Formal Foundations for Semi-parsing

Vadim Zaytsev, Universiteit van Amsterdam
CSMR-WCRE 2014 ERA
CC-BY-SA
Poster in the next room

URGH! THRAK CREATE FIRE! FIRE HOT!

FIRE NEAT, THRAK, BUT ME INVENT CLUB! CLUB FOR HIT THINGS WITH.

OOH...

HELLO, FRIENDS! IT SEEMS THAT I HAVE INVENTED GRAMMAR.

THANKS! ME AND THRAK CAN COMMUNICATE FAR MORE EASILY NOW THAT WE HAVE HELPING VERBS AND ARTICLE ADJECTIVES.

"THRAK AND I"

Cyanide and Happiness © Explosm.net
The Grammar Hammer of 2012

Vadim Zaytsev, vadin@grammarware.net
Software Analysis and Transformation (SWAT) Team
Centrum Wiskunde & Informatica (CWI)
Amsterdam, The Netherlands
December 19, 2012

Contents
1 Introduction
2 Preliminaries
  2.1 Background notions
  2.2 Major contributions in a nutshell
  2.3 Minor contributions
  2.4 Motivation for this report
3 Topic overview
  3.1 Guided grammar consequence
  3.2 Grammar transformation languages
  3.3 Notation-driven grammar consequence
  3.4 Notation-parametric grammar consequence
  3.5 Non-parametric grammar consequence
  3.6 Implication diagrams
  3.7 Example diagrams
4 Venues
  4.1 Conferences
  4.2 Inquiries venues
5 Controlling resources
  5.1 Literature
  5.2 Software
6 Acknowledgements

1 Introduction

The purpose of this report is documenting personal research results of the year 2012 in a form primarily intended for assessment of their scientific merit as a foundation for future work, not for quantitative assessment of the resulting publication record. This can be considered as an aggressive form of self-publishing initiative [2,3] where scientific and engineering contributions are not only cataloged, but also put in perspective by a separate first-class scientific knowledge object. This report is mostly meant for my SWAT colleagues. However, it is open for broad audience and meant to be readable by any researcher with reasonable degree of familiarity with computer science. It can be consumed as a self-contained document, but many details are not pulled in from available referenced sources.

We start right away with a the overview of the field (§2.1) followed by brief descriptions of major (§2.2) and minor (§2.3) contributions, followed by a more elaborate motivation for creation of this document (§3.6). Next, all research topics are laid out in detail one by one [9]. For the sake of brevity, a separate overview of all involved venues (§4) is included. §5 concludes the report.

2 Preliminaries

2.1 Background notions

Software language is a concept that generalizes over programming languages, markup languages, database schemas, data structures, abstract data types, data types, modeling languages, ontologies, etc. Whenever we choose some degree of commitment to structure, we can identify it with a language, which elements (symbols) can be separately defined and the allowed combinations of them can be somehow specified. Studying software language engineering is important because of possibly gained insights into relations between the way such languages are defined and used in different technological spaces (e.g., we can study data binding as a way to map a relational database to an object model, or language conversations as a way to compare an XML schema with a syntax definition).
Grammar-based source code analysis

• A spectrum of approaches w.r.t. tolerance

• Tolerance increases from right to left

• Figure borrowed (for extension) from:

S. Klusener, R. Lämmel, Deriving Tolerant Grammars from a Base-line Grammar, ICSM 2003
Semi-parsing examples

- Skeleton grammars
  - Productions for interesting constructs are reused
  - Default productions used for the rest

- Robust multilingual parsing
  - Island grammars for multiple languages
  - Combined and stitched together

N. Synytskyy, J. R. Cordy, T. R. Dean, Robust Multilingual Parsing Using Island Grammars, CASCON'03.
All methods of semi-parsing

- ad hoc lexical analysis
- hierarchical lexical analysis
- lexical conceptual structure
- iterative lexical analysis
- fuzzy parsing
- parsing incomplete sentences
- island grammars
- lake grammars
- robust multilingual parsing
- gap parsing
- noise skipping
- bridge grammars
- skeleton grammars
- breadth-first parsing
- iterative syntactic analysis
- grammar relaxation
- agile parsing
- permissive grammars
- hierarchical error repair
- panic mode
- noncorrecting error recovery
- practical precise parsing

Boolean grammars

• Set theory:
  • union, intersection, complement
• Context-free grammars:
  • disjunction
• Conjunctive grammars:
  • disjunction, conjunction
• Boolean grammars:
  • disjunction, conjunction, negation

The use for conjunction

- Statement is...
  - keyword, expression, block
- Statement is also a chunk between dots/semicolons/...
- So, we define a statement
  - as a chunk and as a detailed statement
The use of negation

• An identifier can be anything... but a keyword
  • filtering

• Embedded SQL query
  • skip until “END-EXEC”

• Existing approaches are hard to compare
  • reject productions, lookahead restrictions, ordered disjunction, production priorities, ...

Parsings schemata

- Parsing process as a deduction system
  - Initial items (partial parse trees)
  - Deduction steps (based on production rules)
  - Final items (full parse trees, fact representations)
- Uniform spec of an algorithm
  - Implementations can vary

Conclusion

- Lots of methods => good
- Understanding => ???
- Mess needs to be cleaned up
- Can formal methods help?
- I’ll try
  - Boolean grammars
  - parsing systems
- Comfortaa by Johan Aakerlund (SIL OFL)
Conclusion

- Lots of methods => good
- Understanding => ???
- Mess needs to be cleaned up
- Can formal methods help?
  - I’ll try
    - Boolean grammars
    - parsing systems
- Comfortaa by Johan Aakerlund (SIL OFL)

related attempt: