# Tools for Multilingual Grammar-Based Translation on the Web

**Aarne Ranta** and **Krasimir Angelov** and **Thomas Hallgren**
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
aarne@chalmers.se, krasimir@chalmers.se, hallgren@chalmers.se

## Abstract

This is a system demo for a set of tools for translating texts between multiple languages in real time with high quality. The translation works on restricted languages, and is based on semantic interlinguas. The underlying model is GF (Grammatical Framework), which is an open-source toolkit for multilingual grammar implementations. The demo will cover up to 20 parallel languages.

Two related sets of tools are presented: grammarian's tools helping to build translators for new domains and languages, and translator's tools helping to translate documents. The grammarian's tools are designed to make it easy to port the technique to new applications. The translator's tools are essential in the restricted language context, enabling the author to remain in the fragments recognized by the system.

The tools that are demonstrated will be applied and developed further in the European project MOLTO (Multilingual On-Line Translation) which has started in March 2010 and runs for three years.

## 1 Translation Needs for the Web

The best-known translation tools on the web are Google translate[1] and Systran[2]. They are targeted to **consumers** of web documents: users who want to find out what a given document is about. For this purpose, **browsing quality** is sufficient, since the user has intelligence and good will, and understands that she uses the translation at her own risk.

Since Google and Systran translations can be grammatically and semantically flawed, they don't reach **publication quality**, and cannot hence be used by the **producers** of web documents. For instance, the provider of an e-commerce site cannot take the risk that the product descriptions or selling conditions have errors that change the original intentions.

There are very few automatic translation systems actually in use for producers of information. As already noted by Bar-Hillel (1964), machine translation is one of those AI-complete tasks that involves a trade-off between coverage and precision, and the current mainstream systems opt for coverage. This is also what web users expect: they want to be able to throw just anything at the translation system and get something useful back. Precision-oriented approaches, the prime example of which is METEO (Chandioux 1977), have not been popular in recent years.

However, from the producer's point of view, large coverage is not essential: unlike the consumer's tools, their input is predictable, and can be restricted to very specific domains, and to content that the producers themselves are creating in the first place. But even in such tasks, two severe problems remain:

- The **development cost problem**: a large amount of work is needed for building translators for new domains and new languages.
- The **authoring problem**: since the method does not work for all input, the author of the source text of translation may need special training to write in a way that can be translated at all.

These two problems have probably been the main obstacles to making high-quality restricted language translation more wide-spread in tasks where it would otherwise be applicable. We address these problems by providing tools that help developers of translation systems on the one hand, and authors and translators—i.e. the users of the systems—on the other.

In the MOLTO project (Multilingual On-Line Translation)[3], we have the goal to improve both the development and use of restricted language translation by an order of magnitude, as compared with the state of the art. As for development costs, this means that a system for many languages and with adequate quality can be built in a matter of days rather than months. As for authoring, this means that content production does not require the use of manuals or involve trial and error, both of which can easily make the work ten times slower than normal writing.

In the proposed system demo, we will show how some of the building blocks for MOLTO can already now be used in web-based translators, although on a

[1] www.google.com/translate
[2] www.systransoft.com
[3] www.molto-project.eu



*Proceedings of the ACL 2010 System Demonstrations*, pages 66–71,
Uppsala, Sweden, 13 July 2010. 

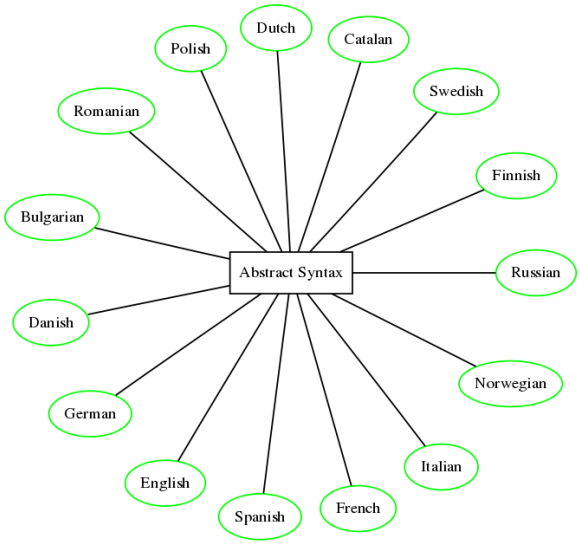


Figure 1: A multilingual GF grammar with reversible mappings from a common abstract syntax to the 15 languages currently available in the GF Resource Grammar Library.

smaller scale as regards languages and application domains. A running demo system is available at http://grammaticalframework.org:41296.

