MAC 316 – Conceitos Fund. de Linguagens de Programação

Prof: Ana C. V. de Melo
Terceiro Trabalho – Implmentação de uma linguagem de expressões
Prazo de Entrega: até dia 10/06/2012
Número de integrantes da equipe: 5 (máximo)

SML - programação funcional

Considere uma linguagem funcional simplificada (LFSimp) definida informalmente como segue (sintaxe concreta):

```
SimbolosTerminais : fun, if, (, ), +, -, *, And, Or, ==, [, ]

Programa    ::= Aplicacao
Aplicacao   ::= (DecFuncao, ListaExp)
ListaExp    ::= [Expressao, ...]        {0 ou mais expressões}
DecFuncao   ::= (''fun'', ListaParam,  Expressao)
ListaParam  ::= [Id, ...]               {0 ou mais identificadores}
Id          ::= a | ... | z             {apenas 1  letra}
Expressao   ::= Valor | ExpBinaria | IfThenElse | Aplicacao
Valor       ::= ValorInteiro | ValorBooleano | ValorFuncao
ExpBinaria  ::= ExpAritm | ExpBool
ExpAritm    ::= (Expressao, ''+'',   Expressao)
              | (Expressao, ''-'',   Expressao)
ExpBool     ::= (Expressao, ''And'', Expressao)
              | (Expressao, ''Or'',  Expressao)
              | (Expressao, ''=='',  Expressao)
IfThenElse  ::= (''if'', ExpBool, Expressao, Expressao)

ValorInteiro ::= ... -1 | 0 | 1 ...  {valores inteiros de SML}
ValorBooleano::= true | false        {já definidos em SML}
ValorFuncao  ::= Aplicacao           {valor resultante da
                                      aplicação de uma função}
```

Esta linguagem não possui declaração de tipos e cada um dos operadores, tanto aritmético quanto booleano, é aplicado a expressões. Neste trabalho não realizaremos a verificação de tipos sobre as funções criadas, nem sobre os operadores predefinidos. Assim, consideramos que os tipos associados a aplicação das funções estão sempre corretos.

A avaliação de um programa é a aplicação da definição da função aos termos (expressões) correspondentes: o primeiro termo é associado ao primeiro parâmetro e assim sucessivamente. A avaliação das expressões binárias é a usual: o operador só poderá ser aplicado depois de avaliado cada um dos operandos. Note que os valores considerados foram apenas informalmente definidos. Aqui, consideramos que os tipos inteiro e booleano da linguagem são os mesmos definidos em SML. O parâmetro atual de uma função pode ser a aplicação de uma outra função (o que chamamos de função de alta-ordem).

Uma outra simplificação usada aqui é a ausência de nomes para as funções (assim como temos no λ–cálculo). Um programa é então um par contendo a definição de uma função junto com uma lista de termos para a aplicação. Dessa forma, um termo de aplicação é uma expressão e pode, portanto, ser tanto um valor quanto uma expressão binária ou uma aplicação de função. Exemplo:

`((''fun'', [x,y], (x,''+'',y)), [5,(3,''+'',3)])`. Neste exemplo, temos a aplicação de uma função (soma de dois números) às expressões 5 e `(3,''+'',3)`, tendo como resultado o valor 11 (5 é associado a x e a expressão `(3,''+'',3)` é associada a y).

Você deve implementar, em SML, um sistema de redução (avaliação) para a linguagem sugerida acima (veja as reduções para λ-cálculus no Capítulo 8). Nesse sistema, devem ser implementadas ambas as estratégias de redução estudadas: *sob demanda* e *a priori*. Considere que todas as expressões são bem-formadas, e note que elas podem ser facilmente manipuladas em SML. Por isso, não há necessidade de desenvolver um *parser*, basta desenvolver as estratégias de avaliação sobre as expressões (tuplas). Crie quantas funções julgar necessário, mas o seu sistema deverá ter, no mínimo, as funções:

```
fun  sobdemanda ...      {avaliação sob demanda}
fun  apriori ...         {avaliação a priori}
```

O resultado da avaliação de um programa deve ser mostrado passo a passo: o primeiro passo da avaliação, segundo, ...

A maioria das ferramentas para SML são de domínio público, a mais recomendada por ter bastante material de apoio é a SML/NewJersey, a qual foi desenvolvida por um consórcio de universidades e empresas: http://www.smlnj.org/

Existem alguns tutoriais sobre SMLa partir do site do smlnj, aqui estão alguns recomendados:

- Tutorial com alguns conceitos associados:
  `http://homepages.inf.ed.ac.uk/stg/NOTES/`
- Tutorial com dicas para a implementação de linguagens:
  `http://www.cs.cmu.edu/~rwh/smlbook/book.pdf`
- Tutorial + uso do smljn (bibliotecas e ferramentas):
  `http://www.cs.cornell.edu/riccardo/prog-smlnj/notes-011001.pdf`
- Um guia de sobrevivência smlnj:
  `http://www.cs.cmu.edu/afs/cs/local/sml/common/smlguide/smlnj.htm`

OBS: é proibido usar variáveis de referência neste EP.