Um compilador simples de uma passagem

Visão geral

Prof. Edson Alves

Faculdade UnB Gama

Sumário

Análise gramatical

- A análise gramatical é o processo de se determinar se uma cadeia de tokens pode ser gerada por uma gramática

Análise gramatical

- A análise gramatical é o processo de se determinar se uma cadeia de tokens pode ser gerada por uma gramática
- O compilador deve ser capaz de construir uma árvore gramatical, mesmo que de forma implícita

Análise gramatical

- A análise gramatical é o processo de se determinar se uma cadeia de tokens pode ser gerada por uma gramática
- O compilador deve ser capaz de construir uma árvore gramatical, mesmo que de forma implícita
- Um analisador gramatical pode ser construído para qualquer gramática

Análise gramatical

- A análise gramatical é o processo de se determinar se uma cadeia de tokens pode ser gerada por uma gramática
- O compilador deve ser capaz de construir uma árvore gramatical, mesmo que de forma implícita
- Um analisador gramatical pode ser construído para qualquer gramática
- Para qualquer gramáticas livres de contexto existe um analisador gramatical que analisa N tokens com complexidade $O(N^3)$

Análise gramatical

- A análise gramatical é o processo de se determinar se uma cadeia de tokens pode ser gerada por uma gramática
- O compilador deve ser capaz de construir uma árvore gramatical, mesmo que de forma implícita
- Um analisador gramatical pode ser construído para qualquer gramática
- Para qualquer gramáticas livres de contexto existe um analisador gramatical que analisa N tokens com complexidade $O(N^3)$
- Contudo, existem analisadores lineares para quase todas as gramáticas livres de contexto que surgem na prática

Análise *top-down* e *bottom-up*

- Há duas classes principais de analisadores gramaticais

Análise *top-down* e *bottom-up*

- Há duas classes principais de analisadores gramaticais
- Analisadores *top-down* a construção parte da raiz da árvore gramatical para suas folhas

Análise *top-down* e *bottom-up*

- Há duas classes principais de analisadores gramaticais
- Analisadores *top-down* a construção parte da raiz da árvore gramatical para suas folhas
- Analisadores *bottom-up* partem das folhas em direção à raiz

Análise *top-down* e *bottom-up*

- Há duas classes principais de analisadores gramaticais
- Analisadores *top-down* a construção parte da raiz da árvore gramatical para suas folhas
- Analisadores *bottom-up* partem das folhas em direção à raiz
- Os analisadores *top-down* são mais populares, pois é possível construir analisadores eficientes desta classe de forma manual

Análise *top-down* e *bottom-up*

- Há duas classes principais de analisadores gramaticais
- Analisadores *top-down* a construção parte da raiz da árvore gramatical para suas folhas
- Analisadores *bottom-up* partem das folhas em direção à raiz
- Os analisadores *top-down* são mais populares, pois é possível construir analisadores eficientes desta classe de forma manual
- Já os analisadores *bottom-up* podem manipular uma gama mais ampla de gramáticas

Análise *top-down* e *bottom-up*

- Há duas classes principais de analisadores gramaticais
- Analisadores *top-down* a construção parte da raiz da árvore gramatical para suas folhas
- Analisadores *bottom-up* partem das folhas em direção à raiz
- Os analisadores *top-down* são mais populares, pois é possível construir analisadores eficientes desta classe de forma manual
- Já os analisadores *bottom-up* podem manipular uma gama mais ampla de gramáticas
- Geradores de analisadores gramaticais tendem a usar métodos *bottom-up*

Construção *top-down* de uma árvore gramatical

1. Inicie na raiz, rotulada pelo não-terminal de partida

Construção *top-down* de uma árvore gramatical

1. Inicie na raiz, rotulada pelo não-terminal de partida
2. Repita os seguintes passos:

Construção *top-down* de uma árvore gramatical

1. Inicie na raiz, rotulada pelo não-terminal de partida
2. Repita os seguintes passos:
   (a) Para o nó n, rotulado pelo não-terminal A, selecione uma das produções para A e construa os filhos de n com os símbolos do lado direito da produção

Construção *top-down* de uma árvore gramatical

1. Inicie na raiz, rotulada pelo não-terminal de partida
2. Repita os seguintes passos:
   (a) Para o nó n, rotulado pelo não-terminal A, selecione uma das produções para A e construa os filhos de n com os símbolos do lado direito da produção
   (b) Encontre o próximo nó no qual uma subárvore deve ser construída

