



Universidade de Évora
Escola de Ciências e Tecnologia
Licenciatura em Engenharia Informática

Linguagens de Programação 2016/2017

Primeiro Trabalho Prático

Interpretador de Cálculo Lambda

Docente:

Teresa Gonçalves

Discentes:

David Parreira, nº 33257;

Rute Veladas nº 33402.

Évora, 7 de Maio de 2017

Introdução

No âmbito da disciplina de Linguagens de Programação foi solicitada a elaboração de um interpretador de cálculo lambda, que transforme o termo lambda original no termo a-equivalente onde todas as variáveis têm nomes distintos. Para tal optou-se por utilizar linguagem de programação C recorrendo aos analisadores sintáticos Flex e Bison.

Funcionamento do Interpretador

A primeira fase da realização do nosso trabalho consiste na implementação de um analisador lexical e de um analisador sintático para a nossa linguagem. O analisador lexical foi implementado através do uso do Flex e o analisador sintático foi implementado através do uso do Bison.

Os terminais da linguagem são os seguintes:

- As letras, que são minúsculas:
[a-z] {
yylval.letter = strdup(yytext);
return LETTER;
}
- Lambda, que pela notação escolhida é o equivalente a "λ":
"λ" return LAMBDA;
- Ponto:
"." return PONTO;
- Parêntesis esquerdo:
"(" return PARE;
- Parêntesis direito:
")" return PARD;
- Espaços, novas linhas e tabs (serão ignorados):
[\n\t]

Para o analisador sintático procedeu-se à elaboração das regras da gramática, e a gramática utilizada foi a seguinte:

program: terms ;

terms : term
 | term terms
 ;

term: abs
 | app
 | PARE terms PARD
 ;

abs: LAMBDA LETTER PONTO terms ;

app : LETTER

| LETTER terms

;

A regra *terms* da gramática utilizada permite ao programa analisar se existem um ou vários termos.

A regra *term* define que um termo pode ser uma abstração, uma aplicação, ou qualquer uma destas dentro de parêntesis.

A regra *abs* define em que consiste uma abstração e a regra *app* define em que consiste uma aplicação, sendo que uma aplicação pode consistir numa única letra, ou no conjunto de vários termos.

Adotaram-se ainda as seguintes convenções fornecidas pela professora no enunciado:

- a aplicação é a construção com maior prioridade;
- a aplicação associa à esquerda;
- o corpo de uma λ -abstração estende-se para a direita até onde for possível.

A segunda fase da realização do trabalho consiste na construção do termo a-equivalente a partir do termo lambda encontrado.

O interpretador lê um termo lambda através de standard input e caso não ocorra nenhum erro sintático, escreve no terminal o termo original e o termo a-equivalente e o programa acaba. Se for detetado algum erro sintático o programa termina com essa indicação.

Exemplos Elaborados

O nosso interpretador foi testado com todos os exemplos fornecidos pela docente no enunciado. A gramática elaborada aceita todos os casos, porém o interpretador não funciona completamente em casos como o exemplo 2 abaixo referido, sendo este o único problema encontrado no programa.

1. $\leftarrow (!x. (x\ y\ (!a.a))\ (!x.\ x))$

$\rightarrow (!o\ (o\ y\ (!m.m))\ (!s.s))$

Mostra o caso de uma única abstração cujo termo após o ponto é uma aplicação composta por dois termos, em que um deles é composto duas aplicações seguidas de uma abstração.

2. $\leftarrow (!x.\ !y.\ x\ y)\ y\ (!x.\ \underline{x}\ (!x.\ x)\ x)$

$\rightarrow (!o.\ !m.\ o\ m)\ y\ (!s.\ \underline{x}\ (!b.\ b)\ s)$

Mostra o caso de uma variável independente (y) e uma abstração x com outra abstração x dentro. Não funciona completamente neste exemplo pois não substitui a variável antes da abstração.

3. $\leftarrow !x. (!a. a \ y) \ (!x. x)$
 $\rightarrow !o. (!m. m \ y) \ (!s. s)$

Mostra outro caso de uma variável independente (y) dentro de duas abstrações x e a respectivamente.

4. $\leftarrow !x. x \ y$
 $\rightarrow !o. o \ y$

Aqui demonstramos o caso mais básico de uma abstração simples com a variável independente y .

Bibliografia

- Slides/Exercícios fornecidos pela docente.
- <http://www.utdallas.edu/~gupta/courses/apl/lambda.pdf>