## 2 Multilingual Grammars

The translation tools are based on **GF**, **Grammatical Framework**[4] (Ranta 2004). GF is a **grammar formalism**—that is, a mathematical model of natural language, equipped with a formal notation for writing grammars and a computer program implementing parsing and generation which are declaratively defined by grammars. Thus GF is comparable with formalism such as HPSG (Pollard and Sag 1994), LFG (Bresnan 1982) or TAG (Joshi 1985). The novel feature of GF is the notion of **multilingual grammars**, which describe several languages simultaneously by using a common representation called **abstract syntax**; see Figure 1.

In a multilingual GF grammar, meaning-preserving translation is provided as a composition of parsing and generation via the abstract syntax, which works as an **interlingua**. This model of translation is different from approaches based on other comparable grammar formalisms, such as synchronous TAGs (Shieber and Schabes 1990), Pargram (Butt & al. 2002, based on LFG), LINGO Matrix (Bender and Flickinger 2005, based on HPSG), and CLE (Core Language Engine, Alshawi 1992). These approaches use **transfer rules** between individual languages, separate for each pair of languages.

Being interlingua-based, GF translation scales up linearly to new languages without the quadratic blow-up of transfer-based systems. In transfer-based systems, as many as $n(n-1)$ components (transfer functions) are needed to cover all language pairs in both directions. In an interlingua-based system, $2n+1$ components are enough: the interlingua itself, plus translations in both directions between each language and the interlingua. However, in GF, $n+1$ components are sufficient, because the mappings from the abstract syntax to each language (the **concrete syntaxes**) are **reversible**, i.e. usable for both generation and parsing.

Multilingual GF grammars can be seen as an implementation of Curry's distinction between **tectogrammatical** and **phenogrammatical** structure (Curry 1961). In GF, the tectogrammatical structure is called abstract syntax, following standard computer science terminology. It is defined by using a **logical framework** (Harper & al. 1993), whose mathematical basis is in the **type theory** of Martin-Löf (1984). Two things can be noted about this architecture, both showing improvements over state-of-the-art grammar-based translation methods.

First, the translation interlingua (the abstract syntax) is a powerful logical formalism, able to express semantical structures such as context-dependencies and anaphora (Ranta 1994). In particular, dependent types make it more expressive than the type theory used in Montague grammar (Montague 1974) and employed in the Rosetta translation project (Rosetta 1998).

Second, GF uses a **framework for interlinguas**, rather than one universal interlingua. This makes the interlingual approach more light-weight and feasible than in systems assuming one universal interlingua, such as Rosetta and UNL, Universal Networking Language[5]. It also gives more precision to special-purpose translation: the interlingua of a GF translation system (i.e. the abstract syntax of a multilingual grammar) can encode precisely those structures and distinctions that are relevant for the task at hand. Thus an interlingua for mathematical proofs (Hallgren and Ranta 2000) is different from one for commands for operating an MP3 player (Perera and Ranta 2007). The expressive power of the logical framework is sufficient for both kinds of tasks.

One important source of inspiration for GF was the WYSIWYM system (Power and Scott 1998), which used domain-specific interlinguas and produced excellent quality in multilingual generation. But the generation components were hard-coded in the program, instead of being defined declaratively as in GF, and they were not usable in the direction of parsing.

## 3 Grammars and Ontologies

Parallel to the first development efforts of GF in the late 1990's, another framework idea was emerging in web technology: XML, Extensible Mark-up Language, which unlike HTML is not a single mark-up language but a framework for creating custom mark-up lan-

[4] www.grammaticalframework.org

[5] www.undl.org

guages. The analogy between GF and XML was seen from the beginning, and GF was designed as a formalism for multilingual rendering of semantic content (Dymetman and al. 2000). XML originated as a format for structuring documents and structured data serialization, but a couple of its descendants, RDF(S) and OWL, developed its potential to formally express the semantics of data and content, serving as the fundaments of the emerging Semantic Web.

Practically any meaning representation format can be converted into GF's abstract syntax, which can then be mapped to different target languages. In particular the OWL language can be seen as a syntactic sugar for a subset of Martin-Löf's type theory so it is trivial to embed it in GF's abstract syntax.

