Compiladores - Prova 1, gabarito

5/10/2017 - Prof. Fábio Macêdo Mendes

Q1-jedi (2.0 pts): Gramáticas simples

a) Não, pois as expressões que possuem apenas 1 "b" possuem pelo menos 3 aaa segundo as 2 primeiras regras de derivação.

b) Em s-exprs:

(E1 a (E2 b (E3 a) a (E3 a)) b (E3 a))

Onde distinguimos os três ramos da definição de E na gramática.

c) Sabendo que apenas E1 e E2 introduzem b's na gramática, basta provar que se expr1 e expr2 possuem um número de a's maior que b's, o mesmo vale para as expressões derivadas.

Seja a1 e b1 o número de a's e b's em expr1 e a2 e b2 para expr2. A análise para E1 e E2 é a mesma, já que ambas introduzem um valor de a e outro de b:

```
a_final = a1 + a2 + 1
b_final = b1 + b2 + 1
```

uma vez que a1 > b1, então a_final > b1 + a2 + 1. Sabendo que a2 > b2, também temos que `a_final > b1 + b2 + 1 = b_final`, QED.

(8 min)

Q1-padawan (1.5pts): Gramáticas simples

a) A expressão pode ser derivada como `(a * a) * a` ou `a * (a * a)`, fazendo a associatividade tanto à esquerda quanto à direita:

Derivação 1:

```
E -> E * E -> E * a -> E * E * a
  -> a * E * a -> a * a * a
```

Derivação 2:

```
E -> E * E -> a * E -> a * E * E
  -> a * E * a -> a * a * a
```

b) Uma gramática não ambígua seria: `E = E * a | a` que só permite associatividade à esquerda pois o novo elemento é sempre um átomo `a` colocado à direita.

(5 min)

Q2-jedi (3.0 pts): Linguagens regulares

a)		
[V] `42`	[V] `+42.`	[V] `-42`
[V] `3.1415`	[F] `.15`	[F] `1.2e10`
[F] `--1.5`	[V] `1.2.3`	[V] `1..10`

b) N -> (-|+|e) (dd*) (.d*|e)

c) N -> (-|+|e) (dd* | (d|dd|ddd)(_ddd) *) (.d*|e)

(7 min)

Q2-padawan (2.0 pts): Linguagens regulares

Linguagem regular:

```
N -> (d|dd|ddd)(_ddd)*
```

(3 min)

Q3-jedi (3.0 pts) Análize de código

a) Tokens

KEYWORD(DEF) NAME(say_hi) LPAR NAME(x) RPAR COLON INDENT NAME(print) LPAR STRING(“Hello”) OP(+) NAME(x) RPAR

b) Com s-exprs:

(DEF NAME(say_hi) (ARGS NAME(x)) (BLOCK (CALL NAME(print) (PLUS STRING(“Hello”) NAME(x)))))

(7 min)

Q3-padawan (2.0 pts) Análize de código

a) Tokens

NAME(X) EQ(=) NUMBER(10) OP(+) NAME(b) OP(*) NUMBER(42)

b) Com s-exprs:

(ASSIGN NAME(x) (PLUS (TIMES NUMBER(10) NAME(b)) NUMBER(42))

)

(5 min)

Q4-jedi (3.5 pts) Gramática para o JSON

Gramática BNF:

```
expr
    : atom
    | list
    | object

list
    : [ list_args ]

list_args
    : args , expr
    | expr

object
    : { pairs }

pairs
    : pairs , pair
    | pair

pair
    : STRING : expr

atom
    : true
    | false
    | null
    | NUMBER
    | STRING
```

(7 min)

Q4-padawan (2.5 pts) Gramática dos dicionários

Gramática BNF:

```
dictionary
    : { pairs }

pairs
    : pairs, pair
    | pair

pair
    : STRING : atom

atom
    : STRING
    | NUMBER
```

(4 min)