makes use of the SMT system trained on patents to translate the GF English lexicon. This way, the vocabulary is disambiguated towards the biomedical domain, but still there are non-parseable chunks with unknown vocabulary in the lexicon that cannot be translated using the grammar.

To gain robustness in the final system, the output of the GF translator is used as *a priori* information for a higher level SMT system. The SMT baseline is fed with phrases which are integrated in two different ways. In both cases SMT leads the translation since it is the system that chooses the final reordering of the translation, GF constraints parts of the translation.

Hard Integration (HI): Phrases with GF translation are forced to be translated this way. The system can reorder the chunks and translates the untranslated chunks, but there is no interaction between GF and pure SMT phrases.

Soft Integration (SI): Phrases with GF translation are included in the translation table with a certain probability so that the phrases coming from the two systems interact. Probabilities in the SMT system are estimated from frequency counts in the usual way; the probabilities in the GF system are a fixed value in the interval [0, 1] for all the phrases. This probability is given to the chunk translation pair as a whole, so when competing with SMT translations that have four translation probabilities (phrase-to-phrase and word-to-word in the two directions) the probability mass is divided among them to combine the systems in the translation table. Notice that a probability of one for a phrase does not imply a sure translation not only because of this, but also because at the end, the language model chooses the translation.

4. Results and discussion

The complete hybrid system and the individual components introduced in Section 2 are evaluated on the patents test set both automatic and manually.

After the pre-process, the test set is divided in 15,922 chunks. From these chunks 33.3% can be translated using the GF patents grammar, and the remaining 66.7% must be passed to the SMT system. Table 5.7 shows the concrete percentages for every kind of chunk. Notice that GF only is designed to deal with the four most frequent types of chunks, and punctuation and conjunctions for example are ignored by GF. For these majority categories, GF can handle half of NP and AP, almost all RP but only 17.4% of VP.

	WER	PER	TER	BLEU	NIST	GTM-2	MTR-pa	RG-S*	ULC
GF	60.96	50.08	58.90	26.56	5.57	22.74	38.76	29.00	16.17
SMT	27.03	17.50	25.32	63.18	9.99	44.58	71.64	72.65	67.14
HI	33.56	21.95	31.24	55.88	9.24	38.81	67.30	67.80	58.84
SI1.0	26.76	17.39	25.10	63.56	10.02	**44.86**	**71.96**	72.89	67.56
SI0.5	**26.63**	**17.32**	**25.02**	**63.60**	**10.03**	44.84	71.94	**72.93**	**67.60**
SI0.0	27.08	17.48	25.36	63.15	9.99	44.54	71.60	72.66	67.11

Table 5.8: Automatic evaluation of the baselines and hybrid systems.

There are several reasons why GF cannot translate the chunks. In 18.3% of the cases the chunks could not be parsed by the GF English grammar. When parsed, 15.5% of the chunks could not be translated due to missing words in the bilingual lexicon and to a lesser extent 1.1% could not be translated because of the missing information about agreement. 31.3% of the chunks are labelled as *Other* (punctuation marks, item markers, etc.) and ignored by GF.

Splitting the sentences in chunks proved to be crucial for the final translation. 84.7% of the fragments to be translated contained at least one chunk that could not be parsed by the English grammar, and even more, 93.1% of the fragments contained at least one chunk that could not be translated. So, the coverage of a GF translation at sentence level would be of only 6.9%. At chunk level the coverage increases up to 33.3%.

Still this limited coverage cannot compete with that of a statistical system. Table 5.8 reports an automatic evaluation using several lexical metrics for both GF and SMT individual systems (top rows). This set of metrics is a subset of the metrics available in the Asiya evaluation package [16]. For all the metrics the SMT system beats the GF one in a significant way. This is mainly due to the coverage, SMT is able to translate the whole sentence which is not the case of GF. However, GF is able to deal with some grammatical issues that cannot be recovered statistically. The most evident example is agreement in gender and number. Contrary to English, French adjectives and nouns agree in gender and number and relative pronouns agree with their relative. This is taken into account by construction in GF so that mistaken SMT translations such as "le médicament séparée" is correctly translated as "le médicament séparé" (*the separate medicament*) or "composition pharmaceutique selon la revendication 1, dans lequel" is correctly translated as "composition pharmaceutique selon la revendication 1, dans lequelle" (*the pharmaceutical composition of claim 1, wherein*).

These are minor details from the point of view of the lexical evaluation metrics however, they make a difference to the reader. Although in few occasions the understanding of the sentence is compromised because of the lack of agreement, the fluency of the output is not harmed.

Therefore we incorporate these well-formed translations into the SMT system. A hard integration of the translations does not allow them to interact. GF translations are always used and the statistical decoder reorders them and completes the translation with its own phrase table. This system is named HI in Table 5.8. Results are below those of the SMT system because the system is being forced to use the high quality translations together with translations of elements not considered. Just to give an example, GF will highly benefit from incorporating a grammar to deal with compounds and numbers. Currently these elements typical of the domain are not specifically approached.

A softer integration of the translations is done by the family of systems denoted by

	SMT	Tied	SI0.5
Tester1	4	9	10
Tester2	3	13	7
Tester3	2	17	4
Tester4	6	5	12
Total	15	44	33

Table 5.9: Manual evaluation of the 23 different sentences from a random subset of 100 sentences.

GF	Une utilisation selon la revendication 3, dans laquelle le médicament séparé est administré at the same time as...
SMT	Utilisation selon la revendication 3, dans laquelle le médicament séparée est administré en même temps que...
HI	Une utilisation selon la revendication 3, dans laquelle le médicament séparé est administré en même temps que...
SI0.5	Utilisation selon la revendication 3, dans laquelle le médicament séparé est administré en même temps que...
Ref.	Utilisation selon la revendication 3, dans laquelle le médicament **séparé** est administré en même temps que...

