Grammar Engineering for a Customer: a Case Study with Five Languages

Aarne Ranta
University of Gothenburg
and Digital Grammars AB
aarne@chalmers.se

Christina Unger
Bielefeld University
cunger@cit-ec.uni-bielefeld.de

Daniel Vidal Hussey
University of Gothenburg
daniel.vidal.hussey@gmail.com

Abstract

This paper describes a grammar-based translation system built by a company for a paying customer. The system uses a multilingual grammar for English, Finnish, German, Spanish, and Swedish written in GF (Grammatical Framework). The grammar covers a corpus of technical texts in Swedish, describing properties of places and objects related to accessibility by disabled people. This task is more complex than most previous GF tasks, which have addressed controlled languages. The main goals of the paper are: (1) to find a grammar architecture and workflow for domain-specific grammars on real data (2) to estimate the quality reachable with a reasonable engineering effort (3) to assess the cost of grammar-based translation and its commercial viability.

1 Introduction

While statistical methods dominate in assimilation (browsing quality) translation, grammars have been argued to have a niche in dissemination (publication quality). The rationale is that such tasks are often domain-specific and need high precision rather than wide coverage. A recent effort in this direction was the European MOLTO project (Hallgren et al., 2012), which developed tools for such tasks building on the grammar formalism GF (Grammatical Framework, (Ranta, 2011)).

MOLTO also built showcases for a few domains (mathematics (Saludes and Xambó, 2011), paintings (Damova et al., 2014), business models (Davis et al., 2012), and touristic phrases (Ranta et al., 2012)). But these showcases were all dealing with CNL (controlled natural language), which was defined by the grammar writers and designed to be processable by formal grammars. The present paper takes a step beyond these research prototypes, as the language to be translated is not controlled, but naturally written by different authors at different times. The system was ordered by a paying customer to solve a real problem. Also the size of the language is larger than in the mentioned MOLTO applications.

The task was to create a translation system for a web service documenting the accessibility to different sites, e.g. whether they can be visited by wheelchair users[1]. The service provider had a set of text templates written in Swedish, for instance stating that *the width of the door is [X]*. These templates had previously been translated by professional translators to English and partly to other languages. Also Google translate had been used for some languages.

For quality reasons, Google translate was deemed unsatisfactory by the customer. Manual translation was problematic because of its high cost and low speed: the system is updated by new texts continuously, and their translations should appear without delays. Therefore the customer contracted a company[2] to build an automatic system that could deliver high-quality translations faster than before.

This paper addresses a part of the task: a grammar used for translating from Swedish to English, Finnish, German, and Spanish. The translation system parses Swedish sentences (i.e. templates) and generates translations in other languages, by using the interlingual grammar architecture of GF. The translations are manually revised and post-edited, partly because the customer wants to be sure about their quality, partly because the input is noisy and contains typos, grammar errors, and other problems not amenable to purely grammar-based automatic translation. A fully automatic system would require more control on the input

[1] Tillgänglighetsdatabasen, www.t-d.se.

[2] Digital Grammars AB, www.digitalgrammars.com.

Proceedings of the Grammar Engineering Across Frameworks (GEAF) Workshop, 53rd Annual Meeting of the ACL and 7th IJCNLP, pages 1–8,
Beijing, China, July 26-31, 2015.

language. This was left to future work.

The approach we chose was the "Embedded CNL" (Ranta, 2014): a Controlled Natural Language embedded in a general purpose syntactic grammar. The parser gives priority to CNL analyses whenever possible, but also provides coarser analyses as back-up. This makes the automatic translation robust. Since the system knows which grammar rules are used for each part of the translation, it can show confidence information to the user and the post-editor.

The main questions of this paper are:

- How to best build an embedded CNL system for a task like this?
- How good is the quality, in terms of the usual MT scores and post-editing required?
- Is this approach commercially viable, i.e. competitive with human of translation?

The paper has the following structure: Section 2 describes the text corpus to be translated. Section 3 is a very brief introduction to GF. Section 4 outlines the structure of the grammar and the grammar writing process. Section 5 outlines the translation and post-editing workflow. Section 6 gives evaluation on two dimensions: the time taken by grammar writing and post-editing, and the usual scores (BLEU) for translation quality. Section 7 discusses related work. Section 8 concludes, trying to answer the three questions and give recommendations for later work.

