Linguagens formais e autômatos

Análise sintática

Gabriel V C Candido
gabriel.candido@ifpr.edu.br

Instituto Federal do Paraná - Pinhais

INSTITUTO FEDERAL
Paraná
Campus Pinhais

Sumário

Sumário

Análise sintática

Dada uma gramática G e uma palavra $w \in \Sigma^*$,
como saber se $w \in L(G)$?

Podemos produzir w com a gramática G? $S \Rightarrow^* w$?

Sumário

Derivações à esquerda

Vimos que podem haver várias derivações para uma mesma palavra

Para resolver o problema, qualquer derivação basta. Mas seria interessante fazer isso de forma algorítmica, e reduzir o espaço de busca.

Derivações à esquerda

A nossa definição de gramática diz que $w \in L(G)$ se, somente se, existe uma derivação de w a partir de S em G.

Derivações à esquerda

Uma palavra w está na $L(G)$ se, somente se, existe uma derivação à esquerda que sai de S e chega em w.

Derivações à esquerda

Uma palavra w está na $L(G)$ se, somente se, existe uma derivação à esquerda que sai de S e chega em w.

A parte $\Leftarrow$ é trivial, pois se existe uma derivação mais à esquerda, então existe uma derivação, então w está na linguagem.

Derivações à esquerda

Uma palavra w está na $L(G)$ se, somente se, existe uma derivação à esquerda que sai de S e chega em w.

Para $\Rightarrow$: w está na linguagem é a nossa hipótese. Então existe uma derivação.

Derivações à esquerda

Uma palavra w está na $L(G)$ se, somente se, existe uma derivação à esquerda que sai de S e chega em w.

Para $\Rightarrow$: w está na linguagem é a nossa hipótese. Então existe uma derivação.

Nosso desafio é mostrar que se existe uma derivação, então existe uma derivação á esquerda.

Derivações à esquerda

Uma palavra w está na $L(G)$ se, somente se, existe uma derivação à esquerda que sai de S e chega em w.

Para $\Rightarrow$:

- $S \Rightarrow w_1 \Rightarrow w_2 \Rightarrow \cdots \Rightarrow w_{n-1} \Rightarrow w_n = w$
- Em algum momento, $w_k \Rightarrow w_{k+1}$ não é mais à esquerda, então $w_k = u_1 A u_2 B u_3$
- Como não foi mais à esquerda, expandimos B: $u_1 A u_2 v u_3$
- Mais tarde, obrigatoriamente, tivemos que aplicar a regra A: $u_1 p u_2 v u_3 \Rightarrow^* w$.

Derivações à esquerda

continuação

- Mas então, podemos expandir primeiro A: $w_k \Rightarrow u_1 p u_2 B u_3$.
- Mais tarde expandiremos B: $u_1 p u_2 v u_3 \Rightarrow^* w$

Podemos repetir o raciocínio para $k+2, k+3, \ldots, n$

Sumário

Ambiguidade

Podem existir mais de uma derivação para uma mesma palavra $w \implies$ *ambiguidade*!

João ganhou o livro do Drummond.

Isso é importante para nós pois linguagens de programação não podem admitir ambiguidades: por isso são tão rígidas; por isso não programamos em língua natural.

Ambiguidade

Definição

Uma gramática livre de contexto G é ambígua se existe uma palavra $w \in L(G)$ que pode ser derivada por duas derivações distintas.

Exemplo 1

- $G : S \rightarrow aS|Sa|a$

G é ambígua pois existem duas derivações mais à esquerda para aa:

- $S \Rightarrow aS \Rightarrow aa$
- $S \Rightarrow Sa \Rightarrow aa$

Note que $L(G) = a^+$ que pode ser gerada pela seguinte gramática, sem ambiguidade:

- $S \rightarrow aS|a$

Ambiguidade

Ambiguidade é uma propriedade de gramáticas e não de linguagens!

Ainda assim, existem linguagens que são inerentemente ambíguas: não há gramáticas sem ambiguidade.

Exemplo 2

- $G : S \rightarrow bS|Sb|a$
- Note que $L(G) = b^*ab^*$
- Para bab:
  - $S \Rightarrow bS \Rightarrow bSb \Rightarrow bab$
  - $S \Rightarrow Sb \Rightarrow bSb \Rightarrow bab$

G_1:

- $S \rightarrow bS|aA$
- $A \rightarrow bA|\lambda$

G_2:

- $S \rightarrow bS|A$
- $A \rightarrow Ab|a$

$L(G) = L(G_1) = L(G_2)$, e G_1 e G_2 não são ambíguas!

Exemplo 3

- $G : S \rightarrow aSb|aSbb|\lambda$
- Note que $L(G) = \{a^n b^m, 0 \leq n \leq m \leq 2n\}$
- Para *aabbb*:
  - $S \Rightarrow aSb \Rightarrow aaSbbb \Rightarrow aabbb$
  - $S \Rightarrow aSbb \Rightarrow aaSbbb \Rightarrow aabbb$
- Então G é ambígua. Como fazer uma gramática não ambígua?

Para essa linguagem, podemos pensar em primeiro gerar todos os *a*s que casam com um *b* e só depois gerar os *a* que casam com dois *b*s:

- $S \rightarrow aSb|A|\lambda$
- $A \rightarrow aAbb|abb$

Sumário

Grafo de uma gramática

Podemos construir um grafo onde os nós são as formas sentenciais e as arestas ligam nós que podem ser gerados a partir de outro.

Podemos considerar sempre as derivações mais à esquerda

Um caminho nesse grafo representa a derivação de uma palavra. Para resolver o problema de análise sintática, podemos recorrer a algoritmos de grafos: encontrar um caminho no grafo.

Grafo de uma gramática

Grafo localmente finito

O grafo pode ser infinito (a AD pode ser infinita), mas como existe um número finito de regras, o número de filhos é finito.

Se o grafo tem ciclos, a gramática é ambígua.

Estratégias de busca em grafos

Algoritmos *top-down*: partem do símbolo de partida e buscam por w.

Algoritmos *bottom-up*: partem da palavra w e buscam por S.

As buscas podem ser em largura ou em profundidade.