Figure 5.3: Example where GF translates with the correct gender of the adjective and the SMT completes the untraslated words.

SI in Table 5.8. In this case, GF translations are given a probability which ranges from null to one with the same value given to all the phrases. Several experiments have been carried out for different values in the interval. We show in the bottom rows of Table 5.8 just three of them: 0, 0.5 and 1. Relative probabilities between the systems result not to be as important as the fact of allowing the interaction.

The combination of all the phrases improves the translations according to all the lexical metrics considered. There is an increment of 0.42 points of BLEU, 0.30 of TER and 0.46 of ULC, an uniform linear combination of 13 variants of the metrics considered. Improvements are moderate because of two reasons. First, SMT translations are already good for a start. Second, the amount of issues that GF handles are limited to be reflected on automatic metrics.

We have conducted a manual evaluation of the translations. To do this, 100 sentences have been randomly selected and four evaluators have been asked to indicate the grammatically most syntactically correct translation between two options: the SMT translation and the SI0.5 hybrid translation. The main aspects that we evaluated were correct agreement and properly inflected words.

For the whole testing corpus, 78.47% of the sentences were identically translated by the SMT and HMT. For our manually tested corpus, we only inspected the 23 sentences where the systems had a different output. The results can be seen in Table 5.9. The hybrid system is better than the SMT one according to the four evaluators, and the improvements come from discrepancies in gender, number and agreement. The SMT translations were preferred in the cases where the hybrid translation failed to translate certain words, so that the final claim has a visible hole –which makes it syntactically incorrect.

Figure 5.3 shows an example sentence where these features are observed. GF is doing the gender agreement between noun and adjective correctly ("séparée" vs. "séparé") but is not able to translate the full sentence ("at the same time as"). The two hybrid systems in this case are able to construct the correct translation which coincides with the reference.

5. Conclusions and future work

This work presents a HMT system for patent translation. The system exploits the high coverage of statistical translators and the high precision of GF to deal with specific issues of the language.

At this moment the grammar tackles agreement in gender, number and between chunks, and reordering within the chunks. Although the cases where these problems apply are not extremely numerous both manual and automatic evaluations consistently show their preference for the hybrid system in front of the two individual translators.

The coverage of the grammar can be extended in order to deal with more typical structures present in patent documents. The coverage of VP is particularly low because of the missing verbs from the French lexicon and the syntactically complex verb phrases –such as cascades of nested verbs, which are not handled by the patents grammar yet. Also, a grammar to translate compounds will be included as they are a significant part of the biomedical documents. Moreover, the grammar component can be extended to handle the ordering at sentence level besides of the reordering within the chunks. This is specially interesting to deal with languages like German where the structure of the sentence is different from the structure in English for example.

The previous improvements will increase the number of chunks that can be parsed by the grammar; in order to increase the percentage of translations it is also necessary to improve the lexicon building procedure. An obvious improvement would be a bilingual dictionary of idioms, so that the translation would not just map word-to-word, but also phrase-to-phrase.

Finally, we plan to implement another version of the hybrid system where GF grammars are applied at an later stage –after the English chunks are translated into French by the SMT system. The GF grammars will be used to to restore the agreement for chunks like VP, RP and AP, like before. The main difference is that due to an earlier use of SMT, one can capture idiomatic constructions better, and use GF just in the end for improving syntactic correctness.

Acknowledgements

This work has been partially funded by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement number 247914 (MOLTO project, FP7-ICT-2009-4-247914).

Bibliography

[1] Ranta, A.: Grammatical Framework: Programming with Multilingual Grammars. CSLI Publications, Stanford (2011) ISBN-10: 1-57586-626-9 (Paper), 1-57586-627-7 (Cloth).

[2] Koehn, P., Hoang, H., Mayne, A.B., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., Herbst, E.: Moses: Open source toolkit for statistical machine translation. In: Annual Meeting of the Association for Computation Linguistics (ACL), Demonstration Session. (Jun 2007) 177–180

[3] Ceausu, A., Tinsley, J., Way, A., Zhang, J., Sheridan, P.: Experiments on Domain Adaptation for Patent Machine Translation in the PLuTO project. Proceedings of the 15th Annual Conference of the European Association for Machine Translation (EAMT 2011) (2011)

[4] España-Bonet, C., Enache, R., Slaski, A., Ranta, A., Màrquez, L., Gonzàlez, M.: Patent translation within the molto project. In: Proceedings of the 4th Workshop on Patent Translation, MT Summit XIII, Xiamen, China (sep 2011) 70–78

[5] Ehara, T.: Rule based machine translation combined with statistical post editor for japanese to english patent translation. MT Summit XI Workshop on patent translation, 11 September 2007, Copenhagen, Denmark (2007) 13–18

[6] Ehara, T.: Statistical Post-Editing of a Rule-Based Machine Translation System. Proceedings of NTCIR-8 Workshop Meeting, June 15–18, 2010 (2010) 217—-220

[7] Sheremetyeva, S.: Natural language analysis of patent claims. In: Proceedings of the ACL-2003 workshop on Patent corpus processing - Volume 20. PATENT '03, Stroudsburg, PA, USA, Association for Computational Linguistics (2003) 66–73

[8] Sheremetyeva, S.: Less, Easier and Quicker in Language Acquisition for Patent MT. MT Summit X, Phuket, Thailand, September 16, 2005, Proceedings of Workshop on Patent Translation (2005) 35–42

[9] Sheremetyeva, S.: On Extracting Multiword NP Terminology for MT. EAMT-2009: Proceedings of the 13th Annual Conference of the European Association for Machine Translation, ed. Lluís Màrquez and Harold Somers, 14-15 May 2009, Universitat Politècnica de Catalunya, Barcelona, Spain (2009) 205–212