2 The corpus

The starting point was a set of texts in Swedish. Most of them had manual translations in English, many also in Finnish and German. The customer was happy with the English translations but wanted to replace the Finnish and German ones, as well as to create Spanish translations. The number of texts was around 1,900, mostly short sentences of under 10 words. But as the texts also had heavy HTML markup, which had to be rendered correctly in translation, the corpus with tags and repetitions of texts had 26,000 tokens.

The most interesting part of the markup was the **variable**, a segment into which some actual value is inserted when the text is used to describe some object. As the first step of translation, we erased all markup but the variable. As the last step, we inserted it back by using an alignment between the source and the translation.

Figure 1 shows a sample from the corpus. The variables are in brackets. The brackets contain identifiers marking what kind of value is to be inserted in them; the final grammar distinguishes between 13 different variables, which typically belong to different syntactic categories.

After the removal of markup, the number of unique texts was 1,185, word count 7,309. There were 1,258 unique words and 980 unique lemmas, as measured by the morphological analyser of SALDO (Borin et al., 2013). 906 of these were content words (nouns, adjectives, verbs), most of which had precise technical meanings.

In the grammar writing process, it turned out that many word combinations must be treated as multiword constructions, since their translations are not compositional. For instance, Swedish *motsvarande* is literally *corresponding*, but the proper translation of *NP eller motsvarande* is *NP or similar* in English, *NP o elemento parecido* in Spanish.

72 such constructions were included in the final lexicon. This number is relatively low because Swedish forms compounds without spaces, and these were easy to identify as tokens at the outset. For comparison, the final English lexicon has 274 multiwords. Starting with English would thus involve more work in identifying the multiwords.

3 GF and resource grammars

GF started at Xerox (XRCE) to support multilingual generation in controlled-language scenarios (Dymetman et al., 2000). A GF grammar consists of an **abstract syntax**, which captures the semantics of the application domain, and a set of **concrete syntaxes**, which map abstract syntax trees into strings of different languages. As an example from the domain of this paper, one could have an **abstract syntax function**

```
fun Length :
  Object -> Measure -> Fact
```

to model sentences such as *the length of the changing table is 120 cm*. The concrete syntax is given by a **linearization rule**,

```
lin Length o m =
  "the length of" ++ o ++ "is" ++ m
```

This rule is nothing but a string template, which together with the abstract syntax rule is a decomposition of the context-free grammar rule

```
Fact ::=
 "the length of" Object "is" Measure
```

```
avåkningsskyddet är placerat på [object]
the protective guards are placed on [object]

begränsad gångyta är [units] bred
the limited pedestrian area is [units] wide

karusell [exist] som är anpassad för rullstol
roundabout adapted for wheelchair is [exist]

språk vid visning eller guidning [is] speciellt anpassat för att vara enkelt och lättförståeligt
language when showing or guiding [is] especially adapted to be simple and easy to understand
```

Figure 1: Parallel English-Swedish sentences from the corpus.

to a "tectogrammarical" and "phenogrammatical" rule (Curry, 1961). The strength of GF is that different languages can have not only different linearizations but also different *types* of linearizations. Thus for instance `Object` in English is a string, but in German case-dependent string, which is rendered in the genitive in this construction. The German linearization rule is thus

```
"die Länge" ++ gen o ++ "ist" ++ m
```

which generates *die Länge des Wickeltisches ist 120 cm* for the object whose nominative form is *der Wickeltisch*. The Finnish and Spanish rules are

```
gen o ++ "pituus on" ++ m
"la longitud"++ gen o ++"es de"++ m
```

respectively, where Finnish has a different word order and Spanish adds the preposition *de*.

