Statistical Parsing in GF (C runtime)

Krasimir Angelov, Aarne Ranta, Shafqat Virk,
Lauri Alanko

University of Gothenburg and University of Helsinki

MOLTO
Final presentation, May 2013

Overview

- The New C Runtime
  - statistical ranking
  - robust parsing
  - Python binding
- Applications in Translation
  - English Resource Grammar + Large Lexicon
  - Penn Treebank for GF
  - Large Scale GF Translation

The C Runtime

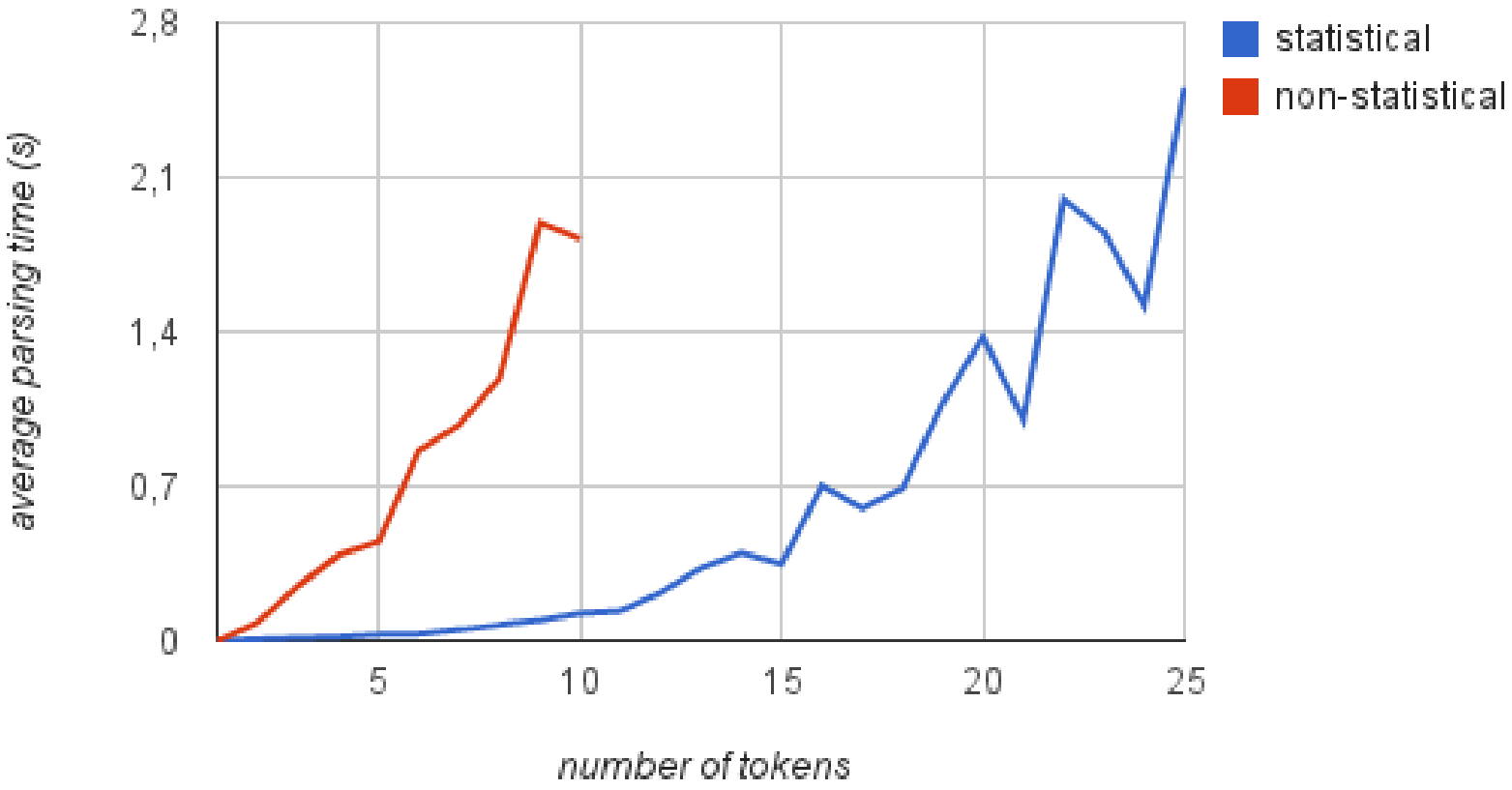
Initial Goals:

