Online Parsing, Type Checking and Advanced Editor for Controlled Languages in GF

Krasimir Angelov

Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

September 8, 2010

The Problem

All UIs based on **word prediction** are helpful only in strictly left-to-right manner. In order to change something in the middle the user should remove everything until the point where the change should be made.

The old **syntax editor** does not have this problem but the users have to manipulate the abstract syntax directly.

Why do not merge?

The two UIs does not merge easily because:

1. **When** we should use the predictive editing and **when** the syntax editor?
2. The parser produces **forest** of abstract syntax trees instead of only one.

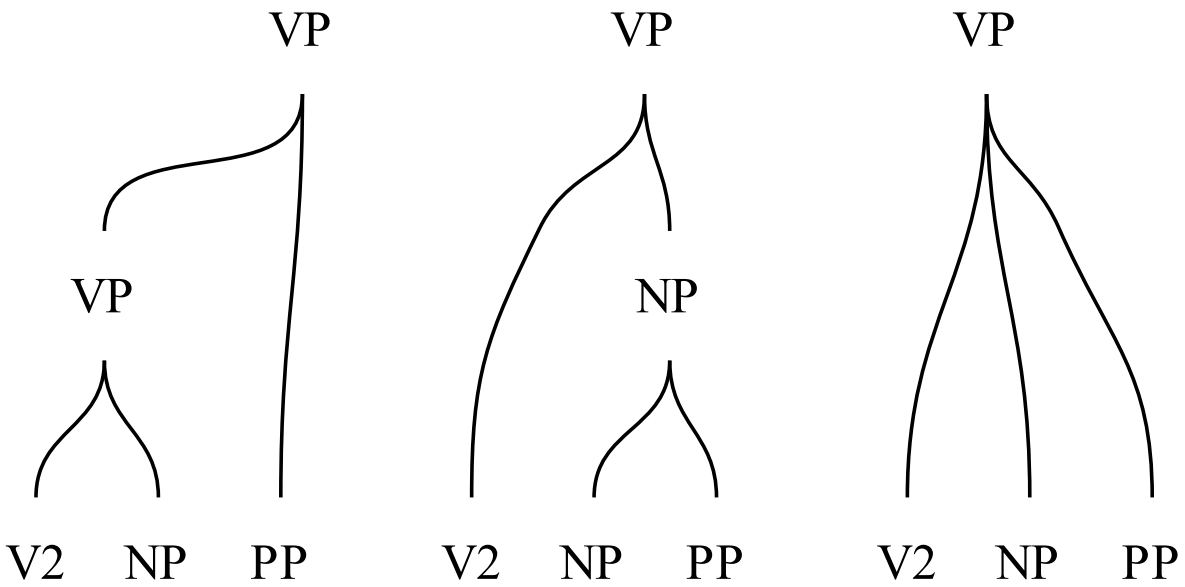
Solution to Problem 1

When and When?

- the user should **use predictive editing all the time**
- the UI should allow editing **not only of complete sentences but also of sub-phrases**

Solution to Problem 2

The Forest Deforestation



Incremental and Online Parser

Incremental Parser - reads the input from left to right and updates the parse forest incrementally after every consumed token

Online Parser - capable of producing partial parse tree even before the whole input was produced

New in GF: the parsing is now online since we can do deforestation of the partial parse forest

The First Prototype of the New UI

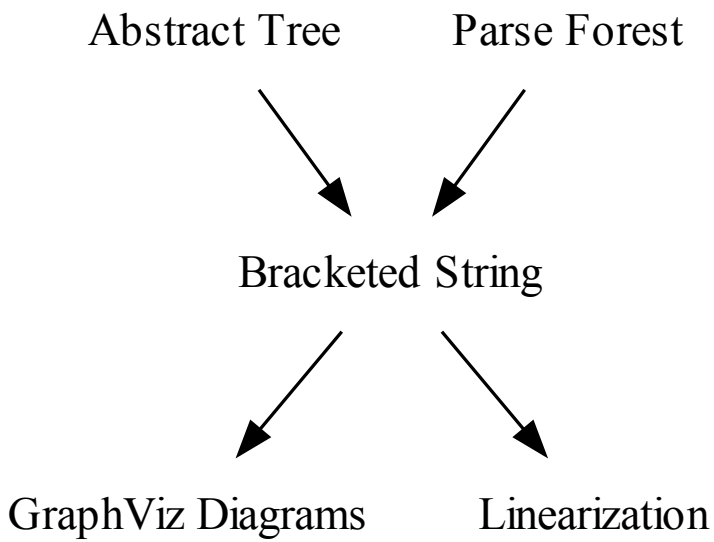
Available online but still hidden:

`http://www.grammaticalframework.org:41296/editor`

- uses the incrementality for word prediction and the online capabilities to produce **phrases like in the syntax editor**.
- still only **readonly phrases**
- autocorrection for **phonetic mutations**
- word completion for **variable names**
- integrated **type checking**
- **multi-line** editor
- improved **grammar browser**
- still under development - incompatibilities between the **different browsers**

Bracketed Strings

- Representation for text annotated with the matching categories
- Now central structure in the GF interpreter - both parsing and linearization



Example of Ambiguity

$(S_0\ (NP_1\ \mathrm{I})\ (VP_2\ (V2_3\ \mathrm{saw})\ (NP_4\ \mathrm{the\ man})\ (PP_5\ \mathrm{with\ the\ telescope})))$

$?0 = PredVP\ \ ?1\ \ ?2$

$?1 = i\_NP$

$?2 = AdvVP\ (ComplV2\ ?3\ ?4)\ ?5$

$\quad \mid ComplV2\ ?3\ (AdvNP\ ?4\ ?5)$

$?3 = see\_V2$

$?4 = the\_man\_NP$

$?5 = with\_the\_telescope\_PP$

Note: possible partial translation

Example of Discontinuity

($Command_{0,0}$ ($Action_{1,0}$ switch) ($Device_{2,0}$ the lights) ($Action_{1,1}$ on))

?0 = *CAction light* ?1 ?2

?1 = *switchOn light switchable*

...

Bracketed Strings for Robust Parsing

A Naive Receipe for Robust Parser:

- Parse with GF and in case of success, return the the parse tree
- If it fails return the bracketed string
  (S the original input is here)

More Seriously:

- Bracketed String can be produced by any **statistical parser**.
- **Partial abstract syntax trees** can be produced by GF for the fragments which are in the scope of the grammar.

Related:

- Mixed free text + controlled language

Incremental Type Checking

We type check the parse forest instead of individual trees.

This allows us:

- to type check shared sub trees in the forest only once
- to detect the type error locations in the input

Not integrated with the word completion - too hard.

Summary

- combines **Syntax Editing and Predictive Editing**
- completion for **variable names**
- autocorrection for **phonetic mutations**
- integrated **type checker**
- **multi-line** editor
- improved **grammar browser**
- some ideas for integration of **statistical parsers**