The first GF grammars were small, typically involving up to 200 abstract syntax rules; their context-free expansions could of course be thousands of rules, due to parametric variations such as case. But it soon turned out that writing such grammars from scratch for each application was untenable, as each application had to reimplement morphology and syntax. To relieve this task, the **GF Resource Grammar Library** (RGL) (Ranta, 2009) was created, inspired by the resource grammars of CLE (Core Language Engine) (Rayner et al., 2000). The current RGL includes 30 languages, implementing the inflectional morphology and a comprehensive part of the syntax of each language. The RGL has a common API (Application Programmer's Interface) based on an abstract syntax. Thus for instance

```
possCN : CN -> NP -> CN
```

is a function that forms the possessive "CN of NP" for any CN (common noun) and NP (noun phrase) in any of the 30 languages. The linearization rules of `Length` can now be written

```
mkCl (mkNP the_Det
  (possCN (mkCN length_N)) o) m
```

uniformly in these four languages, by just varying the definition of the constant `length_N`, which in turn can be written

```
mkN "length"
mkN "Länge"
mkN "pituus"
mkN "longitud"
```

using the **smart paradigms** (Détrez and Ranta, 2012) of each language, which infer the morpological properties of words from one or more forms. In Spanish, the argument `m` must be made into a prepositional phrase `(prepNP de_Prep m)`.

The RGL has increased the productivity of CNL implementations in GF, so that a system with a few hundred abstract syntax rules can be created in a few days and portable to any new language in a few hours (Ranta et al., 2012). The RGL has recently also scaled up to open-domain translation, due to improved parsing algorithms and statistical disambiguation (Angelov and Ljunglöf, 2014), chunk-based back-up of syntactic parsing (Ranta, 2014), and the ease of building large lexica with smart paradigms. As a demonstration, a **wide-coverage translator** (WCT) has been released as a mobile app (Angelov et al., 2014).

4 The grammar writing process

The grammar writing task was given the following constraints, for reasons of commercial viability:

1. The abstract syntax and Swedish should be built in 2 person weeks.
2. Each of the other languages should be built in 1 person week, including the postprocessing of the translations.
3. Later translations of similar text should be five times as fast as human translation.

4. Each language could be implemented by a different programmer, who does not need to know the other grammars.
5. The grammarians need basic GF skills and native knowledge of the target language, but no Swedish.

Constraints 1 and 2 meant that we had to stop the grammar development at some point and proceed to post-editing, even if the grammar was not yet perfect. Thus the workflow for each target language (after the abstract syntax and Swedish) was defined as follows:

- Days 1,2: Initial grammar, complete for the abstract syntax.
- Days 3,4: Translation + post-editing + grammar improvement loop.
- Day 5: Post-editing to deliver the final translations.

Constraint 3 means that the grammar must be good enough to support faster translation in later tasks. Constraint 4 means that the work can be done in parallel, constraint 5 that the grammar can easily be extended with new languages. In the actual task reported here, the three programmers were native speakers of Finnish, German, and Spanish, two of them GF experts and the third one a student with a basic course of GF covering most of the textbook (Ranta, 2011).

The source texts were in Swedish, but requiring knowledge of Swedish would make it too hard to find grammarians for new languages. Two of the grammarians actually had to work on the basis of the English translations. It turned out useful to write an English concrete syntax to help their work, so that they knew exactly what was intended with each abstract rule. The English grammar was not a part of the deal, but it was needed for this purpose as well as for future use. It was also a sanity check of the abstract syntax: building a grammar with only one language in mind could result in an abstract syntax that is not abstract enough, as pointed out in the "best practices" of the MOLTO project (Hallgren et al., 2012).

The process thus started with the abstract syntax together with concrete syntaxes for Swedish and English. We had two main options for the grammar writing process:

- **bottom-up**: build a dedicated CNL and implement it with the RGL (resource grammar library);
- **top-down**: start with the WCT (wide-coverage translator) and improve it by domain-specific rules (using RGL).

The bottom-up approach was excluded almost immediately, because we had to take the corpus as it was: we had to translate sentences written in different times by different human authors, which could not be expected to follow strict CNL rules.

Figure 2 shows the grammar writing phases. The rectangles are off-the-shelf components from GF open source repositories. The ovals are grammar parts created in the project. The dashed rectangle RGL means that the RGL is used as a library rather than as a part of the run-time translation grammar.

Phase 0: WCT

We started with a baseline using WCT out of the box, just extended with rules for the variables. The results were far from satisfactory. We did receive translations to all sentences, but they were too often relying on robustness (i.e. chunking rather than full analyses), lexical choices were bad, and many words were returned untranslated because they were missing. The same happens with Google translate, because there are many uncommon Swedish words. The grammar at this phase was just the wide-coverage translation grammar of GF.

