O que é e como funciona a análise sintática ascendente e descendente?

Perguntada 4 anos, 9 meses atrás Ativa 4 anos, 9 meses atrás Vista 3mil vezes

5

1

De acordo com algumas pesquisas que fiz, a Análise Sintática na computação, conhecido como *parsing* em inglês, é o processo de analisar uma sequência de entrada (lida de um arquivo de computador ou do teclado, por exemplo) para determinar sua estrutura gramatical segundo uma determinada gramática formal.

> A análise sintática transforma um texto na entrada em uma estrutura de dados, em geral uma árvore, o que é conveniente para processamento posterior e captura a hierarquia implícita desta entrada. Através da análise léxica é obtido um grupo de tokens, para que o analisador sintático use um conjunto de regras para construir uma árvore sintática da estrutura.

O que é e como é realizado o procedimento de análise sintática ascendente e descendente?

compiladores

Compartilhar

Melhore esta pergunta Seguir

editada 13/04/17 às 12:59

Comunidade Bot

1

perguntada 2/02/17 às 12:11

viana

27,2mil 17 78 170

1 Resposta

Ativos | Mais antigo | Votos

1 Resposta

Ativos | Mais antigo | Votos

3

De forma resumida:

- **Análise sintática ascendente**, também chamada de *bottom-up*, o analisador pode iniciar com um entrada de dados e tentar reescrevê-la até o símbolo inicial. Intuitivamente, o analisador tentar localizar os elementos mais básicos, e então elementos maiores que contêm os elementos mais básicos, e assim por diante. Exemplo: analisador sintático LR (**L**eft-to-right **R**ight-most-derivation).
- **Análise sintática descendente**, também chamada de *top-down*, o analisador pode iniciar com o símbolo inicial e tentar transformá-lo na entrada de dados. Intuitivamente, o analisador inicia dos maiores elementos e os quebra em elementos menores. Exemplo: analisador sintático LL (**L**eft-to-right **L**eft-most-derivation)

Análise sintática

A análise sintática é a segunda etapa do processo de compilação (a primeira é a análise léxica) e na maioria dos casos utiliza gramática livre de contexto para especificar a sintaxe de uma linguagem de programação. A principal tarefa do analisador sintático, conhecido como *parsing*, é determinar se o programa de entrada representado pelo fluxo de tokens possui as sentenças válidas para a linguagem de programação. No caso afirmativo, queremos adicionalmente descobrir a maneira (ou uma das maneiras) pela qual a cadeia pode ser derivada seguindo as regras da gramática.

Fazendo uma analogia gramática da língua portuguesa, que estuda a disposição das palavras em uma frase, essa parte da compilação é responsável por determinar se uma cadeia de símbolos léxicos pode ser gerada por uma gramática.

As gramáticas livres de contexto representam uma gramática formal e pode ser escrita através de algoritmos fazendo a derivação de todas as possíveis construções da linguagem. Essas derivações tem como objetivo determinar se um fluxo de palavras se encaixa na sintaxe da linguagem de programação.

Uma derivação é uma sequência de substituições de não-terminais por uma escolha das regras de produção gramaticais. Quando fazemos a derivação deve-se aplicar a regra de produção para substituir cada símbolo não terminal por um símbolo terminal, isso permite identificar se certa cadeias de caracteres pertence a linguagem, as regras expandem todas as produções possíveis. Como resultado desse processo temos a árvore de derivação, que é uma alternativa gráfica para mostrar o processo de derivação de uma sentença em uma gramática.

A maneira pela qual a árvore de derivação da cadeia analisada x é construída, diz se a análise sintática é descendente ou ascendente.

- **Análise sintática descendente**, também chamada de *top-down*: a árvore de derivação correspondente a x é construída de cima para baixo, ou seja, da raiz (o símbolo inicial S) para as folhas, onde se encontra x. Nesse tipo de análise, é preciso decidir *qual*

regra $A \to \beta$ será aplicada a um nó rotulado por um não-terminal A. A *expansão* de A é feita criando nós filhos rotulados com os símbolos de β.