The translation problem defined in terms of an ontology is radically different from the problem of translating plain text from one language to another. Many of the projects in which GF has been used involve precisely this: a meaning representation formalized as GF abstract syntax. Some projects build on previously existing meaning representation and address mathematical proofs (Hallgren and Ranta 2000), software specifications (Beckert & al. 2007), and mathematical exercises (the European project WebALT[6]). Other projects start with semantic modelling work to build meaning representations from scratch, most notably ones for dialogue systems (Perera and Ranta 2007) in the European project TALK[7]. Yet another project, and one closest to web translation, is the multilingual Wiki system presented in (Meza Moreno and Bringert 2008). In this system, users can add and modify reviews of restaurants in three languages (English, Spanish, and Swedish). Any change made in any of the languages gets automatically translated to the other languages.

To take an example, the OWL-to-GF mapping translates OWL's classes to GF's categories and OWL's properties to GF's functions that return propositions. As a running example in this and the next section, we will use the class of integers and the two-place property of being divisible ("$x$ is divisible by $y$"). The correspondences are as follows:

```
Class(pp:integer ...)
  ⇕
cat integer
ObjectProperty(pp:div
  domain(pp:integer)
  range(pp:integer))
  ⇕
fun div :
  integer ->  integer ->  prop
```

## 4 Grammar Engineer's Tools

In the GF setting, building a multilingual translation system is equivalent to building a multilingual GF grammar, which in turn consists of two kinds of components:

- a language-independent abstract syntax, giving the semantic model via which translation is performed;
- for each language, a concrete syntax mapping abstract syntax trees to strings in that language.

While abstract syntax construction is an extra task compared to many other kinds of translation methods, it is technically relatively simple, and its cost is moreover amortized as the system is extended to new languages. Concrete syntax construction can be much more demanding in terms of programming skills and linguistic knowledge, due to the complexity of natural languages. This task is where GF claims perhaps the highest advantage over other approaches to special-purpose grammars. The two main assets are:

- Programming language support: GF is a modern functional programming language, with a powerful type system and module system supporting modular and collaborative programming and reuse of code.
- **RGL**, the **GF Resource Grammar Library**, implementing the basic linguistic details of languages: **inflectional morphology** and **syntactic combination functions**.

The RGL covers fifteen languages at the moment, shown in Figure 1; see also Khegai 2006, El Dada and Ranta 2007, Angelov 2008, Ranta 2009a,b, and Enache *et al.* 2010. To give an example of what the library provides, let us first consider the inflectional morphology. It is presented as a set of lexicon-building functions such as, in English,

```
mkV : Str -> V
```

i.e. function `mkV`, which takes a string (`Str`) as its argument and returns a verb (`V`) as its value. The verb is, internally, an inflection table containing all forms of a verb. The function `mkV` derives all these forms from its argument string, which is the infinitive form. It predicts all regular variations: `(mkV "walk")` yields the purely agglutinative forms *walk-walks-walked-walked-walking* whereas `(mkV "cry")` gives *cry-cries-cried-cried-crying*, and so on. For irregular English verbs, RGL gives a three-argument function taking forms such as *sing,sang,sung*, but it also has a fairly complete lexicon of irregular verbs, so that the normal application programmer who builds a lexicon only needs the regular `mkV` function.

Extending a lexicon with domain-specific vocabulary is typically the main part of the work of a concrete syntax author. Considerable work has been put into RGL's inflection functions to make them as "intelligent" as possible and thereby ease the work of the

[6] EDC-22253, `webalt.math.helsinki.fi`
[7] IST-507802, 2004–2006, `www.talk-project.org`

users of the library, who don't know the linguistic details of morphology. For instance, even Finnish, whose verbs have hundreds of forms and are conjugated in accordance with around 50 conjugations, has a one-argument function `mkV` that yields the correct inflection table for 90% of Finnish verbs.