[10] Eisele, A., Federmann, C., Saint-Amand, H., Jellinghaus, M., Herrmann, T., Chen, Y.: Using moses to integrate multiple rule-based machine translation engines into a hybrid system. In: Proceedings of the Third Workshop on Statistical Machine Translation. StatMT '08 (2008) 179–182

[11] Chen, Y., Eisele, A.: Hierarchical hybrid translation between english and german. In Hansen, V., Yvon, F., eds.: Proceedings of the 14th Annual Conference of the European Association for Machine Translation, EAMT, EAMT (5 2010) 90–97

[12] Habash, N., Dorr, B., Monz, C.: Symbolic-to-statistical hybridization: extending generation-heavy machine translation. Machine Translation **23** (2009) 23–63

[13] Federmann, C., Eisele, A., Chen, Y., Hunsicker, S., Xu, J., Uszkoreit, H.: Further experiments with shallow hybrid mt systems. In: Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR. (July 2010) 77–81

[14] España-Bonet, C., Labaka, G., Díaz de Ilarraza, A., Màrquez, L., Sarasola, K.: Hybrid machine translation guided by a rule-based system. In: Proceedings of the 13th Machine Translation Summit, Xiamen, China (sep 2011) 554–561

[15] Ranta, A.: The GF resource grammar library. Linguistic Issues in Language Technology **2**(1) (2009)

[16] Angelov, K.: The Mechanics of the Grammatical Framework. PhD thesis, Chalmers University of Technology, Gothenburg, Sweden (2011)

[17] Stolcke, A.: SRILM – An extensible language modeling toolkit. In: Proc. Intl. Conf. on Spoken Language Processing. (2002)

[18] Och, F.J., Ney, H.: A systematic comparison of various statistical alignment models. Computational Linguistics **29**(1) (2003) 19–51

[19] Koehn, P., Shen, W., Federico, M., Bertoldi, N., Callison-Burch, C., Cowan, B., Dyer, C., Hoang, H., Bojar, O., Zens, R., Constantin, A., Herbst, E., Moran, C.: Open Source Toolkit for Statistical Machine Translation. Technical report, Johns Hopkins University Summer Workshop. http://www.statmt.org/jhuws/ (2006)

[20] Och, F.J.: Minimum error rate training in statistical machine translation. In: Proc. of the Association for Computational Linguistics, Sapporo, Japan (July 6-7 2003)

[21] Papineni, K., Roukos, S., Ward, T., Zhu, W.J.: Bleu: a method for automatic evaluation of machine translation. In: Proceedings of the Association of Computational Linguistics. (2002) 311–318

[22] Tsuruoka, Y., Tateishi, Y., Kim, J., Ohta, T., McNaught, J., Ananiadou, S., Tsujii, J.: Developing a robust part-of-speech tagger for biomedical text. In Bozanis, P., Houstis, E.N., e., eds.: Advances in Informatics. Volume 3746. Springer Berlin Heidelberg (2005) 382–392

[23] Romary, L., Salmon-Alt, S., Francopoulo, G.: Standards going concrete: from lmf to morphalou. In: Proceedings of the Workshop on Enhancing and Using Electronic Dictionaries. ElectricDict '04, Stroudsburg, PA, USA, Association for Computational Linguistics (2004) 22–28

[24] Giménez, J., Màrquez, L.: Asiya: An Open Toolkit for Automatic Machine Translation (Meta-)Evaluation. The Prague Bulletin of Mathematical Linguistics (94) (2010) 77–86

3 Hybrid Translation for European Biomedical Patents

Cristina España-Bonet Ramona Enache
Erzsébet Galgóczy Aarne Ranta Lluís Màrquez

Abstract: Patent translation is of great importance for patent agencies where some parts of the documents must be available in several languages. For European patents these languages include English, French and German. We present and compare here the translation engines developed for translating English patents into French and German. The translators include a statistical system (SMT) that assures a full coverage, an interlingua engine with statistical components based on Grammatical Framework (GF) that assures grammatically correct translations, and a hybrid system (HMT) combination of both. Biomedical patents are chosen as the domain where to develop and test the systems. Due to its limited coverage, the pure GF system showed the lowest performance, whereas the HMT gave slight improvements over pure SMT.

1. Introduction

Patent classification, retrieval and translation are an important focus in nowadays research. The high number of patents being registered in the patent agencies and the necessity for these patents to be translated into several languages are a motivation to automate its translation between various language pairs. Different methods are used for this task, ranging from statistical machine translation systems [1, 2] to hybrid systems [3, 4, 5].

In the last years two European projects have been devoted to patent analysis: PLuTO (Patent Language Translations Online[14]) and MOLTO (Multilingual Online Translation[15]). PLuTO developed hybrid systems combining example-based and hierarchical techniques. On the other hand, one of MOLTO's use cases aimed at extending a grammar-based translator with an SMT to gain robustness in the translation of patents.

Hybrid translation systems can be especially appropriate for patent translation. The language of patents, although having a large amount of vocabulary and richness of grammatical structure, still uses a formal style that can be interpreted by a grammar. Parallel corpora in the official languages are also available so that SMT systems output fluent translations with a wide coverage. With these premises it seems natural to combine a rule based system with statistical components. This work follows this approach. We enrich a GF grammar to deal with the translation of patents in the biomedical domain and train an in-domain SMT system. A full hybrid system with components of the two individual ones is also developed.

The paper is organised as follows. Section 3 introduces the basic resources and individual systems to be applied to patent translation. Section 3 presents the architecture of the translation engines here developed and Section 3 analyses the results and compares them with other devoted translators. Finally Section 3 draws the conclusions.

