ITC: Introdução à Teoria da Computação

Marcos Castilho

DInf/UFPR

9 de junho de 2021

Análise sintática: introdução

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

Isto é, como saber se $S \overset{*}{\underset{G}{\Rightarrow}} w$?

Derivações à esquerda e ambiguidade

$w \in L(G)$ se $S \overset{*}{\underset{G}{\Rightarrow}} w$;

Sabemos que podem haver várias derivações distintas para uma mesma palavra;

Teoricamente, qualquer uma delas basta;

Mas, para o computador, deve haver uma maneira de reduzir o espaço de busca;

Para isto, o teorema seguinte é importante.

Teorema

Seja $G = (V, \Sigma, P, S)$ uma gramática livre de contexto. Uma palavra $w \in L(G)$ se, e somente se, existe uma derivação mais à esquerda de w a partir de S.

Prova

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

A outra parte é mais complicada, conforme veremos a seguir.

Prova, continuação

Seja $S \Rightarrow w_1 \Rightarrow w_2 \Rightarrow \ldots \Rightarrow w_{n-1} \Rightarrow w_n = w$ uma derivação qualquer de w, não necessariamente mais à esquerda. Vamos mostrar como obter a partir desta uma que seja mais à esquerda.

Seja w_k a primeira forma sentencial nesta derivação para a qual a regra aplicada não tenha sido mais à esquerda

Se não há este w_k então não há nada mais a provar

Logo, $k < n$. Vamos mostrar como obter a uma nova derivação de w, de tamanho n, de tal maneira que a primeira regra que não foi aplicada mais à esquerda ocorra após k.

Seguinte esta ideia no máximo $n - k$ vezes obteremos uma derivação mais à esquerda de w.

Prova, continuação

Assim, temos:

$S \Rightarrow w_1 \ldots \Rightarrow w_k \Rightarrow w_{k+1} \Rightarrow \ldots \Rightarrow w_n = w$

Por construção, $S \stackrel{*}{\Rightarrow} w_k$ é uma derivação mais à esquerda

Também por construção, $w_k = u_1 A u_2 B u_3$, onde:

- $u_1 \in \Sigma^*$
- w_{k+1} foi obtido por uma regra do tipo $B \rightarrow v$

Portanto:

$S \stackrel{*}{\Rightarrow} w_k = u_1 A u_2 B u_3 \Rightarrow u_1 A u_2 v u_3 = w_{k+1} \stackrel{*}{\Rightarrow} w_n = w$

Prova, continuação

Mas $w \in \Sigma^*$, portanto em algum momento entre $k+1$ e $n-1$ uma regra do tipo $A \to p$ foi aplicada

Seja este momento representado por $j+1$, assim:

$w_j = u_1 A q \Rightarrow u_1 p q = w_{j+1}$, na derivação original!

De tal maneira que as regras aplicadas nos passos $k+2$ até j transformaram a palavra $u_2 v u_3$ em q

Finalmente, na derivação original certamente ocorreu que $w_{j+1} \stackrel{*}{\Rightarrow} w_n = w$.

Agora, com tudo isto detalhado, uma derivação mais a esquerda será construída.

Prova, continuação

$S \overset{k}{\Rightarrow} w_k = u_1 A u_2 B u_3$, que por construção é mais à esquerda

$\Rightarrow u_1 p u_2 B u_3$, aplicando $A \to p$

$\Rightarrow u_1 p u_2 v u_3$, aplicando $B \to v$

$\overset{j-k-1}{\Longrightarrow} u_1 p q = w_{j+1}$, pois $u_2 v u_3 \overset{*}{\Rightarrow} q$

$\overset{n-j-1}{\Longrightarrow} w_n = w$, pois $w_{j+1} \overset{*}{\Rightarrow} w_n$

Esta derivação é mais à esquerda.

Observações

O teorema só se aplica para palavras $w \in \Sigma^*$

Pode-se mostrar um teorema equivalente para derivações mais à direita

$v \underset{L}{\overset{*}{\Rightarrow}} w$ e $v \underset{R}{\overset{w}{\Rightarrow}}$ são, respectivamente, as derivações mais à esquerda e mais à direita de w.

Observação

Não é necessariamente verdade que existe uma única derivação mais à esquerda para uma palavra w.