As an example of a syntactic combination function of RGL, consider a function for predication with two-place adjectives. This function takes three arguments: a two-place adjective, a subject noun phrase, and a complement noun phrase. It returns a sentence as value:

```
pred : A2 -> NP -> NP -> S
```

This function is available in all languages of RGL, even though the details of sentence formation are vastly different in them. Thus, to give the concrete syntax of the abstract (semantical) predicate `div x y` ("*x* is divisible by *y*"), the English grammarian can write

```
div x y = pred
  (mkA2 "divisible" "by") x y
```

The German grammarian can write

```
div x y = pred
  (mkA2 "teilbar" durch_Prep) x y
```

which, even though superficially using different forms from English, generates a much more complex structure: the complement preposition `durch_Prep` takes care of rendering the argument *y* in the accusative case, and the sentence produced has three forms, as needed in grammatically different positions (*x ist teilbar durch y* in main clauses, *ist x teilbar durch y* after adverbs, and *x durch y teilbar ist* in subordinate clauses).

The syntactic combinations of the RGL have their own abstract syntax, but this abstract syntax is *not* the interlingua of translation: it is only used as a library for implementing the semantic interlingua, which is based on an ontology and abstracts away from syntactic structure. Thus the translation equivalents in a multilingual grammar need not use the same syntactic combinations in different languages. Assume, for the sake of argument, that *x is divisible by y* is expressed in Swedish by the transitive verb construction *y delar x* (literally, "*y* divides *x*"). This can be expressed easily by using the transitive verb predication function of the RGL and switching the subject and object,

```
div x y = pred (mkV2 "dela") y x
```

Thus, even though GF translation is interlingua-based, there is a component of transfer between English and Swedish. But this transfer is performed at compile time. In general, the use of the large-coverage RGL as a library for restricted grammars is called **grammar specialization**. The way GF performs grammar specialization is based on techniques for optimizing functional programming languages, in particular **partial evaluation** (Ranta 2004, 2007). GF also gives a possibility to run-time transfer via semantic actions on abstract syntax trees, but this option has rarely been needed in previous applications, which helps to keep translation systems simple and efficient.

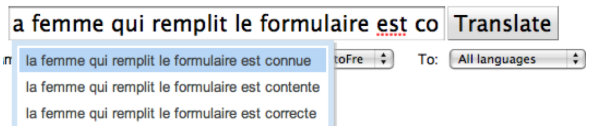


Figure 2: French word prediction in GF parser, suggesting feminine adjectives that agree with the subject *la femme*.

As shown in Figure 1, the RGL is currently available for 15 languages, of which 12 are official languages of the European Union. A similar number of new languages are under construction in this collaborative open-source project. Implementing a new language is an effort of 3–6 person months.

## 5 Translator's Tools

For the translator's tools, there are three different use cases:

- restricted source
  - production of source in the first place
  - modifying source produced earlier
- unrestricted source

Working with restricted source language recognizable by a GF grammar is straightforward for the translating tool to cope with, except when there is ambiguity in the text. The real challenge is to help the author to keep inside the restricted language. This help is provided by **predictive parsing**, a technique recently developed for GF (Angelov 2009)[8]. Incremental parsing yields **word predictions**, which guide the author in a way similar to the T9 method[9] in mobile phones. The difference from T9 is that GF's word prediction is sensitive to the grammatical context. Thus it does not suggest all existing words, but only those words that are grammatically correct in the context. Figure 2 shows an example of the parser at work. The author has started a sentence as *la femme qui remplit le formulaire est co* ("the woman who fills the form is *co*"), and a menu shows a list of words beginning with *co* that are given in the French grammar and possible in the context at hand; all these words are adjectives in the feminine form. Notice that the very example shown in Figure 2 is one that is difficult for n-gram-based statistical translators: the adjective is so far from the subject with which it agrees that it cannot easily be related to it.

Predictive parsing is a good way to help users produce translatable content in the first place. When modifying the content later, e.g. in a wiki, it may not be optimal, in particular if the text is long. The text can

[8] Parsing in GF is polynomial with an arbitrary exponent in the worst case, but, as shown in Angelov 2009, linear in practice with realistic grammars.

[9] `www.t9.com`

`Pred known_A (Rel` **woman_N** `(Compl fill_V2 form_N))`
the woman who fills the form is known
la femme qui remplit le formulaire est connue
⟶
`Pred known_A (Rel` **man_N** `(Compl fill_V2 form_N))`
the **man** who fills the form is known
*l'* **homme** qui remplit le formulaire est *connu*

Figure 3: Change in one word (boldface) propagated to other words depending on it (italics).

contain parts that depend on each other but are located far apart. For instance, if the word *femme* ("woman") in the previous example is changed to *homme*, the preceding article *la* has to be changed to *l'*, and the adjective has to be changed to the masculine form: thus *connue* ("known") would become *connu*, and so on. Such changes are notoriously difficult even for human authors and translators, and can easily leave a document in an inconsistent state. This is where another utility of the abstract syntax comes in: in the abstract syntax tree, all that is changed is the noun, and the regenerated concrete syntax string automatically obeys all the agreement rules. The process is shown in Figure 3. The one-word change generating the new set of documents can be performed by editing any of the three representations: the tree, the English version, or the French version. This functionality is implemented in the GF **syntax editor** (Khegai & al. 2003).