[14]http://www.pluto-patenttranslation.eu/
[15]http://www.molto-project.eu/

2. Basic Resources and Systems

2.1. Corpus

A parallel corpus on the selected domain is needed in order to train the statistical system. For our experiments we use a part of the Marec corpus made available by the CLEF Initiative[16].

There are 119,337 documents with IPC code A61P, the one we chose to represent the biomedical domain. From these documents we select the aligned fragments in every patent with trilingual claims. That leads to 281,283 aligned parallel segments and about 8 million tokens (7,954,491 tokens for English, 7,346,319 for German and 8,906,379 for French).

Besides a standard cleaning of the corpus, the detection and correct tokenisation of chemical compounds has been shown to be crucial in the performance of translators as explained in [2]. The same preprocess and tokenisation is used in this work for all the systems.

2.2. SMT System

We build a state-of-the-art phrase-based SMT system trained on the biomedical domain with the corpus described in Section 3. Its development has been done using standard freely available software. A 5-gram language model is estimated using interpolated Kneser-Ney discounting with `SRILM` [6]. Word alignment is done with `GIZA++` [7] and both phrase extraction and decoding are done with the `Moses` package [8, 9]. The optimisation of the feature weights of the model is done with Minimum Error Rate Training (MERT) [10] against the BLEU [11] evaluation metric. Our model considers the language model, direct and inverse phrase probabilities, direct and inverse lexical probabilities, phrase and word penalties, and a non-lexicalised reordering.

As a byproduct of training the SMT system lexical and phrase tables are obtained. The word-to-word translations are deduced from the alignments given by `GIZA++`. The final translation tables are also obtained from these alignments, using the heuristic "grow-diag-final" and the phrase extraction script in `Moses`. These resources are important to build the patent's lexicon that uses the GF engine.

2.3. GF Translation System

GF is a grammar formalism designed for multilingual grammars [12]. It represents grammars as a pair of an *abstract syntax* —interlingual representation of the semantics and a number of *concrete syntaxes*—representing target languages implementing the abstract syntax. The two main operations performed on GF grammars are *parsing* text to an abstract syntax tree and *linearising* trees into a target language.

The largest and most comprehensive GF application is the resource grammar library [13] which describes the basic syntax of 27 languages. Using this grammar, one can develop application grammars easily, by using the language primitives from the resource grammar corresponding to a given language.

By combining parsing and linearisation, one can achieve automatic translation between any pair of languages of a GF grammar. The advantage is that the translation will be syntactically correct, as each concrete syntax implements the language-specific issues separately. However, the approach is limited by the coverage of the grammar

[16]`http://www.clef-initiative.eu/`

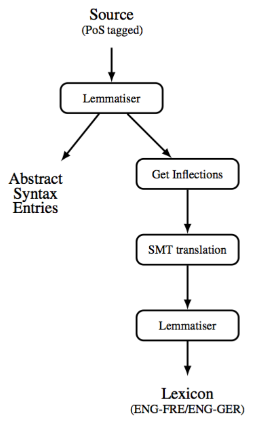


Figure 5.4: The input is preprocessed and used to build the lexicon and mark the chunks to translate.

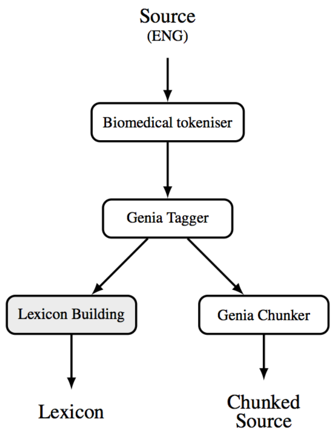


Figure 5.5: Details of the lexicon building process.

and it is difficult to translate free text. The addition of statistical components in patent translation has been the first attempt to use GF for parsing free text [2].

3. Hybrid System

3.1. GF Architecture

This work is a continuation of a basic architecture described in [5]. Among some improvements we point out the fact that a semi-automatic technique for lexicon acquisition has been replaced with a number of automated methods. In addition to this, the original system that translated English claims to French has been adapted to translate claims to German, dealing in this way with the three official languages of the EPO.

Figures 5.4 and 5.5 show the main modules of the GF translator developed for patents. The first thing to notice is the fact that the GF translator works at chunk level due to the difficulty to parse full claims. All the necessary preprocess before really translating a chunk is depicted in Fig. 5.4. After tokenising the text, claims are tagged with part-of-speech (PoS) with `Genia` [14], a PoS tagger trained on the biomedical domain. The output of the tagger is used for two purposes, as input for the chunker and as the source to build the lexicon.

From the PoS-tagged words only the ones labelled as nouns, adjectives, verbs and adverbs are kept, since the GF library already has an extensive list of functional PoS such as prepositions and conjunctions. The extensive GF English lexicon[17] is used as a lemmatiser for the PoS-tagged words, so that one can build their correspondent abstract syntax entry (Figure 5.5). Moreover, all the inflection forms of a given word are obtained from the same resource. This process is made online for most of the systems. For every sentence to translate, the lexicon is enlarged with the corresponding vocabulary. The French/German version of the lexicon is built by translating the individual entries from the English lexicon (all inflection forms) with the SMT individual system trained on the patent corpus. The translations are lemmatised with an extensive GF

[17]The GF English lexicon is based on the Oxford Advanced Learner's Dictionary, and contains around 50,000 English words.

lexicon in order to get their inflection table. The PoS is assumed to be the same as in the English counterpart.