Phase 1: WCT + lexicon

This phase added the missing words to the lexicon, but the syntax was still the WCT syntax. Many compounds were first translated compositionally from their parts: thus *lekplatsområde* became the incomprehensible *game place section* instead of the correct *playground area*. Such problems could have been partly solved by using the corpus as data for statistical phrase alignment and deriving a GF multiword lexicon from that (Enache et al., 2014). But the data was fully available only for English, and it did not have the desired terminological consistency. Thus we ended up creating a lexicon semiautomatically from the corpus, occasionally adding multiword constructions later when new languages required this. The grammar at this phase was the wide-coverage grammar plus the domain-specific lexicon.

Phase 2: CNL as extended subset of WCT

There were two problems with the grammar built using the top-down strategy:

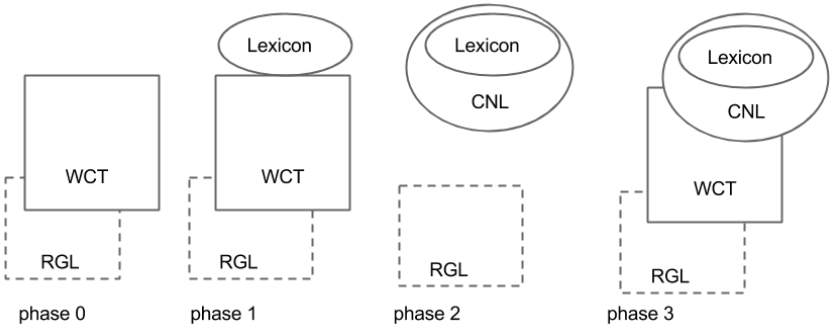

Figure 2: The four phases of grammar development.

- **Ambiguity**: including all syntax rules creates ambiguities that are not adequately solved by the generic statistical model of WCT.
- **Incompleteness**: even though all sentences are parsed, the parses don't capture idiomatic constructions typical of this domain.

The incompleteness concerned not only multiword constructions, but also general syntactic rules. Some authors of the corpus had used a telegraphic style, with missing articles and endings; Swedish expresses definiteness in morphology, for instance, *dörren* = *dörr* + *en* ("the door"). These phenomena required new rules in the general syntax, not just domain-specific CNL rules.

At the same time, only some of the RGL structures turned out to be actually needed. Less than a third of the 216 combination rules in the RGL abstract syntax were used in correct analyses of the corpus; the rest were just creating spurious parses

The module system of GF provides a way to reuse and recombine parts of grammars, even of grammars given as "black boxes" such as the RGL in its standard binary distribution. Using these techniques, we made two kinds of changes on top of the standard RGL:

- **Removing** rules until we got just those parses that made sense for the domain.
- **Adding** rules that enabled parsing input that was not parsed correctly before.

We ended up removing all but 30 rules of the standard RGL and adding about 40 new rules, having to do with the telegraphic style, measument units, and existential constructions. In addition, 16 rules were needed to embed the variables in standard syntactic constructions. They were not always just words, but syntactic functions.

For example, the existential variable `[exist]`, as shown in Figure 1, marks a semantic function that is realized in different ways in different languages. The abstract syntax is simply

```
fun VarExistNPS : NP -> S
```

The linearization has a variable that gets both positive and negative values, "there is/are (not)". This cannot be done by simple slot-filling, because the filler may depend on the surrounding sentence (e.g. "is/are" in English) and because, conversely, the surrounding sentence may depend on the slot filler (in Finnish, the NP is in the nominative if the existential is positive, partitive if negative). However, human translators had been ingenious and found ways to generate acceptable language even with slot filling. Thus the existential verb of Swedish was rendered as an adjective in English (*(not) available*) and in German (*(nicht) erhältlich*). Swedish (*finns (inte)*) and Spanish (*(no) hay*) could both be treated with verbs.

From the engineering point of view, the syntax part was extremely simple. The GF source code is around 180 lines for all languages, as opposed to the lexicon, which is around 1050 lines. The RGL and its common API come with the promise that writing grammars by using the RGL is equally complicated for all languages supported, which is corroborated by these figures; see the full code statistics in Table 2.

