Proceedings of 1st International Workshop in Computational Linguistics for Uralic Languages (IWCLUL 2015); ‹http://dx.doi.org/10.7557/scs.2015.2›

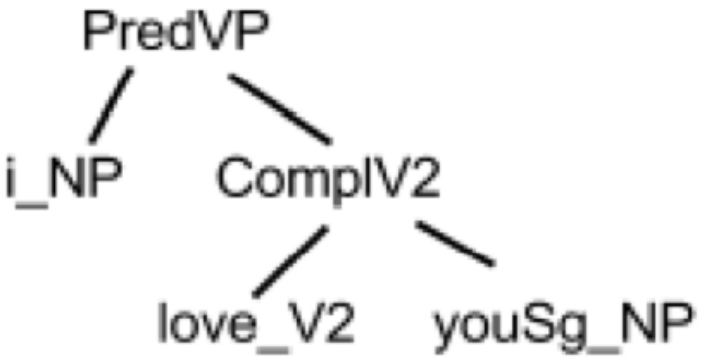
Grammatical Framework Tutorial, with a Focus on Fenno-Ugric Languages

Aarne Ranta and Inari Listenmaa

Göteborgs universitet och Chalmers tekniska högskola

1 GF in a nutshell

Grammatical Framework, GF, is a grammar formalism designed to support **multilingual grammars**. A multilingual grammar has an **abstract syntax**, which is shared by a set of languages. Each of these languages has a **concrete syntax**, which defines a relation between **trees** in the abstract syntax and **string**s in the language. For example, here is an abstract syntax tree representing predication with the verb *love*, subject *I*, and object *you*:



With appropriate concrete syntaxes for Chinese, English, Finnish, and French, we obtain the following strings, together with word alignments determined by the tree structure:

The process of generating the strings from the tree is called **linearization**, and the reverse process is **parsing**. All GF grammars are reversible, in the sense that they can be used for both linearization and parsing.

The main application of GF is **translation**, where the abstract syntax is used as an **interlingua**. But GF can also be used for **language comparison**, because the abstract syntax gives a formally precise way to express common structures. The predication structure (PredVP in the above tree) is an example: it is a common structure for the languages shown, even though the concrete syntaxes are quite different.

Technically, GF is based on **constructive type theory**, which is used for the abstract syntax, and **parallel multiple context-free grammars** (PMCFG), which are used for the concrete syntax. PMCFG is more expressive than context-free grammars, but it still enjoys polynomial parsing; in practice, the parsing speed is usually close to linear. The advantage of PMCFG is that it supports things like morphological variation and discontinuous constituents. This is what makes it possible to share abstract syntax for seemingly very different languages.

Writing PMCFG grammars manually would be very time-consuming and error-prone. An important feature of GF is therefore its support for **functional programming**, as a concise way of writing concrete syntaxes. Functional programming makes it easy to share common parts of code and thereby to avoid repetitive coding almost entirely. GF source code can hence be much more concise than e.g. context-free grammars or regular expressions, and GF has in fact been used as a method to generate code in these formats, independently of its

multilingual uses.

Together with the abstract syntax, the abstractions provided by functional programming methods make it possible to express linguistic generalizations. Such generalizations can be interesting for both linguistic theory (e.g. language typology) and practical engineering (reuse of code within and across languages). In particular, grammars for closely related languages can share major parts of code even in concrete syntax.

With the first release in 1998, GF has been applied to over 30 languages and has over 100 active developers around the world. Its main focus has been on **controlled languages** and **precision-oriented translation**. But the comprehensive **resource grammar library** (**RGL**) also makes it possible to build large-scale translation systems. An experimental system is currently available for 11 languages, covering all 110 language pairs.

GF is open-source software, released under licenses such as GLP (the grammar compiler) and LGPL and BSD (the run-time and RGL). This makes it possible to use GF for any purpose, including proprietary commercial applications. At least 6 companies have used GF in their projects.

2 The tutorial

The goal of the tutorial is to enable the participants to

- explore and reuse the resources currently available in GF
- contribute to the RGL, especially to its lexical resources
- get started with their own grammars

The ultimate goal is the development of RGL implementations for new Uralic languages, but some more training will probably be needed for this than this short tutorial. This can of course be done by achieved by using free material from the GF web page.

The contents of the tutorial can be divided to four lessons, which optimally need three hours in total.

2.1 Hands-on introduction

We will first show a demo of web-based translation in GF. After that, we will build a simple multilingual grammar together, by using the cloud-based grammar editor. Simple examples are enough to show that languages differ in non-trivial ways but can still share an abstract syntax.

The cloud-based grammar editor is a pedagogical tool enabling the use of GF without installing any software. Grammars created in it are also readily usable for translation and other functionalities in the GF cloud.

2.2 Tool and resource overview

We make a tour of the GF web pages to show what is available, how to install the tools, and how to reuse GF grammars on other platforms.

2.3 Morphology

We introduce the technique of **smart paradigms** and show how it is used to define inflection and lexica, with Finnish and Estonian inflection as examples.

2.4 Syntax

We look at some peculiarities of Finnish and Estonian agreement, word order, and nominalized verb phrase constructions as examples. For instance, we see how to deal with the rich system of participles and infinitives of Finnish in translation.

References

- GF home page: http://www.grammaticalframework.org/
- GF book: A. Ranta, *Grammatical Framework. Programming with Multilingual Grammars*, CSLI, Stanford, 2011.
  Chinese translation: 语法框架 为多种自然语言语法编程, Shanghai Jiao Tong University Press, 2014.
- GF Summer School: http://school.grammaticalframework.org/, 19–31 July 2015 in Malta