3.2. Domain-Specific Grammar Structure

The patent grammar has concrete syntaxes for English, French and German and is build on the resource grammars with some additions for dealing with specific features of the patent domain and with chunk translation. These additions include 4 new categories for parsing chunks (`Parse_AdjP, Parse_AdvP, Parse_VP, Parse_NP`), 3 for agreement among them (`NP_VP, NP_AdjP, NP_RelP`) and aslo 8 functions that deal with these agreements and the parsing of chunks. 9 extensions to the resource grammar dealing with general-purpose constructions have been included. Examples of these constructions with descriptive names are `Nominalisation` ("*recovering* the antibodies"), `Aggregation` (*mouse antibody* – antibody of mouse), `Adjectivisation` ("*immunised* mouse") or `GerundAsAdj` ("*following* states"). Finally the grammar contains 20 constructions to deal with claim-specific data.

3.3. Domain-Specific Lexicon

We devise several ways of automating the lexicon acquisition process. For this, one needs to construct a base dictionary from which the final lexicon is built. Two different dictionaries are created:

Core Lexicon. It is made of 198 words. These elements are selected as the most common English words appearing in the training corpus used in the SMT system. The English words are translated to French and German and, since it is a small dictionary, it has been manually checked assuring a correct translation. The dictionary contains only single words, not multiple mappings or phrases.

Static Lexicon. The methods for acquiring the lexicon differ for French and German. For French, the lexicon contains 3,983 words and is built from the SMT translation tables. It is constructed by looking for the most likely translation of the English words in the translation tables. Contrary to the core lexicon, the static one contains one-to-many translations.

For German, the static lexicon is built using the SMT word alignments. From the aligned word-to-word pairs, the ones with the highest respective probabilities are selected and lemmatised using the GF dictionary and smart paradigms[18]. In the same step, word pairs get matched with a previously SMT-built English lexicon based on the patents corpus; so in the end, we are left with mappings from said English lexicon to entries from the German GF dictionary. A dictionary extracted from Wiktionary is also used to finally build a static lexicon with 43,084 entries.

Dictionary of German Compounds. The lexicons just described contain one-to-one and one-to-many translations. While this is a workable approach in translating English to French, the mappings are ambiguous in the case of English to German. For example, the English phrase *nucleotide sequence* translates to German as *Nucleotidsequenz*. If one attempts a one-to-one mapping, one inevitably ends up with *nucleotide* $\rightarrow$ *Nucleotidsequenz* and *sequence* $\rightarrow$ *Nucleotidsequenz*.

There are two accepted ways to deal with the problem [15]. First, on can split German compounds in the original corpus in order to make the one-to-one mapping possible and rejoin the constituents in a later step. Alternatively, one can create new "single-

[18]A smart paradigm is GF function that takes the basic form of a word and builds the whole GF representation, by using a set of rules for inferring the other forms.

	WER	PER	TER	BLEU	NIST	GTM2	MTRst	CPO_{c}	CPO_{p}	SPO_{p}	SP$_{\mathrm{pN}}$	ULC
GF-Sts	63.58	54.09	62.30	24.22	5.17	21.39	35.34	24.34	25.53	29.91	3.59	64.46
GF-SaBs	64.73	49.77	62.58	24.24	5.36	20.82	38.15	29.04	30.33	34.74	4.15	70.94
GF-SaEs	**63.43**	48.63	**61.30**	**25.50**	**5.55**	**21.59**	**39.92**	29.80	31.10	**35.97**	**4.27**	74.01
GF-UnBs	64.74	49.77	62.59	24.24	5.36	20.82	38.15	29.04	30.33	34.75	4.15	70.93
GF-UnEs	**63.43**	**48.62**	61.31	**25.50**	**5.55**	**21.59**	**39.92**	**29.81**	**31.11**	**35.97**	**4.27**	**74.02**

Table 5.10: GF translation for the English-to-French language pair.

	WER	PER	TER	BLEU	NIST	GTM2	MTRst	CPO_{c}	CPO_{p}	SPO_{p}	SP$_{\mathrm{pN}}$	ULC
GF-Sts	80.75	73.11	80.07	17.11	3.71	15.85	25.57	15.01	14.21	7.32	2.90	50.74
GF-SaBs	**73.45**	**62.57**	**72.01**	**21.67**	**4.80**	**19.44**	**34.69**	22.14	20.50	20.50	3.91	**75.71**
GF-SaEs	74.70	64.04	73.59	20.70	4.61	18.74	33.21	21.03	19.80	19.80	**3.93**	72.71
GF-UnBs	73.46	62.58	72.02	**21.67**	4.79	**19.44**	34.68	**22.15**	**20.52**	**20.52**	3.91	**75.71**
GF-UnEs	74.70	64.04	73.59	20.70	4.61	18.74	33.21	21.03	19.80	19.80	**3.93**	72.71

Table 5.11: GF translation for the English-to-German language pair.

word" compound entries for English, which can be mapped to the German compound. In our approach we decided to go with the first method by applying a greedy method of looking up the German compound or its substrings matching to compounding words in the dictionary and splitting them if there was a match. With this process a dictionary with 7,774 entries is built.

Dynamic lexicon acquisition. In addition to the static lexical resources, different variants of the main architecture use different automated lexicon acquisition techniques:

Static. Uses the static dictionaries for French and German and does not add new lexical items ($GF\text{-}Sts$).

Runtime safe. Uses a base lexicon (core, $GF\text{-}SaBs$, or static, $GF\text{-}SaEs$) to start with and adds the lexical items that are not present there. The method assumes that the English words are found in the English monolingual dictionary and that the target translation is found in the corresponding target dictionary and has the right PoS tag. We assigned a confidence score to each of the translated chunks. The score is 4 for unambiguous translations and 1 for the ambiguous ones.

Runtime unsafe. Uses a base lexicon (core, $GF\text{-}UnBs$, or static, $GF\text{-}UnEs$) to start with and adds the lexical items that are not present there. The method adds words regardless whether they are found in the monolingual dictionaries and builds the representation with the help of smart paradigms. This approach is used for nouns, adjectives and adverbs. As before, chunk translations are labeled with confidence scores: 4 for translations that are not ambiguous and contain only "safe" words from dictionaries, 1 for translations that are "safe" but ambiguous, and 0.2 for translations that contain at least an "unsafe" word. Dependent "safe" elements such as RelP, AdvP and VP which depend on an NP are labeled as 0.2 in this system.

