Processamento de Linguagens
Licenciatura em Engenharia de Sistemas Informáticos
2024/2025

Segundo Teste de Avaliação (modelo)

Grupo I: gramáticas

1. Considere a seguinte gramática $G =< T, N, S, P >$, cujo **axioma** é S, e as **regras de produção** em P têm a seguinte definição (notação `BNF`):

$$
\begin{array}{rcll}
S & \rightarrow & \texttt{b}\ T\ \texttt{d} & (1)\\
 & | & \texttt{b}\ \texttt{d}\ S & (2)\\
T & \rightarrow & X\ Y\ \texttt{c} & (3)\\
X & \rightarrow & \varepsilon & (4)\\
 & | & \texttt{a}\ X & (5)\\
Y & \rightarrow & \texttt{b} & (6)\\
 & | & \varepsilon & (7)
\end{array}
$$

   1.1. Indique quais são os símbolos *terminais* e quais os símbolos *não terminais* de G.

   1.2. Considerando agora o reconhecimento *bottom-up* (na mesma gramática), apresente a sequência de ações que o reconhecedor *shift/reduce* executa para reconhecer a seguinte frase: `b a b c d`

STACK	FRASE	AÇÃO
	`b a b c d`	*shift*
`b`	`a b c d`	...
...	...	...

   1.3. Represente os passos da *derivação* obtida a partir da tabela da alínea anterior com a traçagem do algoritmo *shift/reduce*.

   1.4. Desenhe a *árvore de sintaxe* que representa a derivação representada na alínea anterior.

2. Considere a linguagem que reconhece frases constituídas por: um ou mais símbolos `'a'`, seguidos de símbolos `'b'` ou `'c'`, sendo que o número de símbolos `'a'`, mais o número de símbolos de `'b'`, é igual ao número de símbolos de `'c'`.
   Seguem alguns exemplos de *frases válidas*:
   `'a b c c'`, `'a c b c'`, `'a c c b'`, `'a a c c'`, `'a c'`, `'a b b c c c'`, ...

   Assuma que já existe um *analisador léxico* que reconhece os *literais*: `'a'`, `'b'` e `'c'`.

   2.1. Especifique a gramática $G =< T, N, S, P >$ capaz de reconhecer a sintaxe desta linguagem.

   2.2. Calcule a *árvore de sintaxe* da derivação, na gramática G, da frase: `'a a c c'`

grupo II: reconhecimento sintático

1. A seguinte gramática define uma linguagem para a descrição de um programa que contém apenas a definição de constantes (apenas com um valor numérico inteiro) e a declaração de variáveis (do tipo inteiro, ou caracter). O **axioma** da gramática é o símbolo *Prog*. As **regras de produção** da gramática têm a seguinte definição (notação `BNF`):

$$
\begin{array}{rcl}
Prog & \rightarrow & Cabec\ Decls\ Fim \\
Cabec & \rightarrow & \texttt{"PROGRAMA" id texto ';'} \\
Decls & \rightarrow & \varepsilon \\
 & | & Decls\ \ Dcl\ \texttt{';'} \\
Dcl & \rightarrow & \texttt{"CONST" id '=' valor} \\
 & | & \texttt{"VAR" id ':'}\ Tipo \\
Tipo & \rightarrow & \texttt{"INTEIRO"} \\
 & | & \texttt{"CARACTER"} \\
Fim & \rightarrow & \texttt{'.'}
\end{array}
$$

   Os símbolos **terminais** admitidos na linguagem são os seguintes:

   palavras reservadas que estão representadas entre aspas na gramática: `CONST`, `VAR`, `INTEIRO` e `CARACTER`;

   literais caracteres reconhecidos de forma isolada: `';'`, `'='`, `':'` e `'.'`;

   pseudo-terminais são símbolos terminais, cujo valor pode variar de acordo com as seguintes desccrições:

   `valor:` número inteiro sem sinal;

   `id:` sequência começada por uma letra minúscula, seguida de mais letras minúsculas e digitos;

   `texto:` sequência de caracteres alfanuméricos delimitados por aspas.

1.1. Calcule a *árvore de sintaxe* que representa a derivação da frase:

```
1 PROGRAMA soma "um exemplo";
2 CONST a=22;
3 VAR   xyz: INTEIRO;
4 .
```

1.2. Especifique o *analisador léxico* (em *flex*) para a gramática em causa, que permita a passagem dos valores dos símbolos **terminais** para o respectivo reconhecedor sintático.

1.3. Especifique o *analisador sintático* (em *yacc*) para as **frases** geradas pela gramática.

1.4. Acrescente à especificação anterior *acções semânticas*, para **indicar a quantidade** de constantes, variáveis inteiro e variáveis caracter.

1.5. Proponha uma *árvore abstrata* correspondente à árvore de sintaxe apresentada na alínea 1.1.