- small and efficient
- portable
- easy to embed in other languages

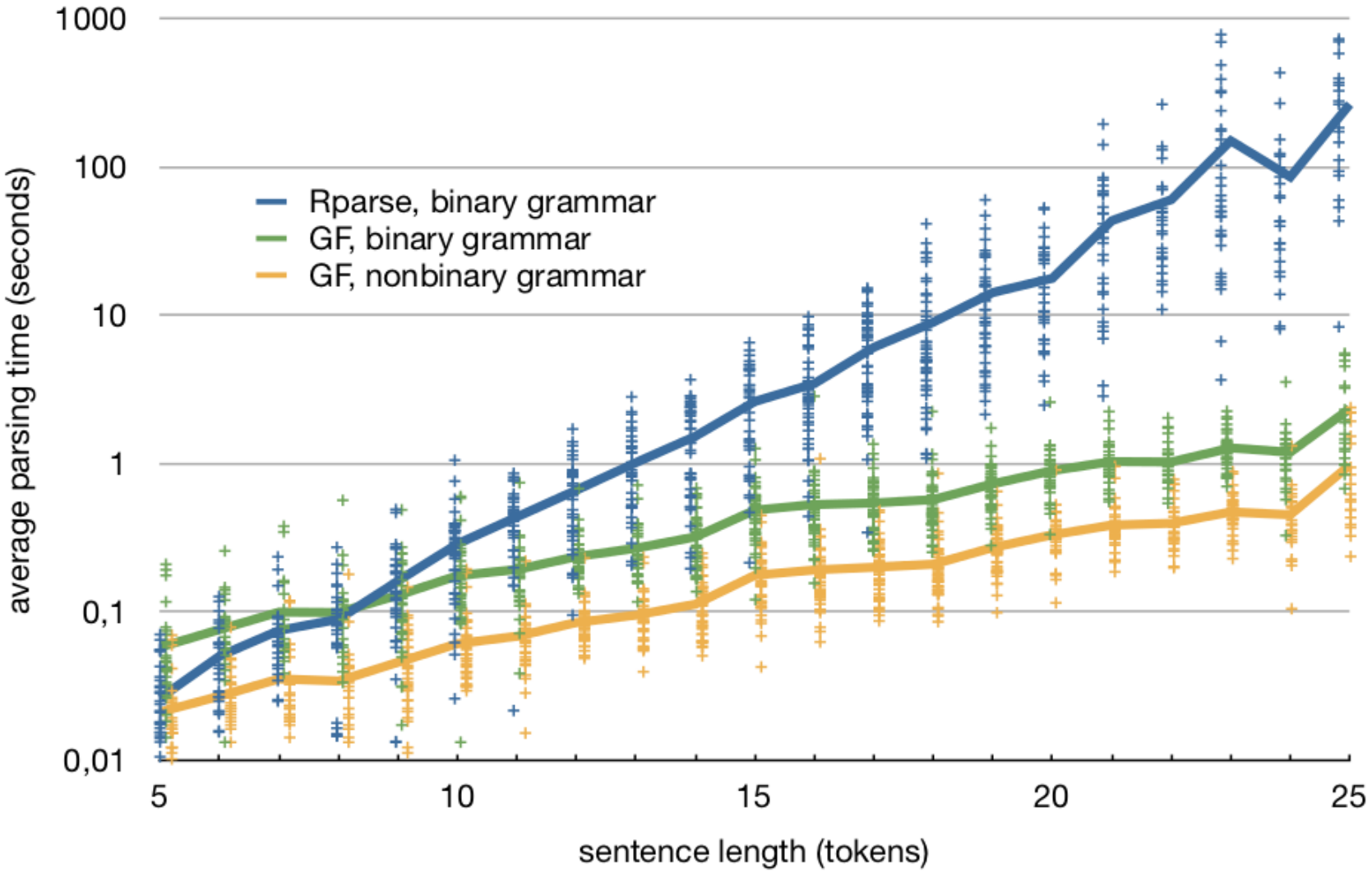
Grew Into:

- statistical engine for PMCFG which is **beyond the state of the art**
  - up to **100 times faster** than state of the art alternatives
  - the same precision
- the engine behind an experimental GF-based translator