Restricted languages in the sense of GF are close to **controlled languages**, such as Attempto (Fuchs & al. 2008); the examples shown in this section are actually taken from a GF implementation that generalizes Attempto Controlled English to five languages (Angelov and Ranta 2009). However, unlike typical controlled languages, GF does not require the absence of ambiguity. In fact, when a controlled language is generalized to new languages, lexical ambiguities in particular are hard to avoid.

The predictive parser of GF does not try to resolve ambiguities, but simply returns all alternatives in the parse chart. If the target language has exactly the same ambiguity, it remains hidden in the translation. But if the ambiguity does make a difference in translation, it has to be resolved, and the system has to provide a possibility of manual disambiguation by the user to guarantee high quality.

The translation tool snapshot in Figure 2 is from an actual web-based prototype. It shows a slot in an HTML page, built by using JavaScript via the Google Web Toolkit (Bringert & al. 2009). The translation is performed using GF in a server, which is called via HTTP. Also client-side translators, with similar user interfaces, can be built by converting the whole GF grammar to JavaScript (Meza Moreno and Bringert 2008).

## 6 The Demo

In the demo, we will show

- how a simple translation system is built and compiled by using the GF grammar compiler and the resource grammar library
- how the translator is integrated in a web page
- how the translator is used in a web browser by means of an integrated incremental parser

A preliminary demo can be seen in http://grammaticalframework.org:41296. All the demonstrated tools are available as open-source software from http://grammaticalframework.org.

The work reported here is supported by MOLTO (Multilingual On-Line Translation. FP7-ICT-247914).

## References

Alshawi, H. (1992). *The Core Language Engine*. Cambridge, Ma: MIT Press.

Angelov, K. (2008). Type-Theoretical Bulgarian Grammar. In B. Nordström and A. Ranta (Eds.), *Advances in Natural Language Processing (GoTAL 2008)*, Volume 5221 of *LNCS/LNAI*, pp. 52–64. URL http://www.springerlink.com/content/978-3-540-85286-5/.

Angelov, K. (2009). Incremental Parsing with Parallel Multiple Context-Free Grammars. In *Proceedings of EACL'09, Athens*.

Angelov, K. and A. Ranta (2009). Implementing Controlled Languages in GF. In *Proceedings of CNL-2009, Marettimo*, LNCS. to appear.

Bar-Hillel, Y. (1964). *Language and Information*. Reading, MA: Addison-Wesley.

Beckert, B., R. Hähnle, and P. H. Schmitt (Eds.) (2007). *Verification of Object-Oriented Software: The KeY Approach*. LNCS 4334. Springer-Verlag.

Bender, E. M. and D. Flickinger (2005). Rapid prototyping of scalable grammars: Towards modularity in extensions to a language-independent core. In *Proceedings of the 2nd International Joint Conference on Natural Language Processing IJCNLP-05 (Posters/Demos)*, Jeju Island, Korea. URL http://faculty.washington.edu/ebender/papers/modules05.pdf.

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

Bringert, B., K. Angelov, and A. Ranta (2009). Grammatical Framework Web Service. In *Proceedings of EACL'09, Athens*.

Butt, M., H. Dyvik, T. H. King, H. Masuichi, and C. Rohrer (2002). The Parallel Grammar Project. In *COLING 2002, Workshop on Grammar Engineering and Evaluation*, pp. 1–7. URL http://www2.parc.com/isl/groups/nltt/pargram/buttetal-coling02.pdf.

Chandioux, J. (1976). MÉTÉO: un système opérationnel pour la traduction automatique des bulletins météreologiques destinés au grand public. *META 21*, 127–133.

Curry, H. B. (1961). Some logical aspects of grammatical structure. In R. Jakobson (Ed.), *Structure of Language and its Mathematical Aspects: Proceedings of the Twelfth Symposium in Applied Mathematics*, pp. 56–68. American Mathematical Society.