3.4. Final SMT Decoding

As it has been seen, the grammar-based translator already makes use of the SMT system trained on patents to translate the GF English lexicon. Although it already uses of hybridisation techniques, we consider this first approximation as a baseline for the more advanced hybrid systems. The reason is that even the vocabulary is disambiguated towards the biomedical domain thanks to the hybridisation, still there are non-parseable chunks with unknown vocabulary in the lexicon that cannot be translated using the

	WER	PER	TER	BLEU	NIST	GTM2	MTRst	CPO_c	CPO_p	SPO_p	SP$_{pN}$	ULC
GF-SaEs	63.43	48.63	61.30	25.50	5.55	21.59	39.92	29.80	31.10	35.97	4.27	34.93
SMT	27.16	**18.00**	25.57	62.40	9.95	44.99	75.71	63.71	65.91	47.54	3.60	78.81
SI-Sts	27.47	18.25	25.88	62.17	9.93	45.01	75.86	63.18	65.69	70.14	7.53	85.92
SI-SaBm	26.96	18.14	25.54	62.50	9.94	45.31	75.83	63.32	65.77	70.14	7.58	86.26
SI-SaEm	27.24	18.54	25.79	62.44	9.93	45.09	75.84	62.71	65.47	69.49	7.58	85.84
SI-UnBm	27.12	18.26	25.58	62.35	9.93	44.98	75.81	63.13	65.78	69.92	7.57	86.05
SI-UnEm	27.09	18.39	25.66	62.56	9.94	45.26	75.95	62.93	65.62	69.80	7.59	86.08
Google 2011	36.17	24.08	34.32	53.46	9.20	36.55	67.73	54.57	56.88	62.45	6.94	73.89
Google 2012	32.46	20.82	30.54	57.86	9.70	40.28	71.56	60.17	62.56	45.48	3.46	73.19
Google 2013	**26.15**	18.05	**24.61**	**66.47**	**10.39**	**46.77**	**77.81**	**65.22**	**68.38**	**72.75**	**7.85**	**89.23**

Table 5.12: Comparative table for the English-to-French language pair.

	WER	PER	TER	BLEU	NIST	GTM2	MTRst	CPO_c	CPO_p	SPO_p	SP$_{pN}$	ULC
GF-SaBs	73.45	62.57	72.01	21.67	4.80	19.44	34.69	22.14	20.50	20.50	3.91	31.24
SMT	**30.93**	**22.82**	**29.33**	57.59	9.40	42.98	67.84	52.89	51.62	51.62	**6.95**	85.03
SI-Sts	32.95	23.83	31.35	56.02	9.22	41.73	66.18	50.53	48.49	48.49	6.75	81.89
SI-SaBm	32.95	23.94	31.43	56.00	9.19	41.89	66.17	51.35	49.76	49.76	6.81	82.47
SI-SaEm	32.63	23.36	30.98	56.59	9.23	42.12	66.94	51.69	50.39	50.39	6.81	83.17
SI-UnBm	32.58	24.50	31.16	56.06	9.21	42.25	65.83	51.67	50.27	50.27	6.87	82.81
SI-UnEm	32.94	23.73	31.39	56.28	9.19	42.16	66.40	51.75	50.04	50.04	6.80	82.77
Google 2011	48.59	34.91	46.53	45.22	8.04	32.45	57.17	41.37	38.08	38.08	5.63	64.47
Google 2012	40.20	26.99	38.29	53.43	9.01	39.89	63.34	49.88	47.23	**65.23**	6.23	79.65
Google 2013	33.25	23.22	31.78	**61.65**	**9.85**	**45.79**	**68.72**	**55.35**	**52.36**	52.36	6.55	**86.17**

Table 5.13: Comparative table for the English-to-German language pair.

grammar. That is to say, the system is not able to translate robustly a whole test set. The percentage of sentences that can be completely translated from beginning to end by GF is 6.9%.

To gain robustness in the final system the output of the GF translator is used as *a priori* information for a higher level SMT system. An SMT system trained in the same way as the SMT baseline is fed with these GF phrases, both in the development process and at decoding time. In our main system, we allow the phrases to interact, and therefore, we name it as a Soft Integration (SI).

Phrases with GF translation are included in the translation table with a certain probability so that the phrases coming from the two systems interact. Probabilities in the SMT system are estimated from frequency counts in the usual way; the probabilities in the GF system are assigned to divide phrases in two or three groups according to the confidence on the translation as explained above. The GF score is divided homogeneously among the four lexical features of the translation table. So, a score of 4 means a score of 1 for each feature (generative and discriminative lexical translation probabilities, and generative and discriminative translation models). This value does not imply a sure translation because other features such as language model, penalties and reordering also interact, but it is given a high value so that SMT phrases at most can tie. In the same way, a score of 1 is converted into a 0.25 for each feature, and the most unsafe translations, those with score 0.2, have a minimum contribution of 0.05 per feature. This small value makes them very difficult to be chosen but still complement the translation in case there is not better choice coming from the SMT.

Several GF translations for every chunk are available. Notice that the final translation table contains both grammatical and non-grammatical phrases coming from the SMT system and only grammatical phrases or chunks from the GF one.

4. Results and Discussion

We evaluate the performance of three different translators for biomedical patents: i) the SMT in-domain system, ii) the GF with automated lexicon acquisition and iii) the complete hybrid system.

