Meerkat Parsers
General Top-down Parser Combinator Library

Anastasia Izmaylova[1], Ali Afroozeh[1], Tijs van der Storm[1]

Centrum Wiskunde & Informatica, 1098 XG Amsterdam, The Netherlands
anastasia.izmaylova@cwi.nl, ali.afroozeh@cwi.nl, storm@cwi.nl

Parser combinators are higher-order functions defined in terms of grammar constructs such as sequence and alternation. In this view of parsing, compared to traditional parser generators, grammars are represented as executable code. Parser combinators provide several benefits over parser generators. First, directly executable grammars eliminate the parser generation phase, making parser combinators ideal for fast language prototyping and experimentation. Second, integration with the host programming language enables easy composition and extension.

Parser combinators have been popular in the Haskell community, most notably the Parsec [1] combinator library. In recent years, with the popularity of Scala as a mixed functional and object-oriented language, the Scala parser combinator library [2] have gained attention as an attractive way of writing embedded parsers. All existing parser combinator libraries are based on recursive-descent parsing, as a result they all inherit its shortcomings. Naive backtracking techniques that extend recursive-descent parsing have exponential worst-case performance, and they cannot directly support left-recursion.

In this talk, we present Meerkat parsers, our Scala-based general top-down parser combinators library that provides the flexibility and direct implementation of plain recursive descent parsing with the expressiveness and performance guarantees of general parsing. Meerkat parsers are able to deal with any grammar, including the ones with (indirect) left recursion, and produce a shared packed parse forest (SPPF) in $O(n^3)$ time and space. The mechanism to deal with left-recursion is based on Johnson's continuation passing-style recognizers [3] and we produce Scott and Johnstone's binarized SPPFs [4,5]. As such it combines best of both worlds: the flexibility of parser combinator style parsing, and the expressive power of state-of-the-art general parsing algorithms.

The focus of this talk is on introducing the Meerkat parsers as a parser combinator library. More specifically we talk about the following features of Meerkat parsers.

Syntax

- Basic combinators
- EBNF combinators
- Left recursive rules
- Simple Grammars
- The Specification Grammar of Java

Extensibility

- Disambiguation filters
- Data-dependent parsing

Performance results

- Parsing highly ambiguous grammars
- Parsing Java

References

1. Leijen, D., Meijer, E.: Parsec: Direct style monadic parser combinators for the real world. Technical report (2001)
2. Moors, A., Piessens, F., Odersky, M.: Parser combinators in scala. Technical report, Katholieke Universiteit Leuven (2008)
3. Johnson, M.: Memoization in top-down parsing. Comput. Linguist. **21**(3) (September 1995) 405–417
4. Scott, E., Johnstone, A., Economopoulos, R.: BRNGLR: a cubic Tomita-style GLR parsing algorithm. Acta informatica **44**(6) (2007) 427–461
5. Scott, E., Johnstone, A.: GLL parse-tree generation. Science of Computer Programming **78**(10) (October 2013) 1828–1844