Dada, A. E. and A. Ranta (2007). Implementing an Open Source Arabic Resource Grammar in GF. In M. Mughazy (Ed.), *Perspectives on Arabic Linguistics XX*, pp. 209–232. John Benjamin's.

Dean, M. and G. Schreiber (2004). OWL Web Ontology Language Reference. URL http://www.w3.org/TR/owl-ref/.

Dymetman, M., V. Lux, and A. Ranta (2000). XML and multilingual document authoring: Convergent trends. In *COLING, Saarbrücken, Germany*, pp. 243–249. URL http://www.cs.chalmers.se/~aarne/articles/coling2000.ps.gz.

Enache, R., A. Ranta, and K. Angelov (2010). An Open-Source Computational Grammar for Romanian. In A. Gelbukh (Ed.), *Intelligent Text Processing and Computational Linguistics (CICLing-2010), Iasi, Romania, March 2010*, *LNCS*, to appear.

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

Hallgren, T. and A. Ranta (2000). An extensible proof text editor. In M. Parigot and A. Voronkov (Eds.), *LPAR-2000*, Volume 1955 of *LNCS/LNAI*, pp. 70–84. Springer. URL http://www.cs.chalmers.se/~aarne/articles/lpar2000.ps.gz.

Harper, R., F. Honsell, and G. Plotkin (1993). A Framework for Defining Logics. *JACM 40*(1), 143–184.

Joshi, A. (1985). Tree-adjoining grammars: How much context-sensitivity is required to provide reasonable structural descriptions. In D. Dowty, L. Karttunen, and A. Zwicky (Eds.), *Natural Language Parsing*, pp. 206–250. Cambridge University Press.

Khegai, J. (2006). GF Parallel Resource Grammars and Russian. In *Coling/ACL 2006*, pp. 475–482.

Khegai, J., B. Nordström, and A. Ranta (2003). Multilingual Syntax Editing in GF. In A. Gelbukh (Ed.), *Intelligent Text Processing and Computational Linguistics (CICLing-2003), Mexico City, February 2003*, Volume 2588 of *LNCS*, pp. 453–464. Springer-Verlag. URL http://www.cs.chalmers.se/~aarne/articles/mexico.ps.gz.

Martin-Löf, P. (1984). *Intuitionistic Type Theory*. Napoli: Bibliopolis.

Meza Moreno, M. S. and B. Bringert (2008). Interactive Multilingual Web Applications with Grammarical Framework. In B. Nordström and A. Ranta (Eds.), *Advances in Natural Language Processing (GoTAL 2008)*, Volume 5221 of *LNCS/LNAI*, pp. 336–347. URL http://www.springerlink.com/content/978-3-540-85286-5/.

Montague, R. (1974). *Formal Philosophy*. New Haven: Yale University Press. Collected papers edited by Richmond Thomason.

Perera, N. and A. Ranta (2007). Dialogue System Localization with the GF Resource Grammar Library. In *SPEECHGRAM 2007: ACL Workshop on Grammar-Based Approaches to Spoken Language Processing, June 29, 2007, Prague*. URL http://www.cs.chalmers.se/~aarne/articles/perera-ranta.pdf.

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

Power, R. and D. Scott (1998). Multilingual authoring using feedback texts. In *COLING-ACL*.

Ranta, A. (1994). *Type Theoretical Grammar*. Oxford University Press.

Ranta, A. (2004). Grammatical Framework: A Type-Theoretical Grammar Formalism. *The Journal of Functional Programming 14(2)*, 145–189. URL http://www.cs.chalmers.se/~aarne/articles/gf-jfp.ps.gz.

Ranta, A. (2009a). Grammars as Software Libraries. In Y. Bertot, G. Huet, J.-J. Lévy, and G. Plotkin (Eds.), *From Semantics to Computer Science*. Cambridge University Press. URL http://www.cs.chalmers.se/~aarne/articles/libraries-kahn.pdf.

Ranta, A. (2009b). The GF Resource Grammar Library. In *Linguistic Issues in Language Technology*, Vol. 2. URL http://elanguage.net/journals/index.php/lilt/article/viewFile/214/158.

Rosetta, M. T. (1994). *Compositional Translation*. Dordrecht: Kluwer.

Shieber, S. M. and Y. Schabes (1990). Synchronous tree-adjoining grammars. In *COLING*, pp. 253–258.