

# Compiler Construction - Parsers

## The Road So Far

Dennis Verheijden <sup>1</sup>    Alan Andrade <sup>2</sup>

<sup>1</sup>Data Science <sup>2</sup>Software Science  
Radboud University Nijmegen

Parser Presentation  
15th March 2018

# Outline

Parser Specifications

Encountered Problems

Pratt Parser

Results

Conclusion





# Parser Specifications

- Top-Down Parser
- Split up in a Scanner and Parser
- Written in Java, so Recursive Descent
- Pivoted later to a Top-Down Operator Precedence Parser (more later).



# Left Recursion

Old Grammar

Here goes the old grammar

New Grammar

New Grammar

# Associativity

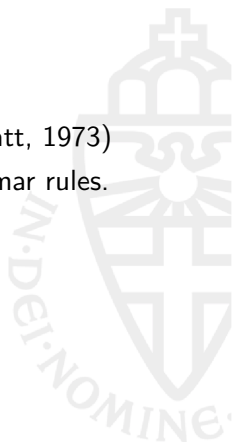
Changing the grammar resulted in the following:

$$\begin{aligned} a * b * c * d &\rightarrow (a * (b * (c * d))) \\ &\rightarrow (((a * b) * c) * d) \end{aligned}$$

So we should change the grammar again... OR **Pratt Parser**

# Pratt Parser

- Improved Recursive Descent Parser (Vaughan Pratt, 1973)
- Associate semantics with tokens instead of grammar rules.
- Rewrite grammar rules to individual Parselets.
- Operators now have a precedence value.
- Complete Grammar Rewrite was needed.





# Example



# Tests

Fun Examples Here





## Journey So Far

- Worked on the project side-by-side for 30 hours
- Creating a parser by hand is tedious without the use of tools
- In hindsight, we were maybe better off using functional languages, or even better hybrid languages like F#.
- 2.5k lines of code (40% Test)