

Um compilador simples de uma passagem

Visão geral

Prof. Edson Alves

Faculdade UnB Gama

Sumário

1. Análise gramatical

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

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

Exemplo: gramática para geração de subtipos em Pascal

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

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.

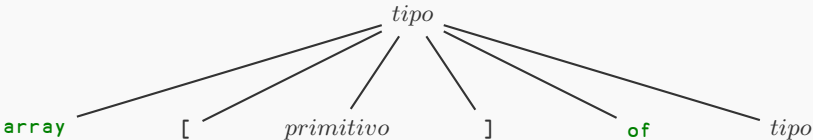
Exemplo de construção *top-down* da árvore gramatical

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

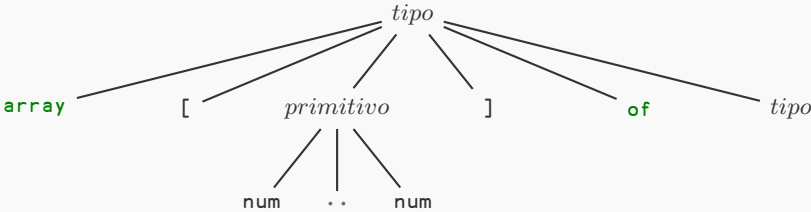
- (c) O filho mais à esquerda tem como rótulo **array**. Como este rótulo coincide com *lookahead*, a construção prossegue para o próximo filho
- (d) *Lookahead* é atualizado para `[` e confrontado com o segundo filho à esquerda da raiz. Como há nova coincidência entre o rótulo e *lookahead*, a construção 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

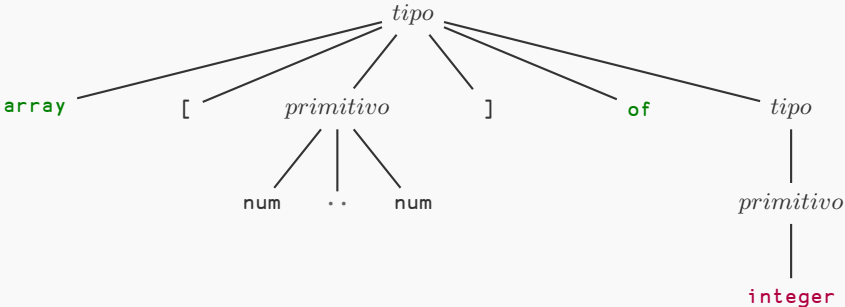
(g) Os próximos tokens (: , num, of) coincidem com os respectivos filhos

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



Análise gramatical preditiva

- ▶ Uma análise gramatical descendente recursiva é um método *top-down* de análise sintática na qual são executados procedimentos recursivos para processar a entrada

Análise gramatical preditiva

- ▶ Uma análise gramatical descendente recursiva é um método *top-down* de análise sintática na qual são executados procedimentos recursivos para processar a entrada
- ▶ Cada não-terminal da entrada é associado a um procedimento

Análise gramatical preditiva

- ▶ Uma análise gramatical descendente recursiva é um método *top-down* de análise sintática na qual são executados procedimentos recursivos para processar a entrada
- ▶ Cada não-terminal da entrada é associado a um procedimento
- ▶ Se *lookahead* determina, sem ambiguidades, o procedimento a ser executado, a análise gramatical descendente recursiva é denominada análise gramatical preditiva

Análise gramatical preditiva

- ▶ Uma análise gramatical descendente recursiva é um método *top-down* de análise sintática na qual são executados procedimentos recursivos para processar a entrada
- ▶ Cada não-terminal da entrada é associado a um procedimento
- ▶ Se *lookahead* determina, sem ambiguidades, o procedimento a ser executado, a análise gramatical descendente recursiva é denominada análise gramatical preditiva
- ▶ A sequência de chamadas de procedimentos no processamento da entrada determina, de forma implícita, a árvore gramatical no processamento da entrada determina, de forma implícita, a árvore gramatical

Análise gramatical preditiva

- ▶ Uma análise gramatical descendente recursiva é um método *top-down* de análise sintática na qual são executados procedimentos recursivos para processar a entrada
- ▶ Cada não-terminal da entrada é associado a um procedimento
- ▶ Se *lookahead* determina, sem ambiguidades, o procedimento a ser executado, a análise gramatical descendente recursiva é denominada análise gramatical preditiva
- ▶ A sequência de chamadas de procedimentos no processamento da entrada determina, de forma implícita, a árvore gramatical no processamento da entrada determina, de forma implícita, a árvore gramatical
- ▶ Além dos procedimentos associados aos não-terminais, a análise pode definir outros procedimentos auxiliares que podem simplificar tarefas como a leitura de tokens e a atualização de *lookahead*

Reconhecimento de tokens

O procedimento `RECONHECER()` confronta o valor de *lookahead* e um determinado token. Em caso de coincidência, ele atualiza *lookahead* com o próximo token da entrada.

