Lista de Exercícios 1

Compiladores

Prof. Leandro I. Pinto

Questão 1. Construa um autômato que aceite a seguinte linguagem:

a) $L = \{(\varepsilon|d^*|dd^*.dd^*|o)\ tal\ que\ d\ é\ um\ dígito\ e\ o \in \{+, -, *, /, '\backslash 0'\}\}$

b) Ex.: o, d, dd, dd.d, dd.ddd

Questão 2. Qual a gramática da linguagem desse autômato?

Questão 3. Implemente uma função que retorne os tokens de uma expressão aritmética;

a) int lexica(char *str, int *pos);

b) A função recebe uma string e uma posição nesta string. A partir de pos, a função procura pelo próximo token e o retorna, incrementando pos até a posição após o token encontrado.

c) Ex.: Entrada = 50 + 20

- Chamada 1 Retorna: CONST
- Chamada 2 Retorna: OPERADOR
- Chamada 3 Retorna: CONST

Questão 4. De acordo com a seção 3.3.3 de [Ref 1] responda:

a) O que são expressões regulares;

b) Cite um exemplo de uso das expressões regulares;

c) Cite e explique 3 extensões das linguagens regulares;

Utilize o código disponível na página para as questões de 5 até 8 a seguir.

Questão 5. Altere o analisador léxico da linguagem SimplePascal para informar a linha onde encontrou cada token.

Questão 6. Altere o analisador léxico da linguagem SimplePascal para que reconheça também:

- Comando de atribuição '**:=** '
- Ponto e vírgula: ';'

Questão 7. Altere o analisador léxico da linguagem SimplePascal para que retorne um inteiro assim que encontrar um token (use tokens.h).

Questão 8. O analisador sintático da linguagem SimplePascal permite apenas atribuição do tipo <T_ID> := <T_ID>;. Altere-o para que seja possível atribuições das formas:

- <T_ID> := <T_ID>;
- <T_ID> := <T_INTEGER>;
- <T_ID> := <T_FLOAT>;

Questão 9. Considere a gramática livre de contexto:

$$S \rightarrow S\,S + \mid S\,S * \mid a$$

a) A gramática é ambígua ou não ambígua? Justifique.
b) Descreva a linguagem gerada por essa gramática.

Questão 10. Demonstre que a gramática abaixo é ambígua. Rescreva a gramática eliminando a ambiguidade de forma que o comando **else** seja associado ao comando if mais próximo.

C → if E then C else C | if E then C | s
E → e

Questão 11. Altere o programa "recursivo.c" para que aceite também as operações de subtração e divisão. Sendo que a divisão deve ter precedência sobre a soma ou subtração.

Questão 12. Qual a árvore gerada pelo programa "recursivo.c" para a expressão (2+3)*4?

Questão 13. Construa a tabela sintática LL(1) para a gramática a seguir:

```
S -> [ L ]
   | a
L -> L,S
   | S
```

Questão 14. Mostre o estado da pilha em cada passo de execução do algoritmo LR sobre a cadeia `id + id * id`. Utilize o método SLR e a gramática a seguir:

$$E \rightarrow E + T \mid T$$
$$T \rightarrow T * F \mid F$$
$$F \rightarrow (E) \mid id$$

Questão 15. Explique o problema do conflito empilhar/reduzir (shift/reduce) :

a) dê um exemplo de gramática que cause esse problema;
b) Como a ferramenta Bison trata esse problema?
c) Toda gramática LR(1) apresentará esse problema se tentarmos construir a tabela de análise através do método SLR?

Esse link pode auxiliar:
https://www.gnu.org/software/bison/manual/html_node/Shift_002fReduce.html

Questão 16. Indique Verdadeiro ou Falso

a) [ ] O método LALR é o método mais geral de reconhecimento sintático;
b) [ ] Gramáticas ambíguas não podem ser reconhecidas por métodos LL. Portanto, deve-se utilizar métodos de reconhecimento sintático mais poderosos como o LR(1);
c) [ ] Os métodos descendentes recursivos reconhecem uma entrada através de tentativas de derivações a partir do símbolo inicial da gramática;
d) [ ] São métodos de reconhecimento top-down: Descendente Recursivo, Descendente Preditivo e LL(1);
e) [ ] São métodos de reconhecimento bottom-up: SLR, LR e LALR;
f) [ ] Se a tabela ACTION criada para uma gramática através do método SLR contiver conflitos, então pode-se dizer que a gramática não é uma gramática SLR;
g) [ ] Uma gramática LALR não pode ser reconhecida por um método de reconhecimento descendente recursivo;

Questão 17. Altere o compilador http://leandroip.com/wp-content/uploads/2018/09/comativ3.zip para que, dada uma expressão de entrada, gere a sua correspondente em formato postfix. Ex.: Sendo a entrada "2 + 3 * 5", o compilador deve gerar "2 3 5 * +". Note que para melhor identificar os números, eles devem ser separados por 1 espaço.

Questão 18. Altere o compilador http://leandroip.com/wp-content/uploads/2018/09/comativ3.zip para que permita:

a) a atribuição de uma variável. Ex.: a = 20; b = 30;
b) Permita usar as variáveis declaradas para calcular expressões. Ex.: a + 20*b;

[Ref 1] AHO, Alfred V et al. (). Compiladores: princípios, técnicas e ferramentas. 2. ed. São Paulo: Pearson/Addison Wesley, 2007. 634 p. ISBN 9788588639249 (broch.).