A consequência deste fato é a noção de *ambiguidade*

Exemplo de frase ambígua:

João ganhou o livro do Drummond

"do Drummond" pode modificar o verbo (objeto indireto); ou pode modificar o nome (adjunto adnominal)

Em resumo

Uma mesma árvore de derivação pode ter sido obtida por duas (ou mais) derivações mais à esquerda

Como um compilador baseia-se na árvore de derivação para gerar código, é preciso definir gramáticas para linguagens de programação que não sejam ambíguas.

Definição

Uma Gramática Livre de Contexto G é *ambígua* se existe uma palavra $w \in L(G)$ que pode ser derivada por duas derivações mais à esquerda distintas.

Se G não é ambígua, dizemos que ela é *não ambígua*.

Exemplo

Seja G a gramática dada pelas regras:

$$S \to aS \mid Sa \mid a$$

G é ambígua, pois as duas derivações abaixo são mais à esquerda para a palavra aa:

$$S \Rightarrow aS \Rightarrow aa$$

$$S \Rightarrow Sa \Rightarrow aa$$

Notar que $L(G) = a^+$, que pode ser gerada pela gramática regular:

$$S \to aS \mid a$$

Observações

Portanto, ambiguidade é uma propriedade de gramáticas e não de linguagens!

No entanto, existem linguagens que são *inerentemente ambíguas*, para as quais não é possível definir uma gramática não ambígua.

Exemplo 1

Seja G a gramática:

$$S \to bS \mid Sb \mid a$$

Evidentemente, $L(G) = b^*ab^*$. Existem duas derivações para bab:

$$S \Rightarrow bS \Rightarrow bSb \Rightarrow bab$$

$$S \Rightarrow Sb \Rightarrow bSb \Rightarrow bab$$

Mas as gramáticas G_1 e G_2 abaixo são tais que $L(G) = L(G_1) = L(G_2)$ e são não ambíguas.

$$G_1 : \begin{cases} S \to bS \mid aA \\ A \to bA \mid \lambda \end{cases} \qquad G_2 : \begin{cases} S \to bS \mid A \\ A \to Ab \mid a \end{cases}$$

Exemplo 2

Seja G a gramática:

$$S \to aSb \mid aSbb \mid \lambda$$

$L(G) = \{a^n b^m \mid 0 \leq n \leq m \leq 2n\}$. G é ambígua, pois:

$S \Rightarrow aSb \Rightarrow aaSbbb \Rightarrow aabbb$

$S \Rightarrow aSbb \Rightarrow aaSbbb \Rightarrow aabbb$

A estratégia para definir uma gramática G_1 que não seja ambígua é primeiro gerar os a's que casam com um b e somente depois gerar os a's que casam com dois b's:

$$G_1 : \begin{cases} S \to aSb \mid A \mid \lambda \\ A \to aAbb \mid abb \end{cases}$$

O grafo de uma gramática

Definiremos um grafo onde os nodos são formas sentenciais obtidas por derivações mais à esquerda (forma sentencial à esquerda), e as arestas são definidas assim:

Se $v \underset{L}{\Rightarrow} w$ em G então w é adjacente a v no grafo.

Definição

Seja $G = (V, \Sigma, P, S)$ uma gramática livre de contexto. O *grafo mais à esquerda da gramática* G, denotado $g(G)$, é o grafo dirigido rotulado (N, P, A) assim definido:

$$N = \{w \in (V \cup \Sigma)^* \mid S \underset{L}{\overset{*}{\Rightarrow}} w\}$$

$$A = \{[v, w, r] \in N \times N \times P \mid v \underset{L}{\Rightarrow} w \text{ por uma aplicação da regra } r\}.$$

Observações

Um caminho de S até w em $g(G)$ representa uma derivação de w a partir de S em G

O rótulo do arco de v a w especifica a regra que foi aplicada em v para obter-se w

Assim, para sabermos se $w \in L(G)$, basta resolvermos o problema de encontrar um caminho de S até w em $g(G)$.