Reconhecimento de tokens

O procedimento RECONHECER() confronta o valor de *lookahead* e um determinado token. Em caso de coincidência, ele atualiza *lookahead* com o próximo token da entrada.

1: **procedure** RECONHECER(*token*)

2: **if** *lookahead* = *token* **then**

3: *lookahead* \leftarrow PROXIMOTOKEN()

4: **else**

5: ERRO()

▷ *lookahead* é uma variável global

Procedimento associado ao não terminal *tipo*

```
1: procedure TIPO(  
2:   if lookahead  $\in$  { integer, char, num } then  
3:     PRIMITIVO(  
4:   else if lookahead =  $\uparrow$  then  
5:     RECONHECER( $\uparrow$ )  
6:     RECONHECER(i d)  
7:   else if lookahead = array then  
8:     RECONHECER(array),  
9:     RECONHECER([)  
10:    PRIMITIVO(  
11:    RECONHECER(]),  
12:    RECONHECER(of)  
13:    TIPO(  
14:  else  
15:    ERRO(  

```

Procedimento associado ao não terminal *primitivo*

```
1: procedure PRIMITIVO( )  
2:   if lookahead = integer then  
3:     RECONHECER(integer)  
4:   else if lookahead = char then  
5:     RECONHECER(char)  
6:   else if lookahead = num then  
7:     RECONHECER(num)  
8:     RECONHECER(:)  
9:     RECONHECER(num)  
10:  else  
11:    ERRO( )
```

Primeiros símbolos

Definição de primeiros símbolos

Seja α o lado direito de uma produção. Então $\text{PRIMEIRO}(\alpha)$ é o conjunto de tokens que figuram como primeiros símbolos de uma ou mais cadeias geradas a partir de α . Se ϵ pode ser gerado a partir de α , então ϵ pertence a $\text{PRIMEIRO}(\alpha)$.

Por exemplo, na gramática de geração de subtipos em Pascal,

$$\text{PRIMEIRO}(\textit{primitivo}) = \{ \text{integer}, \text{char}, \text{num} \}$$

e

$$\text{PRIMEIRO}(\uparrow \text{id}) = \{ \uparrow \}$$

Primeiros símbolos e análise gramatical preditiva

- ▶ A análise gramatical preditiva depende dos conjuntos $\text{PRIMEIRO}(X)$ de todos não-terminais X da gramática

Primeiros símbolos e análise gramatical preditiva

- ▶ A análise gramatical preditiva depende dos conjuntos $\text{PRIMEIRO}(X)$ de todos não-terminais X da gramática
- ▶ Isto acontece principalmente nos casos onde a gramática possui duas ou mais produções para um mesmo não-terminal (por exemplo, $A \rightarrow \alpha$ e $A \rightarrow \beta$)

Primeiros símbolos e análise gramatical preditiva

- ▶ A análise gramatical preditiva depende dos conjuntos $\text{PRIMEIRO}(X)$ de todos não-terminais X da gramática
- ▶ Isto acontece principalmente nos casos onde a gramática possui duas ou mais produções para um mesmo não-terminal (por exemplo, $A \rightarrow \alpha$ e $A \rightarrow \beta$)
- ▶ Para que a análise gramatical recursiva descendente seja preditiva é necessário que os primeiros símbolos de cada produção sejam distintos

Primeiros símbolos e análise gramatical preditiva

- ▶ A análise gramatical preditiva depende dos conjuntos $\text{PRIMEIRO}(X)$ de todos não-terminais X da gramática
- ▶ Isto acontece principalmente nos casos onde a gramática possui duas ou mais produções para um mesmo não-terminal (por exemplo, $A \rightarrow \alpha$ e $A \rightarrow \beta$)
- ▶ Para que a análise gramatical recursiva descendente seja preditiva é necessário que os primeiros símbolos de cada produção sejam distintos
- ▶ No exemplo dado,