The resulting grammar was a CNL based on an abstract syntax that was syntactic rather than semantic. Thus for instance *the length of X is Y* is not parsed as the logical predication `(Length X Y)` but as an NP-VP predication

```
Pred (PossNP X (UseN längd_N))
  (MeasureVP Y)
```

This kind of abstract syntax is also used in ACE (Attempto Controlled English) (Fuchs et al., 2008). The syntactic structures and their logi-

cal semantics are known since (Montague, 1974), which makes ACE well suited for inference. But it is not so good for translation, because syntax may have to be changed when going from one language to another. Adding semantic constructions to the abstract syntax would therefore be the way to go if perfect quality was required.

The grammar at this point was a CNL with the lexicon generated in the previous phase, together with a minimal set of syntax rules. This grammar was able to translate 95% of the corpus.

Phase 3: Embedded CNL in WCT

The grammar of phase 2 left 54 sentences unparsed. These sentences were long, tricky, and often ungrammatical. Extending the grammar would have been hard work with diminishing returns. What is more, the time allocated for grammar development was coming to its end. The practical solution was to switch back to the WCT for the remaining sentences.

Since the same grammar was expected to be used later for new sentences, we wanted a solution with the best of both worlds: use CNL as much as possible and WCT only as a back-up. An embedded CNL does this by combining the two grammars under a common start category, say `S_top`. This category can be produced in two ways:

```
UseCNL : S_cnl -> S_top
UseWCT : S_wct -> S_top
```

The weights in the probabilistic parser are set so that `UseCNL` is given priority over `UseWCT`.

However, even unparsed sentences may have parts that belong to the CNL. For instance, they can contain noun phrases that are technical terms whose translation is domain-specific. To make maximal use of these parts, the robust grammar has **coercion rules** from 12 subsentential categories of the CNL to corresponding categories of the WCT, for instance,

```
CoerceNP : NP_cnl -> NP_wct
```

The weights are again set in a way that gives this rule priority over other rules producing `NP_wct`.

The additional grammar module for the robust grammar is just 70 lines, actually the same for all languages except for the import list telling which RGL modules are used. It can therefore be produced automatically for any new language.

The final version of the grammar was now able to parse all sentences in the corpus, 95% with the CNL and 5% with robustness.

5 The translation workflow

With the abstract syntax, Swedish, and English in place, the grammarians of Finnish, German, and Spanish started their work. The Swedish/English grammarian meanwhile produced a treebank covering the whole corpus, to guide the other grammarians. The grammarians used the trees in the treebank to generate their own translations. Figure 3 shows an example entry from the treebank.

The GF translation is often different from the original human translation, either because it is wrong or because it just expresses the meaning in a different way. In Figure 3, both reasons apply. The translation is wrong because it uses the indefinite article; this in turn because the telegraphic Swedish null determiner is rendered as an indefinite in the English grammar - which is correct in most cases, but not in this one (where the omission of determiner in the source is actually a questionable choice). The grammar moreover uses a different word for *gångyta*, namely *walking area* instead of *pedestrian area*. This is fine, because both translations occur in the reference.

The grammarians used the treebank to guide their grammar development. In the last phase, they moved on to just post-editing the translations. The final deliverable of the grammarians was the concrete syntax modules, the machine translations produced by the grammar, and the post-edited correct translations.

6 Evaluation

The first evaluation question is the quality of the translations. Table 1 shows BLEU scores for each language, computed by the Asiya tool (Gonzàlez et al., 2012) by using the post-edited translations as reference, as well as the percentage of translations that were correct without post-editing. The results cover all 1,185 texts of the corpus and are also shown separately for the CNL and robust translations. The Google translate scores are also with a reference obtained by post-editing the MT result minimally. These scores are for a sample of 40 sentences for CNL and 10 for robust translations and hence less representative. Notice that this is not a proper evaluation in the usual sense, because the grammar is tested on the same material that was used for building it; the purpose here is to measure the quality that can be produced by a limited effort, and also to check how it compares to Google translate, which had been previ-

```
Swe: begränsad gångyta är [units] bred
GF:  (Pred (NullDetCN (AdjCN (PastPartA begränsa_V2) (UseN gångyta_N))) (measureVP varMeasure bred_A))
Eng: a restricted walking area is [units] wide
Ref: the limited pedestrian area is [units] wide
```

Figure 3: Example of a GF treebank entry: Swe (source), GF (tree), Eng (GF translation), Ref (human translation).

