# Macros em Scheme e Sistema de tipos

Bruno de Freitas Bonatto
Lucas Corsac
Lucca Strelow Milano

# O que são macros?

Geralmente pensamos em dois tipos de macros:

- Estilo C e assembly
  - Substituição de sequências de caracteres
  - Quase um Find/Replace
- Estilo Excel e comandos Vim
  - Macros mais poderosos
  - Gravam uma sequência de operações e as repetem sobre o input

# O que são macros?

Em ambos os casos anteriores tratamos o input como uma sequência linear de caracteres, ou no caso do excel como um conjunto de dados com pouca estrutura. Mas podemos pensar de outra forma: macros são funções de texto para texto, no caso de macros em linguagens de programação podemos dizer que são funções de programa para programa!

# Macros em Scheme (((((LISP)))))

Em Scheme, e outras Lisps, programamos usando uma abstração diferente: nossos programas não são sequências de caracteres ASCII, mas sim árvores de símbolos, levamos a ideia de “Code is Data” ao extremo. Nossos macros refletem isso se tornando mais poderosos!

# Macros em Scheme (((((LISP)))))

Como programas em Scheme refletem diretamente as suas ASTs nossos macros podem usar essa estrutura para fazerem “substituições” mais específicas. Como macros são funções arbitrárias eles podem ter qualquer resultado!

# Usando macros para criar DSLs

A criação de DSLs (Domain Specific Languages) em linguagens imperativas é extremamente trabalhosa, quase precisamos criar um compilador para a linguagem, o que diminuí muito uma das maiores vantagens de usar uma DSL: a facilidade de criação.

# Demonstração!

# Usando macros para criar DSLs

Podemos usar os macros mais poderosos de Scheme para traduzir da nossa DSL para a linguagem base, quase como um compilador.

# Demonstração!

# Macros como sistema de tipos

Como vimos podemos usar macros podemos usar macros para transformar programas de uma linguagem em outra. Nada impede que nossa DSL tenha um sistema de tipos mesmo se nossa linguagem base não tenha!

# Macros como sistema de tipos

Utilizamos uma representação intermediária e macros que agem sobre essa representação para fazer o *typecheck* então convertemos para a linguagem base que executa o programa.

# Macros de linguagem em (((((((((Racket)))))))))

Racket foi feita com o propósito de facilitar a definição de novas linguagens embutidas, exemplos incluem scribble uma linguagem para a criação de pdfs, datalog uma linguagem de programação lógica estilo prolog, e uma linguagem para criação de guis e uma para a criação de *web servers*.

# Macros de linguagem em (((((((((Racket)))))))))

Racket é boa para a criação de linguagens porque permite o acesso do programador ao seu lexer e parser (em Racket chamado de *reader*) e seu analisador semântico (o *expander*) que o programador pode substituir e alterar, assim como ter acesso à informações da representação interna da sintaxe.

# Demonstração!

# Obrigado pela atenção!