Linguagens de Programação

Nuno Carriço & Rúben Peixoto
38489 & 37514

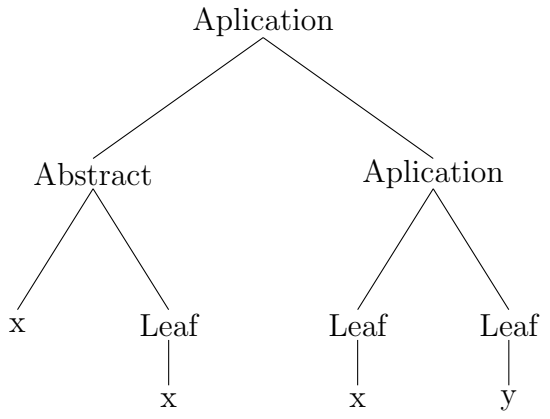
Abril 2019

Introdução

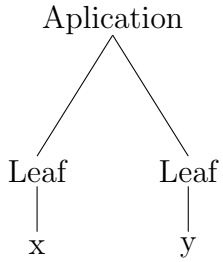
Este trabalho consiste na conversão de uma expressão λ em beta equivalente.

Para isto o grupo usou o método call-by-name. A razão pela qual o grupo utilizou este método foi pelo facto de se conseguir encontrar sempre uma expressão que esteja na forma irredutível, isto é, chega-se a uma expressão onde não se consegue encontrar nenhum redex.

Árvores Sintáticas



Árvore sintática antes da aplicação do algoritmo



Árvore sintática após a conversão para beta equivalente

Funcionamento do Interpretador

O algoritmo criado pelo grupo consiste em percorrer a árvore sintática construída anteriormente até encontrar uma ramo do lado esquerdo que seja uma abstração. No lado direito tem que estar obrigatóriamente uma abstração, uma aplicação ou uma folha. Com isto é identificado o redex necessário.

Posto isto, e como se quer o ramo do lado direito no ramo do lado esquerdo, vai se identificar a primeira variável ligadora dessa abstração. De seguida, cria-se um dicionário onde a chave é a variável ligadora e o valor é o ramo direito.

Agora o programa vai percorrer o ramo esquerdo, que é uma abstração, de modo a procurar as variáveis que correspondam à chave do dicionário. Ao encontrar essas variáveis ligadas, estas vão ser substituidas pelo valor que está no dicionário. Esse valor não é nada mais do que o ramo do lado direito que foi previamente identificado e guardado no dicionário.

É importante referir que, como a função beta equivalência só faz uma substituição de cada vez, a cada chamada de função beta equivalente é aplicado a função alpha equivalência. Tal foi feito para que as variáveis tivessem nomes diferentes e assim distinguir as variáveis, sendo mais fácil de saber quais as variáveis ligadas que precisam realmente de ser substituídas.

Exemplos Extra

Utilizando os exemplos do primeiro trabalho temos:

\fx.(f x) x

Usou-se este exemplo para verificar que a execução sem parêntesis a envolver as variaveis ligadoras.

input <- \fx.(f x) x
output -> \fx.(f x) x

x \fx.(f x)

Como foi definido uma variável que não tem uma variável ligadora, o programa considerar que essa variável é uma variável livre.

input <- x \fx.(f x)
output -> x \fy.(f y)

(\fx.(f x)) x

Usou-se este exemplo para verificar que a execução de uma expressão com parêntesis envolvendo as variaveis ligadoras e um termo.

input <- (\fx.(f x)) x
output -> \a.(x a)

\f.f y \y.y (\z.z y)

Este exemplo tem como objetivo mostrar que, se uma variável é usada antes da sua variável ligadora então aquela é considerada variável livre.

input <- \f.f y \y.y (\z.z y)
output -> y (\z.z y)

\x.(\y.x z) (\z.z) x (\x.x (\w.w) x) x

Mostra que o, neste caso a variável ligadora x se estende o mais à direita no ambito desta abstração.

input <- \x.(\y.x z) (\z.z) x (\x.x (\w.w) x) x
output -> \x.x z x (\b.b (\w.w) b) x

Webgrafia

Stackoverflow:
- https://stackoverflow.com/questions/2612802/how-to-clone-or-copy-a-list