In order to do a complete analysis with automatic metrics we do not use a single metric such as the standard BLEU but an heterogeneous set. The `Asiya` evaluation package [16] is used for this purpose. From the full set of metrics available for English, French and German we select a representative subset build up with:

Lexical metrics: PER [17], TER [18] and WER [19] based on edit distances; BLEU and NIST [20], based on n-gram matching; and GTM [21] and METEOR [22], based on the F-measure.

Syntactic metrics [23]: SPO_{p} and SP$_{pN}$, based on the lexical overlap or the NIST score over the *p*art-of-speech (Shallow Parsing); and CPO_{p}, and CPO_{c}, based on the lexical overlap among *p*art-of-speech or *c*onstituents of constituent parse trees.

The Uniform Linear Combination (ULC) of metrics used in the evaluation considers all these metrics.

Tables 5.10 and 5.11 show the automatic evaluation of the GF translator for French and German respectively. The nomenclature of the five systems is that defined in the previous section, and the s at the end of the names indicates that one considers single GF translations: from the all the translation options generated, only the most frequent in the corpus is kept. Scores associated to each chunk have no effect at this point.

Translation with a static lexicon (GF-Sts) is significantly below those with runtime lexicons. 4,877 and 2,857 chunks can be translated with a static lexicon for French and German respectively, whereas if one builds the dynamic lexicon on top of it the number grows up to 7,150 and 6,784. So, GF-Sts is worse because it translates less chunks.

For dynamic lexicons two conclusions can be extracted. First, runtime safe and unsafe make no difference at all. The number of chunks translated by the two versions is almost the same and confidence scores do not have any effect here. Second, building the lexicon on top of the base one (GF-$xxBs$) is better for German, but starting from the static lexicon (GF-$xxEs$) is better for French. In both cases the positive difference is of 1 point of BLEU and more than 3 of ULC for example.

10,306 chunks are detected by `Genia`, plus 3,107 punctuation elements. So, approximately half of the tokens are translated by GF alone. There might be several reasons why GF cannot translate the chunks. First and most important, in some cases the chunks cannot be parsed by the GF English grammar. When parsed, there might be a lack of information about agreement or, in case of dynamic lexicons, missing words in the bilingual lexicon. As an example, let's see the effect on the GF-$UnBs$ system. In this case 3,449 chunks cannot be translated. In a 82% of the cases chunks cannot be translated because of the parsing, a 13% is due to the missing entries in the lexicons and the remaining 5% is due to missing the agreement.

To complement these untranslated chunks we combine the GF translations with the SMT ones. The statistical system by its own has a high performance (see for example BLEUs close to 60 in Tables 5.12 and 5.13). The domain can be considered restricted for a SMT and the most repetitive structures are correctly translated. These structures are also covered by GF, so, in most cases translations coming from the two individual systems coincide. Within the phrases selected in the final translation of the full hybrid systems (SI-$xxxx$) only between 5-7% of the common chunks are uniquely translated by GF, 45-60% are translations only seen in the SMT system and between 35-50% appear in both. With this low contribution of GF in the final translation, SMT and hybrid are close one to each other. For English-to-French translation the hybrid system is globally better whereas for English-to-German the statistical is.

In the final decoding of the full hybrid, the language model which, together with the word penalty, has the highest weight in the final score of a translation, is favouring translations with structures similar to those seen in the training data. So, when the final decoding has freedom to choose, it reproduces the most frequent structures. In some cases that simply favours a correct SMT translation in front of another correct GF translation; in some others, a correct translation is lost because the language model is not seeing at large distances.

The reordering between chunks/phrases is also done by the SMT decoder, only the reordering intra-chunk is done by GF. The following natural step is to allow the grammar to reorder the chunks, and this implies a GF capable of parsing full sentences. Ongoing work is being done for robust parsing in GF [24]. Other improvements are related to the coverage of structures by the grammar and the lexicon.

The tables of results show a comparison with Google Translate[19] too. We have translated the test set at three different periods distant one year, February 2011, April 2012 and May 2013. The hybrid translator is better than Google 2011 and Google 2012, but the current Google translator is the best system for almost all the metrics both for German and French, with the exception of edit distance metrics which favour our in-domain SMT system. Google signed an agreement with the EPO in March 2011 so, their training corpus might already contain EPO patents at this time and the comparison between systems might not be completely fair.

A similar thing happens with the PLuTO translator. It is a hybrid data-driven system built following established design patterns, with an extensible framework allowing for the interchange of novel or previously developed modules as it is defined in [25]. The project dealt with English, French, German, Spanish, Portuguese, Chinese and Japanese. For the languages we are interested in, they have at their disposal the Marec corpus and EPOs translation memories and therefore we have not included their translations in our analysis.

5. Conclusions

This work presents and analyses several translation engines developed for patent translation in the biomedical domain. Given that there is enough parallel corpus available, the SMT system showed to be the most appropriate in terms of performance/effort. However, a manual inspection of the translations confirms that some aspects such as agreement and concordance are better covered by the GF system than by the SMT one. In general, these issues do not damage the understanding of the sentence, but patent translation needs of high precision translation, without any room for ambiguities generated by incorrect translations.

This has been the main aim to build a hybrid translator between GF and SMT. The current HMT system uses more than a 90% of phrases from the statistical system. This is possibly due to the strength that has the language model in the final decoding, and to the fact that it has been estimated in the same corpus where the SMT system has been trained. Besides, the reference translations are also gathered from the Marec corpus, so, at the end, all the automatic mechanisms try to reproduce the specific language of the Marec data.

[19] `http://translate.google.com`

A key element to improve the hybrid system with the GF grammaticality is a robust parsing that allows to keep the GF structure of the source sentence in the translation. This would specially benefit the English-to-German translation, where the SMT system is better. Increasing the number of structures covered by the grammar and the entries in the lexicon would also provide the final system with more grammatical chunks. Regarding the final SMT decoding, the addition of linguistic features could also help in the same direction. This includes considering features such as PoS and the kind of chunk. Features with information on the origin system of every translation option and additional language models for PoS and chunks can also favour grammatically correct output.

