GF - Grammatical Framework

The core translation tools are based on the GF system, a grammar formalism that is, a mathematical model of natural language. It is equipped with a formal notation for writing grammars and with computer programs implementing parsing and generation that are declaratively defined by grammars. The novel feature of GF is the notion of *multilingual grammars*, used to describe several languages simultaneously by a common representation called *abstract syntax*. The abstract syntax enables meaning-preserving translation as a composition of parsing and generation. Thus, GF translation scales up linearly to new languages without the quadratic blowup of transfer-based systems: $n + 1$ components are sufficient to cover n languages, because the mappings from the abstract syntax to each language are usable for both generation and parsing.

Multilingual GF grammars can be seen as an implementation of Curry's distinction between *tectogrammatical* (abstract syntax), defined by using a logical framework, whose mathematical basis is in the type theory of Martin-Löf, and *phenogrammatical* structure.

GF improves over state-of-the-art grammar-based translation methods:

- the translation interlingua is a powerful logical formalism, able to express the finest semantical structures such as contextdependencies and anaphora. In particular, it is more expressive than the simple type theory used in Montague grammar and employed in the Rosetta translation project.
- GF uses a framework for interlinguas, rather than one universal interlingua, making it more light-weight and feasible than systems based on one universal interlingua, such as Rosetta and UNL, Universal Networking Language. It also gives more precision to special-purpose translation: the interlingua of a GF translation system can encode precisely structures and distinctions relevant to the task. Thus an interlingua for mathematical proofs is different from one for commands for operating an MP3 player.

One important source of inspiration for GF was the WYSIWYM system, which used domain-specific interlinguas and produced excellent quality in multilingual generation. However 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.

GF is open source and available, for major platforms, from www.grammaticalframework.org, licensed under GPL (the program) and LGPL (the libraries).

MOLTO's goal is to develop tools for web content providers to translate texts between multiple languages in real time with high quality. Languages are separate modules in the tool and can be varied; prototypes covering a majority of the EU's 23 official languages will be built.

FP7- 247914 molto-project.eu

MOLTO

non multa, sed multum

Tools for Multilingual Grammar-Based Translation on the Web

Grammar Engineer's Tools

Building a multilingual translation system amounts to building a multilingual GF grammar, namely:

- a language-independent abstract syntax;
- for each language, a concrete syntax mapping abstract syntax trees to strings in that language.

Abstract syntax construction is an extra task compared to other translation methods, but it is technically relatively simple, and gets 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 and GF eases it by two main assets:

- Programming language support: GF is a modern functional programming language, with a powerful type system and module system allowing 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.

To give an example, let us 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.

Typically, most of the effort in writing a concrete syntax goes into extending a lexicon with domain-specific vocabulary. RGL's inflection functions are designed as "intelligent" as possible and thereby ease the work of authors unaware 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
```

It is available in all languages of RGL, even though the details of sentence formation differ vastly. Thus, the concrete syntaxes of the abstract (semantical) predicate:

`div x y` ("x is divisible by y"),

are, for English and German:

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

The German 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)

`x durch y teilbar ist` (in subordinate clauses).

The translation equivalents in a multilingual grammar need not use the same syntactic combinations in different languages. For instance, the transitive verb construction `y delar x` (literally, "y divides x"), in Swedish, can be expressed using the transitive verb predication function of the RGL and switching the subject and object:

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

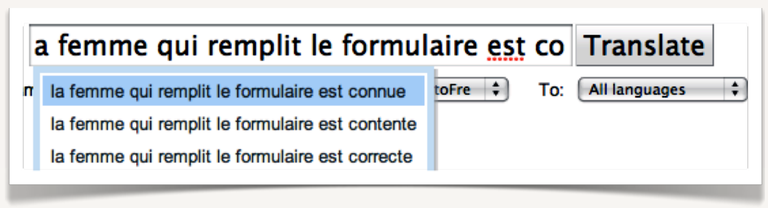
Thus, even though GF translation is interlingua-based, there is a component of transfer between English and Swedish. But this transfer is performed at compile time. In general, the use of the large-coverage RGL as a library for restricted grammars is called grammar specialization. The way GF performs grammar specialization is based on techniques for optimizing functional programming languages, in particular partial evaluation.

Translator's Tools

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

- restricted source
  - production of new source
  - modification/editing of source
- unrestricted source

The translating tool can easily handle a restricted source language recognizable by a GF grammar, except when there is ambiguity in the text. Authoring within the restricted language is helped by predictive parsing, a technique recently developed for GF. Incremental parsing yields grammatically correct word predictions, sensitive to the context, which guide the author in a way similar to the T9 method in mobile phones.



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 how this 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 it may not be optimal, in particular if the text is long. The text can contain parts that depend on each other but are located far apart. For instance, if the word `femme` ("woman") is changed to `homme`, the preceding article `la` has to be changed to `l'`, and the adjective `connue` ("known") would become `connu`, and so on. Such changes are notoriously difficult and can easily leave a document in an inconsistent state. In the GF syntax editor, changes can be performed on the abstract syntax tree and are propagated to the concrete syntax strings, that automatically obey all the agreement rules.

```
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
```

GÖTEBORGS UNIVERSITET

HELSINGIN YLIOPISTO

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

ontotext Semantic Technology Lab

MATRIXWARE

SEVENTH FRAMEWORK PROGRAMME