language	correct	BLEU,GF	Google
Finnish, CNL	48%	**77**	31
Finnish, robust	0%	**31**	20
Finnish, all	46%	**73**	28
German, CNL	44%	**75**	37
German, robust	0%	33	**34**
German, all	42%	**73**	37
Spanish, CNL	34%	**76**	28
Spanish, robust	0%	**39**	25
Spanish, all	32%	**74**	28

Table 1: Quality evaluation.

language	GE	PE	PES	Ws/h	LoC
Finnish	14	6	1200	370	1292
German	18	6	1200	300	1290
Spanish	38	8	900	160	1291
Swedish	12	-	-	-	1303
English	12	-	-	-	1284
abstract	12	-	-	-	1286
prepare	12	-	-	-	400
total	118	20	1100	160	8056

Table 2: The grammar writing and translation effort. GE = grammar engineering (hours), PE = postediting (hours), PES Ws (post-editing speed words per hour), Ws/h (words translated per hour of work), LoC = lines of GF/Haskell code. The "total" for PES and Ws/h means the average of all languages.

ously used by the customer.

The second evaluation question is the productivity of grammar writing as a translation method. Table 2 shows the working hours and lines of GF code spent on different languages. The "prepare" score shows the hours spent on analysing and preparing the data and the amount of Haskell code written to help the task. The main result is that the time budget was held: 200 hours were allocated and 138 used. The high post-edit speed predicts that system development costs are amortized as more texts are translated. The time difference between the Spanish and other grammarians reflects the prior GF expertise of the grammarians and also the availability of some prior Finnish and German translations that helped with the technical terminology. Each new language amortizes the cost of the generic parts (48 hours), due to the interlingual architecture. The interlingual RGL explains why different languages need similar amounts of GF code.

7 Related work

Grammar-based translation is familiar from compilers, where synchronous grammars (Aho and Ullman, 1969) are a technique that has also been applied to human languages. With its explicit interlingua (abstract syntax), GF resembles (Rosetta, 1994), whereas most other approaches are based on transfer, e.g. (Rayner et al., 2000; Butt et al., 2002; Rayner et al., 2006).

The quality of the earlier GF projects using pure CNLs is higher, with typical BLEU scores between 80 and 95 (Rautio and Koponen, 2013). But it is interesting to note that even in those cases, the post-editors were not fully satisfied. Thus it is not clear if translation can be made completely automatic, if dissemination quality is the goal.

8 Conclusion

How to best build a system like this? The embedded CNL approach provided a good balance of quality and robustness. The CNL parsed 95% of the sentences, and covering the rest would have given diminishing returns. Including more semantic constructions (rather than syntactic combinations) in the CNL could have given better quality.

How good is the quality? The BLEU scores were at least 73 for all languages, which meant easy post-editing, at least 900 words per hour.

Is this approach commercially viable? The translation of the corpus to three languages was made at the rate of 160 words per hour, which includes both grammar writing and post-editing. This is comparable to human translation, so that we are just above the bar if only this corpus is considered. However, the high post-editing speed suggests that building grammars is profitable if more text of the same kind is translated later. Moreover, the interlingual architecture of GF enables new languages to be added at lower costs.

References

Alfred V. Aho and Jeffrey D. Ullman. 1969. Syntax directed translations and the pushdown assembler. *Journal of Computer and System Sciences*, 3(1):37 – 56.

Krasimir Angelov and Peter Ljunglöf. 2014. Fast statistical parsing with parallel multiple context-free grammars. In *EACL'14*, pages 368–376.

Krasimir Angelov, Björn Bringert, and Aarne Ranta. 2014. Speech-enabled hybrid multilingual translation for mobile devices. *EACL'14*, pages 41–44.

Lars Borin, Markus Forsberg, and Lennart Lönngren. 2013. Saldo: a touch of yin to wordnet's yang. *Language resources and evaluation*, 47(4):1191–1211.

Miriam Butt, Helge Dyvik, Tracy Holloway King, Hiroshi Masuichi, and Christian Rohrer. 2002. The parallel grammar project. In *Proceedings of the 2002 workshop on Grammar engineering and evaluation-Volume 15*, pages 1–7. Association for Computational Linguistics.