The New vs The Old Parser



The GF Parser vs RParse



Using GF from Python

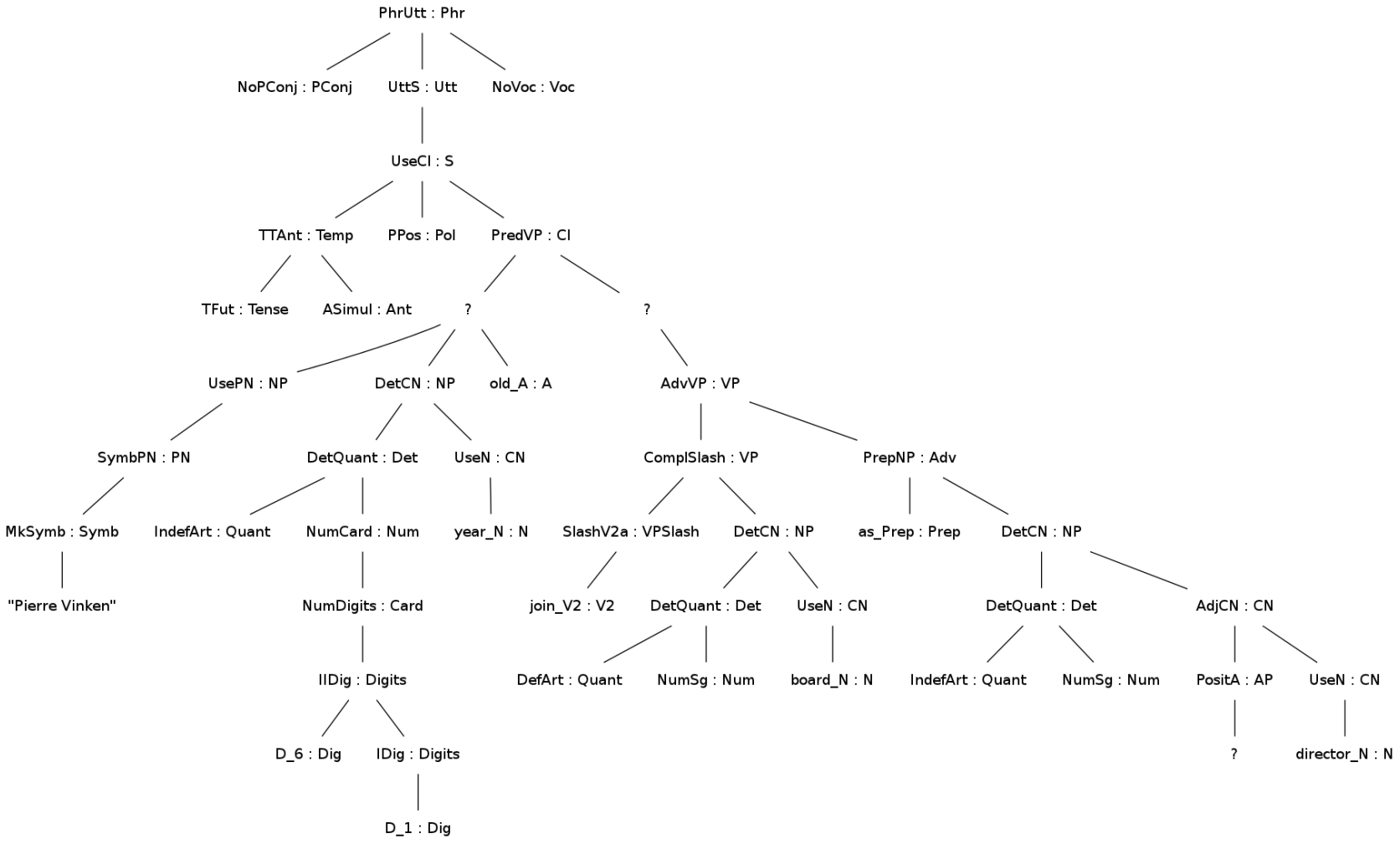
```
import pgf

gr = pgf.readPGF("ParseEngAbs.pgf")

try:
  for (p,e) in gr.languages["ParseEng"].parse(line, n=5)
    sys.stdout.write("["+str(p)+"] "+str(e)+"\n")
    print gr.languages["ParseBul"].linearize(e)
except pgf.ParseError as e:
  print e.message
```