Construção *top-down* de uma árvore gramatical

1. Inicie na raiz, rotulada pelo não-terminal de partida
2. Repita os seguintes passos:
   (a) Para o nó n, rotulado pelo não-terminal A, selecione uma das produções para A e construa os filhos de n com os símbolos do lado direito da produção
   (b) Encontre o próximo nó no qual uma subárvore deve ser construída

Observações:

Construção *top-down* de uma árvore gramatical

1. Inicie na raiz, rotulada pelo não-terminal de partida
2. Repita os seguintes passos:
   (a) Para o nó n, rotulado pelo não-terminal A, selecione uma das produções para A e construa os filhos de n com os símbolos do lado direito da produção
   (b) Encontre o próximo nó no qual uma subárvore deve ser construída

Observações:

(i) A depender da gramática, esta construção pode ser implementada com uma única passagem da entrada, da esquerda para a direita

Construção *top-down* de uma árvore gramatical

1. Inicie na raiz, rotulada pelo não-terminal de partida
2. Repita os seguintes passos:
   (a) Para o nó n, rotulado pelo não-terminal A, selecione uma das produções para A e construa os filhos de n com os símbolos do lado direito da produção
   (b) Encontre o próximo nó no qual uma subárvore deve ser construída

Observações:

(i) A depender da gramática, esta construção pode ser implementada com uma única passagem da entrada, da esquerda para a direita

(ii) O token que está sendo observado é frequentemente denominado *lookahead*

Construção *top-down* de uma árvore gramatical

1. Inicie na raiz, rotulada pelo não-terminal de partida
2. Repita os seguintes passos:
   - **(a)** Para o nó n, rotulado pelo não-terminal A, selecione uma das produções para A e construa os filhos de n com os símbolos do lado direito da produção
   - **(b)** Encontre o próximo nó no qual uma subárvore deve ser construída

Observações:

- **(i)** A depender da gramática, esta construção pode ser implementada com uma única passagem da entrada, da esquerda para a direita
- **(ii)** O token que está sendo observado é frequentemente denominado *lookahead*
- **(iii)** Inicialmente *lookahead* é o token mais à esquerda da entrada

Exemplo: gramática para geração de subtipos em Pascal

$$
\begin{array}{rcl}
\mathit{tipo} & \rightarrow & \mathit{primitivo} \\
 & | & \uparrow \textbf{id} \\
 & | & \textbf{array}\ [\ \mathit{primitivo}\ ]\ \textbf{of}\ \mathit{tipo} \\
 & & \\
\mathit{primitivo} & \rightarrow & \textbf{integer} \\
 & | & \textbf{char} \\
 & | & \textbf{num}\ ..\ \textbf{num}
\end{array}
$$

Exemplo: gramática para geração de subtipos em Pascal

$$
\begin{array}{rcl}
\mathit{tipo} & \rightarrow & \mathit{primitivo} \\
& | & \uparrow \textbf{id} \\
& | & \textbf{array}\ [\ \mathit{primitivo}\ ]\ \textbf{of}\ \mathit{tipo} \\
 & & \\
\mathit{primitivo} & \rightarrow & \textbf{integer} \\
& | & \textbf{char} \\
& | & \textbf{num}\ ..\ \textbf{num}
\end{array}
$$

Observação: os dois pontos ('..') formam um único token.

Exemplo de construção *top-down* da árvore gramatical

Considere a expressão `array [ num .. num ] of integer`, gerada a partir da gramática de subtipos em Pascal.

Exemplo de construção *top-down* da árvore gramatical

Considere a expressão `array [ num .. num ] of integer`, gerada a partir da gramática de subtipos em Pascal.

(a) A construção inicial na raiz da árvore. O rótulo da raiz é o não-terminal de partida

Exemplo de construção *top-down* da árvore gramatical

Considere a expressão `array [ num .. num ] of integer`, gerada a partir da gramática de subtipos em Pascal.

(a) A construção inicial na raiz da árvore. O rótulo da raiz é o não-terminal de partida

$$tipo$$

Exemplo de construção *top-down* da árvore gramatical

Considere a expressão `array [ num .. num ] of integer`, gerada a partir da gramática de subtipos em Pascal.

(a) A construção inicial na raiz da árvore. O rótulo da raiz é o não-terminal de partida

$$tipo$$

(b) A única produção de $tipo$ que inicia com o *lookahead* (neste momento, `array`) é a terceira. Esta produção será usada para a criação dos filhos do nó raiz.