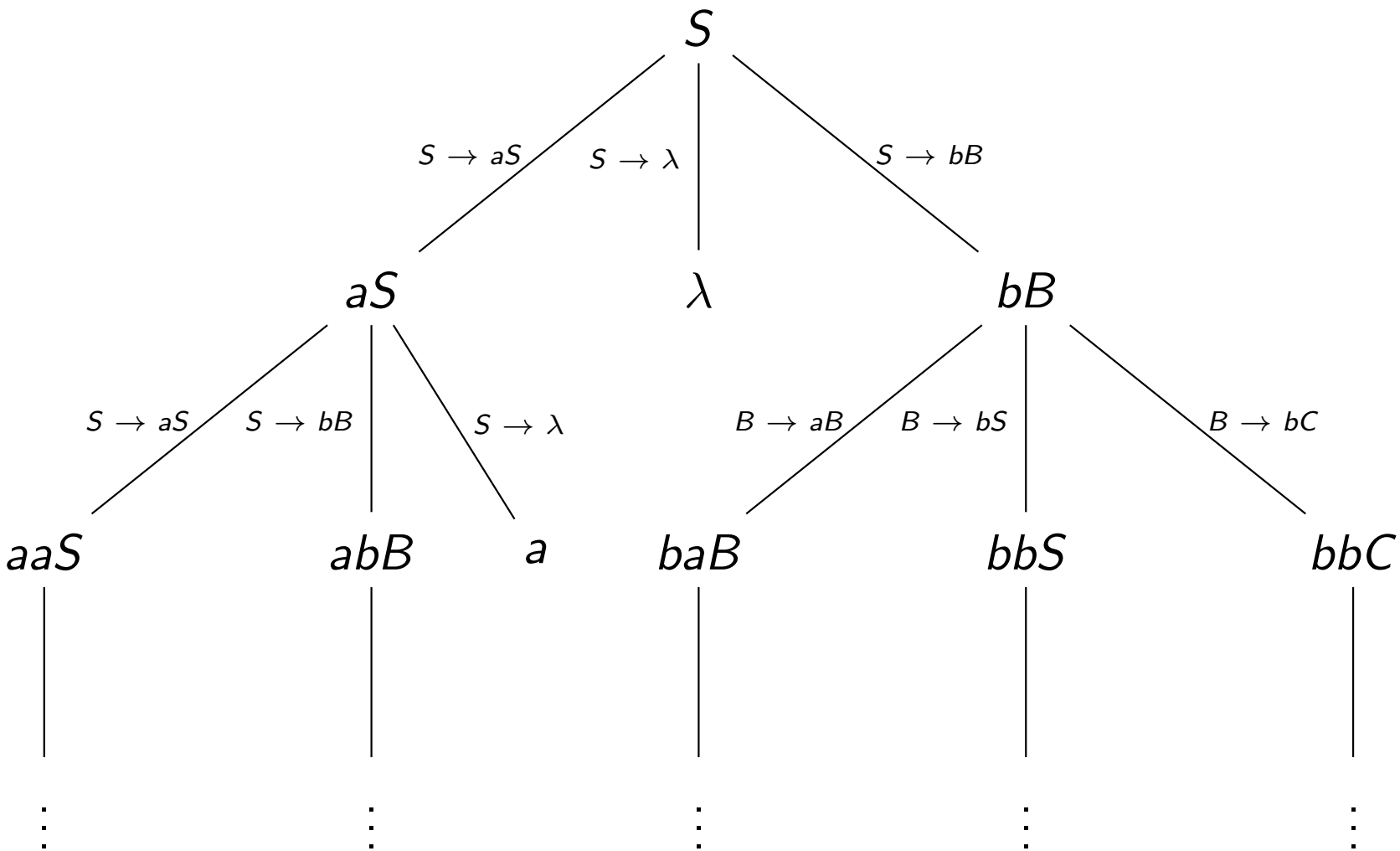
Isto pode ser feito recorrendo-se à Teoria do Grafos!

Definição

Um grafo em que cada nodo tem um número finito de filhos é dito *localmente finito*, ainda que tenha um número infinito de nodos.

É o caso de várias gramáticas livre de contexto interessantes.

Exemplo



Observações

Se toda palavra no grafo tem apenas uma derivação mais à esquerda, o grafo é uma árvore;

Se tem ciclos, a gramática é ambígua.

Observações

Técnicas padrão de busca em grafos são empregadas, mas o grafo não precisa ser totalmente definido, pois é infinito;

Usa-se a noção de *grafo implícito*, que é obtido passo a passo, enquanto os caminhos são examinados;

Assim, o mínimo possível do grafo é construído.

Observações

Pode-se definir algoritmos que iniciam com S e buscam por w, ou seja, a busca é *top-down*;

Também pode-se iniciar com w em busca de S, ou seja, a busca é *bottom-up*;

Ainda é possível realizar buscas em amplitude (*breadth-first*) ou em profundidade (*depth-first*).

Licença

Slides feitos em LaTeX usando beamer e tikz, editados com vim.

Licença