Penn Treebank

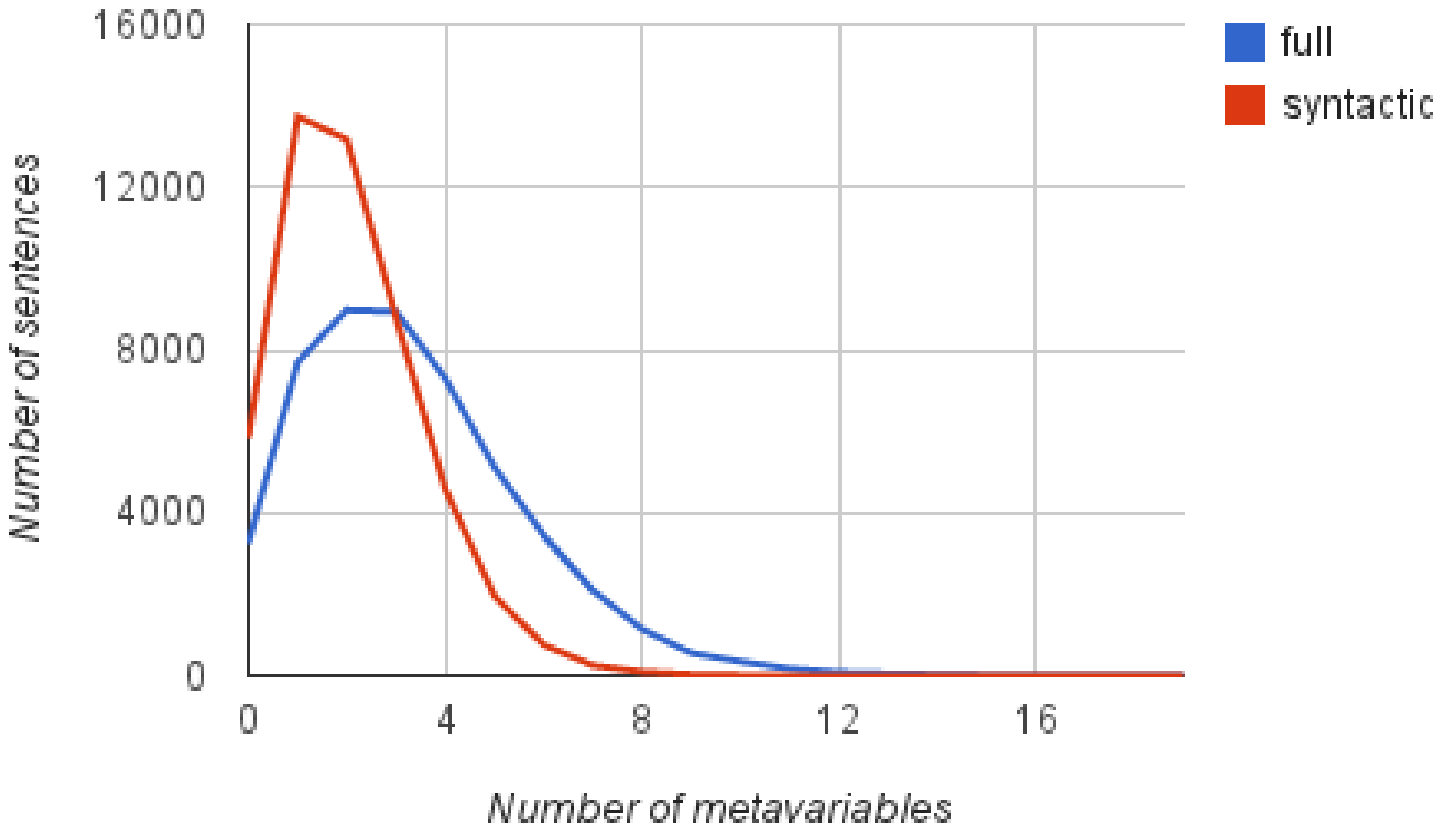
For statistical parsing we need a grammar and a statistical model:

- Training Data from the Penn Treebank
  - the original treebank has been converted to GF abstract trees
- English Resource Grammar + Large Lexicon of about 50 000 lemmas

Penn Treebank



Penn Treebank



Translation Service

- In GF translation is reduced to parsing in one language and linearization in another.
- An experimental online translation service is now available in the MOLTO translation interface

Demo!!

Summary

- **statistical processing** in GF is a new research direction
  - the beginning of new project and not the end
- the **translation service** is only proof of concept
  - need for better disambiguation
  - need for better translation dictionaries
  - need dictionary of idiomatic constructions