Acknowledgements

This work has been partially funded by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement number 247914 (MOLTO project, FP7-ICT-2009-4-247914).

Bibliography

[1] Ceausu, A., Tinsley, J., Way, A., Zhang, J., Sheridan, P.: Experiments on Domain Adaptation for Patent Machine Translation in the PLuTO project. Proceedings of the 15th Annual Conference of the European Association for Machine Translation (EAMT 2011) (2011)

[2] España-Bonet, C., Enache, R., Slaski, A., Ranta, A., Màrquez, L., Gonzàlez, M.: Patent translation within the MOLTO project. In: Proceedings of the 4th Workshop on Patent Translation, MT Summit XIII, Xiamen, China (sep 2011) 70–78

[3] Ehara, T.: Rule based machine translation combined with statistical post editor for japanese to english patent translation. MT Summit XI Workshop on patent translation, 11 September 2007, Copenhagen, Denmark (2007) 13–18

[4] Ehara, T.: Statistical Post-Editing of a Rule-Based Machine Translation System. Proceedings of NTCIR-8 Workshop Meeting, June 15-18, 2010 (2010) 217–220

[5] Enache, R., España-Bonet, C., Ranta, A., Màrquez, L.: A hybrid system for patent translation. In: Proceedings of the 16th Annual Conference of the European Association for Machine Translation (EAMT12), Trento, Italy (may 2012) 269–276

[6] Stolcke, A.: SRILM – An extensible language modeling toolkit. In: Proc. Intl. Conf. on Spoken Language Processing. (2002)

[7] Och, F.J., Ney, H.: A systematic comparison of various statistical alignment models. Computational Linguistics **29**(1) (2003) 19–51

[8] Koehn, P., Shen, W., Federico, M., Bertoldi, N., Callison-Burch, C., Cowan, B., Dyer, C., Hoang, H., Bojar, O., Zens, R., Constantin, A., Herbst, E., Moran, C.: Open Source Toolkit for Statistical Machine Translation. Technical report, Johns Hopkins University Summer Workshop. http://www.statmt.org/jhuws/ (2006)

[9] Koehn, P., Hoang, H., Mayne, A.B., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., Herbst, E.: Moses: Open source toolkit for statistical machine translation. In: Annual Meeting of the Association for Computation Linguistics (ACL), Demonstration Session. (Jun 2007) 177–180

[10] Och, F.J.: Minimum error rate training in statistical machine translation. In: Proc. of the Association for Computational Linguistics, Sapporo, Japan (July 6-7 2003)

[11] Papineni, K., Roukos, S., Ward, T., Zhu, W.J.: Bleu: a method for automatic evaluation of machine translation. In: Proceedings of the Association of Computational Linguistics. (2002) 311–318

[12] Ranta, A.: Grammatical Framework: Programming with Multilingual Grammars. CSLI Publications, Stanford (2011) ISBN-10: 1-57586-626-9 (Paper), 1-57586-627-7 (Cloth).

[13] Ranta, A.: The GF resource grammar library. Linguistic Issues in Language Technology **2**(1) (2009)

[14] Tsuruoka, Y., Tateishi, Y., Kim, J., Ohta, T., McNaught, J., Ananiadou, S., Tsujii, J.: Developing a robust part-of-speech tagger for biomedical text. In Bozanis, P., Houstis, E.N., e., eds.: Advances in Informatics. Volume 3746. Springer Berlin Heidelberg (2005) 382-392

[15] Popović, M., Stein, D., Ney, H.: Statistical machine translation of german compound words. In: FinTAL - 5th International Conference on Natural Language Processing, Springer Verlag, LNCS, Turku, Finland (August 2006) 616–624

[16] Giménez, J., Màrquez, L.: Asiya: An Open Toolkit for Automatic Machine Translation (Meta-)Evaluation. The Prague Bulletin of Mathematical Linguistics (94) (2010) 77–86

[17] Tillmann, C., Vogel, S., Ney, H., Zubiaga, A., Sawaf, H.: Accelerated DP based Search for Statistical Translation. In: Proceedings of European Conference on Speech Communication and Technology. (1997)

[18] Snover, M., Dorr, B., Schwartz, R., Micciulla, L., , Makhoul, J.: A Study of Translation Edit Rate with Targeted Human Annotation. In: Proceedings of AMTA. (2006) 223–231

[19] Nießen, S., Och, F.J., Leusch, G., Ney, H.: An Evaluation Tool for Machine Translation: Fast Evaluation for MT Research. In: Proceedings of the 2nd International Conference on Language Resources and Evaluation. (2000)

[20] Doddington, G.: Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In: Proceedings of the 2nd Internation Conference on Human Language Technology. (2002) 138–145

[21] Melamed, I.D., Green, R., Turian, J.P.: Precision and Recall of Machine Translation. In: Proceedings of the Joint Conference on Human Language Technology and the North American Chapter of the Association for Computational Linguistics (HLT-NAACL). (2003)

[22] Banerjee, S., Lavie, A.: METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. In: Proceedings of ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization. (2005)

[23] Giménez, J., Màrquez, L.: Linguistic features for automatic evaluation of heterogeneous mt systems. In: Proceedings of the Second ACL Workshop on Statistical Machine Translation. (June 2007) 256–264

[24] Angelov, K.: The Mechanics of the Grammatical Framework. PhD thesis, Chalmers University of Technology, Gothenburg, Sweden (2011)

[25] Tinsley, J., Way, A., Sheridan, P.: PLuTO: MT for Online Patent Translation. In: Proceedings of the 9th Conferences of the Association for Machine Translation in the Americas (AMTA 2010). (2010)