Haskell B. Curry. 1961. Some logical aspects of grammatical structure. In Roman Jakobson, editor, *Structure of Language and its Mathematical Aspects: Proceedings of the Twelfth Symposium in Applied Mathematics*, pages 56–68. American Mathematical Society.

Mariana Damova, Dana Dannélls, Ramona Enache, Maria Mateva, and Aarne Ranta. 2014. Multilingual natural language interaction with semantic web knowledge bases and linked open data. In *Towards the Multilingual Semantic Web*, pages 211–226. Springer Berlin Heidelberg.

Brian Davis, Ramona Enache, Jeroen van Grondelle, and Laurette Pretorius. 2012. Multilingual verbalisation of modular ontologies using gf and lemon. In Tobias Kuhn and Norbert Fuchs, editors, *Controlled Natural Language*, volume 7427 of *LNCS/LNAI*, pages 167–184. Springer.

Grégoire Détrez and Aarne Ranta. 2012. Smart paradigms and the predictability and complexity of inflectional morphology. In *EACL (European Association for Computational Linguistics)*, Avignon, April. Association for Computational Linguistics.

Marc Dymetman, Veronika Lux, and Aarne Ranta. 2000. XML and multilingual document authoring: Convergent trends. In *Proc. Computational Linguistics COLING, Saarbrücken, Germany*, pages 243–249. International Committee on Computational Linguistics.

Ramona Enache, Inari Listenmaa, and Prasanth Kolachina. 2014. Handling non-compositionality in multilingual CNLs. In *Controlled Natural Language - 4th International Workshop, CNL 2014, Galway, Ireland, August 20-22, 2014. Proceedings*, volume 8625 of *LNCS*.

Norbert E. Fuchs, Kaarel Kaljurand, and Tobias Kuhn. 2008. Attempto Controlled English for Knowledge Representation. In Cristina Baroglio, Piero A. Bonatti, Jan Małuszyński, Massimo Marchiori, Axel Polleres, and Sebastian Schaffert, editors, *Reasoning Web, Fourth International Summer School 2008*, number 5224 in LNCS, pages 104–124. Springer.

Meritxell Gonzàlez, Jesús Giménez, and Lluís Màrquez. 2012. A Graphical Interface for MT Evaluation and Error Analysis. In *The 50th Annual Meeting of the Association for Computational Linguistics*.

Thomas Hallgren, Aarne Ranta, John Camilleri, Grégoire Détrez, and Ramona Enache. 2012. Grammar Tools and Best Practices. MOLTO Deliverable D2.3, June.

Richard Montague. 1974. *Formal Philosophy*. Yale University Press, New Haven. Collected papers edited by Richmond Thomason.

Aarne Ranta, Ramona Enache, and Grégoire Détrez. 2012. Controlled Language for Everyday Use: the MOLTO Phrasebook. In Tobias Kuhn and Norbert Fuchs, editors, *Controlled Natural Language*, volume 7427 of *LNCS/LNAI*. Springer.

Aarne Ranta. 2009. The GF Resource Grammar Library. *Linguistics in Language Technology*, 2:1–65.

Aarne Ranta. 2011. *Grammatical Framework: Programming with Multilingual Grammars*. CSLI Publications, Stanford.

Aarne Ranta. 2014. Embedded controlled languages. In *Controlled Natural Language - 4th International Workshop, CNL 2014, Galway, Ireland, August 20-22, 2014. Proceedings*, volume 8625 of *LNCS*.

Jussi Rautio and Maarit Koponen. 2013. Deliverable 9.2: Molto evaluation and assessment report.

Manny Rayner, David Carter, Pierrette Bouillon, Vassilis Digalakis, and Mats Wirén. 2000. *The Spoken Language Translator*. Cambridge University Press, Cambridge.

Manny Rayner, Beth Ann Hockey, and Pierrette Bouillon. 2006. *Putting Linguistics into Speech Recognition: The Regulus Grammar Compiler*. CSLI Publications.

M. T. Rosetta. 1994. *Compositional Translation*. Kluwer, Dordrecht.

Jordi Saludes and Sebastian Xambó. 2011. The gf mathematics library. In Pedro Quaresma and Ralph-Johan Back, editors, *Proceedings First Workshop on CTP Components for Educational Software (THedu'11)*, number 79, pages 102–110. Electronic Proceedings in Theoretical Computer Science, 02/2012.