$$\text{PRIMEIRO}(\alpha) \cap \text{PRIMEIRO}(\beta) = \emptyset$$

Primeiros símbolos e análise gramatical preditiva

- ▶ A análise gramatical preditiva depende dos conjuntos $\text{PRIMEIRO}(X)$ de todos não-terminais X da gramática
- ▶ Isto acontece principalmente nos casos onde a gramática possui duas ou mais produções para um mesmo não-terminal (por exemplo, $A \rightarrow \alpha$ e $A \rightarrow \beta$)
- ▶ Para que a análise gramatical recursiva descendente seja preditiva é necessário que os primeiros símbolos de cada produção sejam distintos
- ▶ No exemplo dado,

$$\text{PRIMEIRO}(\alpha) \cap \text{PRIMEIRO}(\beta) = \emptyset$$

- ▶ Caso esta condição se verifique para todos os pares de produções distintas de um mesmo não-terminal, a produção γ deve ser usada se $lookahead \in \text{PRIMEIRO}(\gamma)$

Projeto de um analisador gramatical preditivo

Um analisador gramatical preditivo é um programa que contém um procedimento para cada não-terminal. Cada procedimento deve seguir dois passos:

Projeto de um analisador gramatical preditivo

Um analisador gramatical preditivo é um programa que contém um procedimento para cada não-terminal. Cada procedimento deve seguir dois passos:

1. Determinar a produção a ser usada a partir de *lookahead*. Para tanto, deve ser localizada, entre as produções $\alpha_1, \alpha_2, \dots, \alpha_N$, a produção α_i tal que $lookahead \in \text{PRIMEIRO}(\alpha_i)$ (deve valer a seguinte propriedade: $\text{PRIMEIRO}(\alpha_i) \cap \text{PRIMEIRO}(\alpha_j) = \emptyset$ se $i \neq j$). Se $\alpha_k = \epsilon$ para algum k , α_k deve ser usada se *lookahead* não estiver presente em nenhuma outra produção

Projeto de um analisador gramatical preditivo

Um analisador gramatical preditivo é um programa que contém um procedimento para cada não-terminal. Cada procedimento deve seguir dois passos:

1. Determinar a produção a ser usada a partir de *lookahead*. Para tanto, deve ser localizada, entre as produções $\alpha_1, \alpha_2, \dots, \alpha_N$, a produção α_i tal que $lookahead \in \text{PRIMEIRO}(\alpha_i)$ (deve valer a seguinte propriedade: $\text{PRIMEIRO}(\alpha_i) \cap \text{PRIMEIRO}(\alpha_j) = \emptyset$ se $i \neq j$). Se $\alpha_k = \epsilon$ para algum k , α_k deve ser usada se *lookahead* não estiver presente em nenhuma outra produção
2. Identificada a produção, o procedimento imita a produção, reconhecendo os terminais da produção e chamando os procedimentos dos não-terminais, na mesma ordem da produção

Produções recursivas à esquerda

Definição de produção recursiva à esquerda

Uma produção é recursiva à esquerda se o não-terminal à esquerda da produção figura como primeiro símbolo da produção. Por exemplo, se α e β são sequências de terminais e não-terminais que não iniciam em A , então a produção

$$A \rightarrow A\alpha \mid \beta$$

é recursiva à esquerda.

Produções recursivas à esquerda

Definição de produção recursiva à esquerda

Uma produção é recursiva à esquerda se o não-terminal à esquerda da produção figura como primeiro símbolo da produção. Por exemplo, se α e β são sequências de terminais e não-terminais que não iniciam em A , então a produção

$$A \rightarrow A\alpha \mid \beta$$

é recursiva à esquerda.

Observação: analisadores gramaticais recursivos descendentes pode rodar indefinidamente caso usem uma produção recursiva à esquerda

Produções recursivas à direita

Definição de produção recursiva à direita

Uma produção é recursiva à direita se o não-terminal à esquerda da produção figura como último símbolo da produção. Por exemplo, se α e β são sequências de terminais e não-terminais que não terminam em R , então a produção

$$\begin{aligned} A &\rightarrow \beta R \\ R &\rightarrow \alpha R \mid \epsilon \end{aligned}$$

é recursiva à direita.

Produções recursivas à direita

Definição de produção recursiva à direita

Uma produção é recursiva à direita se o não-terminal à esquerda da produção figura como último símbolo da produção. Por exemplo, se α e β são sequências de terminais e não-terminais que não terminam em R , então a produção

$$\begin{aligned} A &\rightarrow \beta R \\ R &\rightarrow \alpha R \mid \epsilon \end{aligned}$$

é recursiva à direita.

Observação: produções recursivas à direita dificultam a tradução de expressões que contém operadores associativos à esquerda