Universidade Federal do ABC

Técnicas Avançadas de Programação – 2018.Q1

Profs. Daniel M. Martin e Francisco Isidro Massetto

Projeto 3. Verificador de gramáticas

Valor: 10 pontos. Entrega: até 3 de abril.

1 Introdução

Uma das disciplinas mais fundamentais da Ciência da Computação se chama Linguagens Formais e Autômatos. Dentro dessa disciplina, o estudo de gramáticas (das mais variáveis classificações) é algo que demanda atenção e formalismo algébrico.

Podemos definir uma gramática como uma quádrupla ordenada $G = (V, T, P, S)$, onde

- V é o conjunto de símbolos não terminais;
- T é o conjunto de símbolos terminais;
- P é o conjunto de regras de produção;
- S é o símbolo inicial da derivação da gramática.

1.1 Exemplo

$$
\begin{aligned}
V &= \{E, F\} \\
T &= \{a, +, -\} \\
P &= \{ \\
&\qquad E \to E + F \\
&\qquad E \to E - F \\
&\qquad E \to F \\
&\qquad F \to a \\
&\quad \} \\
S &= E
\end{aligned}
$$

Neste exemplo, temos uma gramática livre de contexto que gera expressões aritméticas bem formadas com as operações de soma e subtração e o literal a.

1.2 Convenção

As regras para a definição das gramáticas seguem sempre uma convenção e isso facilita sua compreensão e análise.

Em geral, os símbolos não-terminais são representados por letras maiúsculas, mas eles também podem ser representados por palavras entre < e >. Símbolos terminais são representados por letras

minúsculas, dígitos e símbolos especiais (adição, hífen, vírgula, entre outros). No conjunto das regras de produção, cada regra aparece representada na forma $\alpha \rightarrow \beta$, onde α e β são chamadas de formas sentenciais e podem ser, cada uma, uma combinação de símbolos terminais e não terminais (não pode haver um símbolo que não pertenca a $V \cup T$). A flecha $\rightarrow$ representa justamente a substituição da sequência de símbolos α pela sequência β. O símbolo inicial deve ser um elemento que pertença ao conjunto V e, geralmente, é a primeira variável que aparece na primeira regra.

Com essa convenção, as informações de quais são os símbolos terminais, as variáveis e a variável inicial pode ser depreendida do conjunto de regras. No exemplo da seção anterior, você poderia identificar todos os elementos da gramática apenas olhando para o conjunto de regras.

2 Tarefa

Você deverá fazer um programa que, dado um texto contendo uma definição de gramática, avalia se esse texto efetivamente define uma gramática. Ou seja, você deverá construir um parser para uma gramática que descreve gramáticas, que informa ao usuário se a entrada corresponder efetivamente à uma definição de gramática, ou se há algum erro. No caso do texto de entrada representar uma gramática corretamente, seu programa deverá classificá-la de acordo com a hierarquia de Chomsky.

2.1 Parte I: reconhecimento

Seu programa deve verificar se o texto de entrada pode ser gerado pela gramática livre de contexto explicada abaixo.

Regras

As regras da gramática são:

```
<gramatica>  -->  <regra>  <gramatica>
<gramatica>  -->  <regra>  EOF
<regra>      -->  <sentenca1>  FLECHA  <sentenca2>  EOL
<sentenca1>  -->  <elemento>  <sentenca2>
<sentenca2>  -->  <elemento>  <sentenca2>
<sentenca2>  -->
<elemento>   -->  <variavel>
<elemento>   -->  TERMINAL
<variavel>   -->  LETRAMAIUSCULA
<variavel>   -->  SINALMENOR  <id>  SINALMAIOR
<id>         -->  TERMINAL  <id>
<id>         -->  TERMINAL
```

Note que é possível ter a palavra vazia do lado direito da flecha em qualquer regra. Isso foi formalizado acima na sexta regra de cima para baixo.

Terminais (*Tokens*)

Os terminais dessa gramática são

- `EOL`, que corresponde a um caractere `'\n'` na entrada;
- `FLECHA`, que corresponde a um subtexto `"-->"` na entrada;

- `LETRAMAIUSCULA`, que corresponde a um caractere no intervalo `[A-Z]`;
- `SINALMENOR`, que corresponde a um caractere `'<'` na entrada;
- `SINALMAIOR`, que corresponde a um caractere `'>'` na entrada, exceto se:
  - for precedido de `"--"`, daí ele faz parte de uma `FLECHA`;
- `TERMINAL`, que corresponde a qualquer caractere, exceto se:
  - for um sinal de `'-'` seguido de `"->"`, daí ele deve fazer parte de uma `FLECHA`,
  - for o caractere `'<'`, que deve ser considerado `SINALMENOR`,
  - for o caractere `'>'`, que deve ser considerado `SINALMAIOR`,
  - qualquer caractere no intervalo `[A-Z]`, esses devem ser considerados `LETRAMAIUSCULA`;
- `EOF`, que corresponde ao fim da entrada.

Espaços em branco (`' '` ou `'\t'`) devem ser ignorados e eles jamais serão considerados terminais.

2.2 Parte II: classificação

De acordo com Noam Chmomsky, as gramáticas podem ser classificadas em quatro tipos:

- gramáticas regulares – tipo 3;
- gramáticas livres de contexto – tipo 2;
- gramáticas sensíveis ao contexto – tipo 1;
- gramáticas irrestritas – tipo 0.

Gramáticas regulares

São gramáticas em que todas as regras de produção seguem as formas

$$A \to aB$$
$$A \to \varepsilon$$

ou todas as regras seguem as formas

$$A \to Ba$$
$$A \to \varepsilon$$

onde a é um terminal (pertence a T) e A, B são variáveis (pertencem a V).

Gramáticas livres de contexto

Em resumo, suas regras de produção são todas da forma

$$A \to w$$

onde $A \in V$ e w pertence a $(TUV)^*$.

Gramáticas sensíveis ao contexto

São gramáticas cujas regras de produção seguem a forma:

$$uAw \rightarrow uvw$$

onde $A \in V$, $u, w \in (V \cup T)^*$ e $v \in (V \cup T)^+$.

Gramáticas irrestritas

Em resumo, suas regras de produção seguem a forma:

$$u \rightarrow w$$

onde $u \in (V \cup T)^+$ e $w \in (V \cup T)^*$.

2.3 Entrada e saída

A entrada do programa pode ser qualquer texto (com multiplas linhas, inclusive).

A saída do seu programa deverá ter uma única linha de texto na saída terminada por `'\n'`. Caso a gramática da Seção 2 seja capaz de gerar o texto da entrada, seu programa deve imprimir a palavra `CORRETO`, seguida de um espaço em branco, seguido do número do tipo da gramática segundo a classificação de Noam Chomsky. Se você não implementou o classificador, imprima somente a palavra `CORRETO`. Por outro lado, se o texto da entrada não representa uma gramática, seu programa deve imprimir a palavra `ERRO`.

Exemplo

Entrada:

```
G --> RG
G --> R
R --> aFS;
F --> :=
S --> aS
S --> TS
T --> 1
```

Saída:

```
CORRETO 2
```

3 Pontuação

- Implementação correta do reconhecedor de gramáticas: 8 pontos.
- Implementação correta do classificador: 4 pontos.

O script de correção será divulgado em breve.