Modern Parsing Techniques

All You Need is Generalized GLL

Semyon Grigorev

June 6, 2023

Parsing: Problems and Challenges

Language specification approaches Zoo

- Lexer + Parser vs Scannerless: grammar over tokens vs grammar over characters
- Context-free grammars (CFG): BNF, EBNF, ...
- Data-dependent grammars: utilization of previously parsed data to guide parsing
- Rail diagrams: graphical language
- Parsing Expression Grammars (PEG): CFG with prioritized choice and manual control of lookahead
- . . .

Language/grammar properties

- Determinism and ambiguity
- (Hidden) Left recursion
- . . .

Technical challenges

- Performance
- Error recovery
- Incremental parsing
- Manual control of behavior

ANTLR

- Adaptive LL(*) Parsing: The Power of Dynamic Analysis
- ALL(*)
- Huge grammar zoo
- Can generates parser in Java, C#, Python, JavaScript, Go, C++, Swift, Dart, PHP

+ Simple error recovery
+ Left recursion
+ Prioritized choice
+ Parsing tree and visitors, listeners for it

Grammar	KB/sec
XML	45,993
Java	24,972
JSON	17,696
DOT	16,152
Lua	5,698
C	4,238
Verilog2001	1,994
Erlang	751

Figure: Throughput of lexing+parsing; all input preloaded into RAM[2]

[2] From "Adaptive LL(*) Parsing: The Power of Dynamic Analysis"

Iguana

- GLL-based
- Sources
- Practical General Top-Down Parsers
- ✚ Data-dependent grammars
- ✚ Left recursion
- ✚ Prioritized choice
- ✚ Operator associativity
- ✚ Layout rules

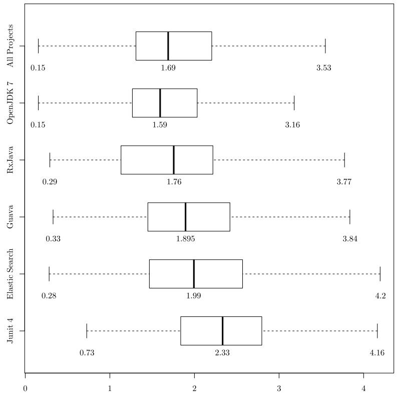

Figure: Iguana performance relative ot ANTLR[3]

[3] From "Practical General Top-Down Parsers"

Tree-Sitter

- GLR-based
- Sources

+ Huge grammar zoo
+ Incremental
+ Error recovery
+ Operators precedence and associativity

− Custom lexers to handle layout
− Parsers in C with bindings to other languages
− Prioritized choice

SDF3 (Spoofax)

- GLR-based
- ✚ Modular
- ✚ Layout rules
- ✚ Operators associativity
- −? Error recovery
- Incremental version under development[4]
- Part of Spoofax language workbench
- Sources
- Documentation
- Multi-purpose Syntax Definition with SDF3

[4]Incremental scannerless generalized LR parsing

Parser combinators

- 👎 Poor performance
- ✗ Left recursion
- ✗ Ambiguity
- ✗ Error recovery
- ✗ Incremental parsing
- 💡 Modern combinator library is not just a set of functions
  - 👍 Good performance
  - + Left recursion
  - + Ambiguity
  - ? Error recovery
  - ? Incremental parsing
  - 👎 You have no control on nontrivial machinery inside

Parsley

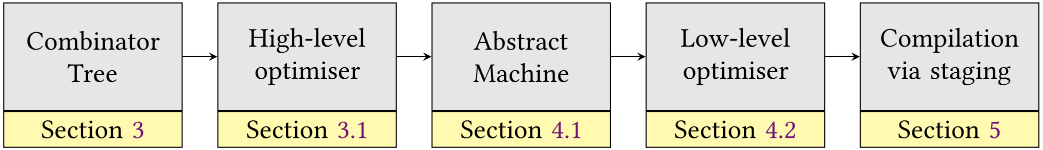

Figure: Parsley internal pipeline[5]

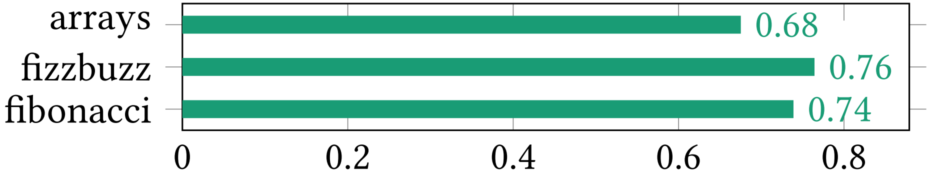

Figure: Parsley performance relative to Bison[6]

[5] From "Staged Selective Parser Combinators"
[6] From "Staged Selective Parser Combinators"

Meerkat

- GLL-like engine for CPS parsers
- Practical, general parser combinators
- ✚ Left recursion
- ✚ Prioritized choice
- ✚ Operator associativity

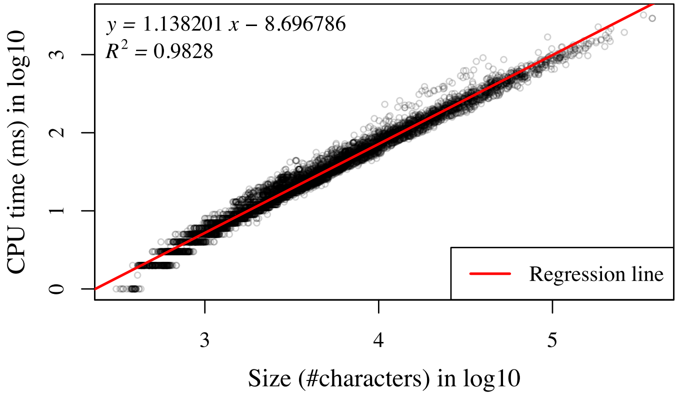

Figure: Meerkat performance on Java files[7]

[7] From "Practical, general parser combinators"

Summary

- ANTLR is a good choice in general case
- TreeSitter may be a good choice for incremental parsing
- Generalized parsing (GLL, GLR) is mature enough to be a base for real-world parsing tools
- Incremental parsing + precise error recovery = nontrivial challenge
    - Don't Panic! Better, Fewer, Syntax Errors for LR Parsers