Apresentação do compilador Dante P2

Autores: João Pedro Abreu, Luis Freitas e Raphael Leardini

Data: 13/06/2018

Universidade Federal Fluminense

Parser

```
(define-peg statement (or declaracao comando))

(define-peg bloco (and wordSeparator (name t1 declaracao)
wordSeparator (name t2 comando)) (blk t1 t2))

Ex:
(define-peg loop (and while spaces
(name condicao boolExp) wordSeparator do spaces
"{"wordSeparator
(name corpo (or bloco comando)) wordSeparator "}")
(whileDo condicao corpo))
```

Parser

```
(define-peg declaracao (or decSeq decUnit))

(define-peg decSeq (and wordSeparator (name t1 decUnit)
(? wordSeparator pointvirg wordSeparator
(name t2 decSeq)))
(cond [t2 (decSeq t1 t2)] [else t1]))

(define-peg decUnit (or constanteBlk variavelBlk))
```

Parser

```
Ex:
(define-peg variavelBlk (and wordSeparator
var wordSeparator (name t1 iniSeq))
(variavelBlk t1))

(define-peg iniSeq (and wordSeparator
(name t1 inicializacaoBlk)
(? wordSeparator virg wordSeparator (name t2 iniSeq)))
(cond [t2 (iniSeq t1 t2)] [else t1]))

(define-peg inicializacaoBlk (and wordSeparator
(name t1 variable) wordSeparator
"="wordSeparator (name t2 (or boolExp aritExp)))
(init t1 t2))
```

Tradução BPLC

Para a tradução para BPLC nada muito rebuscado foi necessário.

Tivemos que incluir um novo conversor para as estruturas de bloco que criamos. Esse conversor por sua vez divide o bloco em duas partes, o variavelBlk(Bloco de Variaveis) e constanteBlk(Bloco de constantes).

Por sua vez eles fazem o match corretamente.

Não tivemos problemas nessa parte, pois as alterações foram minimas.

SMC

O (S,M,C) foi extendido para (E,S,M,C,L) sendo o E o ambiente e o L a lista de localizações. Esta lista é utilizada, quando do retorno do bloco, para limpar a memoria das variaveis criadas internamente. Atualmente a única parte não-funcional(entenda funcional como o paradigma) do codigo é a execução do print e do exit, mas isso poderia ser resolvido extendendo para um (O,E,S,M,C,L) onde o O é uma lista de efeitos colaterais, como descrito na especificação(não realizado ainda).

SMC - Problemas

Assim como na primeira parte, não houveram percalços significativos na execução desta. O smc basicamente se tornou uma tradução assistida da especificação, sem necessidade de entender o que se esta fazendo(nós entendemos).

SMC

Toda operação que "altera"o ambiente e a memoria foi encapsulada no modulo "ambiente.rkt",fazendo com que o smcEval seja um código de reescrita quase puro e simples.

Exatamente por isso podemos extender essa implementação para a utilização de uma função de avaliação, a qual aparece na especificação como "val"e igualmente serve para devolver o valor ou o tipo, fazendo com que, sem uma única linha adicionada, o smcEval possa fazer type-checking.

O único porém seria a avaliação de loops, o qual exigiria algum contexto ou alguma tática ainda não plenamente compreendida.

Bloco

Quando o SMC encontra um bloco, ele salva o ambiente atual e a atual lista de localizações na pilha de valores, destrói o bloco e coloca no final do comando do bloco 'blk, para saber quando deve sair do bloco, ou seja, restaurar o contexto anteriormente salvo e limpar a memoria.

Cada operação de criação de variáveis dentro de um bloco adiciona a localização desta variável na lista de localizações. (i.e cada ref faz um cons, cada 'blk faz um free).

Ambiente

O ambiente é um hash cujas chaves são strings(identificadores) e os valores são (U Number Boolean Loc), onde Loc é uma estrutura que contém apenas um número como membro, para diferenciar variaveis de constantes.