Pratt Parser

Provjeri di ide citat koji

Ultimately, the complexity in the process of parsing comes down to the resolution of this ambiguity. The technique we will develop here uses token objects whose members include binding powers (or precedence levels), and simple methods called nud (null denotation) and led (left denotation). A nud does not care about the tokens to the left. A led does. A nud method is used by values (such as variables and literals) and by prefix operators. A led method is used by infix operators and suffix operators. A token may have both a nud method and a led method. For example, - might be both a prefix operator (negation) and an infix operator (subtraction), so it would have both nud and led

methods.https://crockford.com/javascript/tdop/tdop.html

**Top Down Operator Precedence Parsing**

**Introduction**

Sto je to

“Parsing is the process of structuring a linear representation in accordance with a given grammar” (Grune and Jacobs, 2008). Such broad abstraction of definition implies great application area therefore it is of critical value to implement it in optimal way into code. Many contributors have developed methods for parsing. In effort to explain the mechanics behind this there were introduced //raznolike grammars and automatons. This was believed to be instrument for explaining and implementing advanced automatons and algorithms for parsing (izvor). In the process of finding optimal algorithm some have grown aversion towards theory and were practicaly oriented (Minsky 1970) // provjeri sta je zapravo zelio.

Many contributors have developed algorithms for parsing that heavily rely on formalisms (LR(k), BNF grammar, etc) or on the other hand

In 1973 Vaughan Pratt suggested another view to this problem. Instead of defining language syntax and then writing parser he proposed writing syntax directly in algorithm. He stated that

This parser does not require grammar because it is embedded in algorithm. Pratt explained that this is compromise between those who build parsers relying on syntax and those who do not(Pratt, 1973).

Kako radi

**Related work**

There are many alternatives to pratt parser.

Parsers can be divided into two main groups; top-down and bottom-up (Grune and Jacobs, 2008).

Most famous are LR and LALR parsers.

Kakva jos parsiranja postoje

**Problem and solution**

Koji problemi se mogu svest na ovaj

Recursive descent parsers have drawback when parsing left recursive grammars. i.e. for grammar

S → Sa | a

Pseudocode would look something like this:



As I stated; parsing is big domain. In the world of compilers

Besides using parsers for parsing data it can be put to use in favour of data compression, generating machine instructions in compilers and as a support for logic language.

Data compression is achieved by swapping big and common chunks with smaller. This swapping (or maping) is threated same as in grammar transitions therefore parser can be utilized for this task. Generating instructions is also done by obeying grammar rules that (Grune and Jacobs, 2008).

**Experiments**

There have been made many articles on the topic of *top down operator precendance* algorithm. One of implementations using toy language is extensively explained in article[[1]](#footnote-1).

Pratt stated that his algorithm has been implemented in SCRATCH-PAD and in MACSYMA (Pratt, 1973).

Crockford exploited this algorithm for his application JSlint[[2]](#footnote-2). He also wrote a paper on how it works[[3]](#footnote-3) and displayed code publicly[[4]](#footnote-4).

Another interesting use is for application Desmos[[5]](#footnote-5). They made an article[[6]](#footnote-6) on how they did and mentioned that speed was positive side while negatives were memory overflow because of recursion calls and no guarantee that parser will work intendedly for every case (Lantsman, 2018).

**Conclusion**

Jel cemu ovo

Za sta je dobar

Top down operator precedence parsing is best utilized when is in service of dynamic and functional programming languages (Crockford, 2007).

**References**

PRATT, Vaughan R. 1973. "Top down operator precedences". *Proceedings of the 1st annual ACM SIGACTSIGPLAN symposium on Principles of programming languages*, 41-51

GRUNE, Dick and Ceriel J.H. JACOBS. 2008. *Parsing Techniques*. New York: Springer Science+Business Media, LLC.

AHO, Alfred V. et al. 1986. *Compilers: Principles, Techniques, & Tools*. Boston: Addison-Wesley

CROCKFORD, Douglas. 2007. *Top Down Operator Precedence*. Available at: https://crockford.com/javascript/tdop/tdop.html (Accessed: 13 January 2021).

LANTSMAN, Denis. 2018. *How Desmos uses Pratt Parsers*. Available at: <https://engineering.desmos.com/articles/pratt-parser/> (Accessed: 13 January 2021).

KLADOV, Aleksey. 2020. Simple but Poweful Pratt Parsing. Available at: <https://matklad.github.io/2020/04/13/simple-but-powerful-pratt-parsing.html> (Accessed: 13 January 2021)

1. https://eli.thegreenplace.net/2010/01/02/top-down-operator-precedence-parsing# [↑](#footnote-ref-1)
2. http://jslint.com/ [↑](#footnote-ref-2)
3. https://crockford.com/javascript/tdop/tdop.html [↑](#footnote-ref-3)
4. https://github.com/douglascrockford/JSLint [↑](#footnote-ref-4)
5. https://www.desmos.com/ [↑](#footnote-ref-5)
6. https://engineering.desmos.com/articles/pratt-parser/ [↑](#footnote-ref-6)