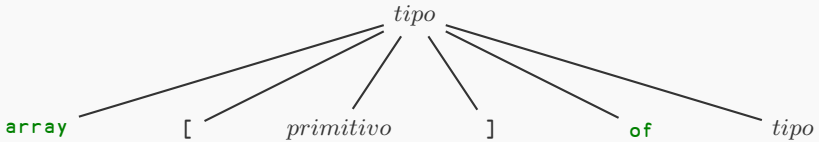
Exemplo de construção *top-down* da árvore gramatical

Considere a expressão `array [ num .. num ] of integer`, gerada a partir da gramática de subtipos em Pascal.

(a) A construção inicial na raiz da árvore. O rótulo da raiz é o não-terminal de partida

$$tipo$$

(b) A única produção de $tipo$ que inicia com o *lookahead* (neste momento, `array`) é a terceira. Esta produção será usada para a criação dos filhos do nó raiz.



Exemplo de construção *top-down* da árvore gramatical

(c) O filho mais à esquerda tem como rótulo `array`. Como este rótulo coincide com *lookahead*, a construção prossegue para o próximo filho

Exemplo de construção *top-down* da árvore gramatical

(c) O filho mais à esquerda tem como rótulo `array`. Como este rótulo coincide com *lookahead*, a construção prossegue para o próximo filho

(d) *Lookahead* é atualizado para [e confrontado com o segundo filho à esquerda da raiz. Como há nova coincidência entre o rótulo e *lookahead*, a construção prossegue

Exemplo de construção *top-down* da árvore gramatical

(c) O filho mais à esquerda tem como rótulo `array`. Como este rótulo coincide com *lookahead*, a construção prossegue para o próximo filho

(d) *Lookahead* é atualizado para [e confrontado com o segundo filho à esquerda da raiz. Como há nova coincidência entre o rótulo e *lookahead*, a construção prossegue

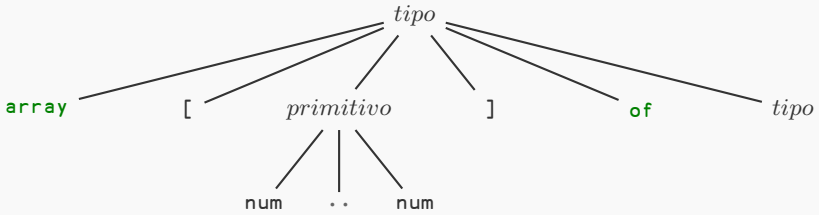
(e) O nó seguinte contém o não-terminal $primitivo$ e *lookahead* contém o token `num`. Assim a terceira produção de $primitivo$ é utilizada para gerar os novos filhos

Exemplo de construção *top-down* da árvore gramatical

(c) O filho mais à esquerda tem como rótulo `array`. Como este rótulo coincide com *lookahead*, a construção prossegue para o próximo filho

(d) *Lookahead* é atualizado para [e confrontado com o segundo filho à esquerda da raiz. Como há nova coincidência entre o rótulo e *lookahead*, a construção prossegue

(e) O nó seguinte contém o não-terminal $primitivo$ e *lookahead* contém o token `num`. Assim a terceira produção de $primitivo$ é utilizada para gerar os novos filhos



Exemplo de construção *top-down* da árvore gramatical

(g) Os próximos tokens (`:`, `num`, `of`) coincidem com os respectivos filhos

Exemplo de construção *top-down* da árvore gramatical

(g) Os próximos tokens (`:`, `num`, `of`) coincidem com os respectivos filhos

(h) O último valor que *lookahead* assum é `integer`, o qual é confrontado com o filho mais à direita da raiz. Como o nó tem como rótulo o não-terminal $tipo$, a primeira produção deste deve ser usada para construir o novo nó, que por sua vez usa a primeira produção de $primitivo$ para construir seu único filho

Exemplo de construção *top-down* da árvore gramatical

(g) Os próximos tokens (`:`, `num`, `of`) coincidem com os respectivos filhos

(h) O último valor que *lookahead* assum é `integer`, o qual é confrontado com o filho mais à direita da raiz. Como o nó tem como rótulo o não-terminal $tipo$, a primeira produção deste deve ser usada para construir o novo nó, que por sua vez usa a primeira produção de $primitivo$ para construir seu único filho