- **Análise sintática ascendente**, também chamada de *bottom-up*: a árvore de derivação correspondente a x é construída de baixo para cima, ou seja, das folhas, onde se encontra x, para a raiz, onde se encontra o símbolo inicial S. Nesse tipo de análise, é preciso decidir *quando* a regra $A \to \beta$ será aplicada, e devemos encontrar nós vizinhos rotulados com os símbolos de β. A *redução* pela regra $A \to \beta$ consiste em acrescentar à árvore um nó A, cujos filhos são os nós correspondentes aos símbolos de β.

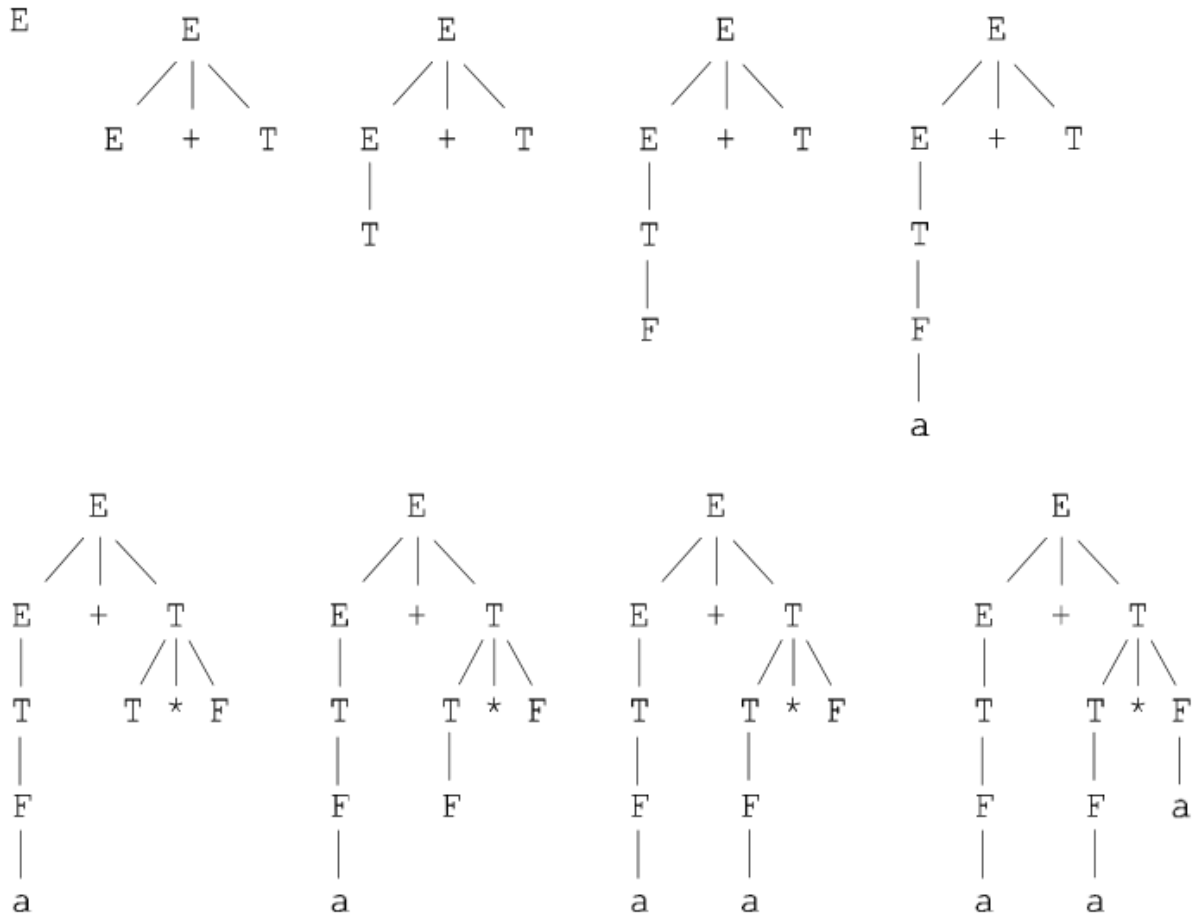
Nos dois tipos, as árvores são construídas da esquerda para a direita. A razão para isso é que a escolha das regras deve se basear na cadeia a ser gerada, que é lida da esquerda para a direita.

Exemplo, considere a seguinte gramática e a cadeia `x = a+a*a`:

1. E → E + T
2. E → T
3. T → T*F
4. T → F

5. F → (E)
6. F → a

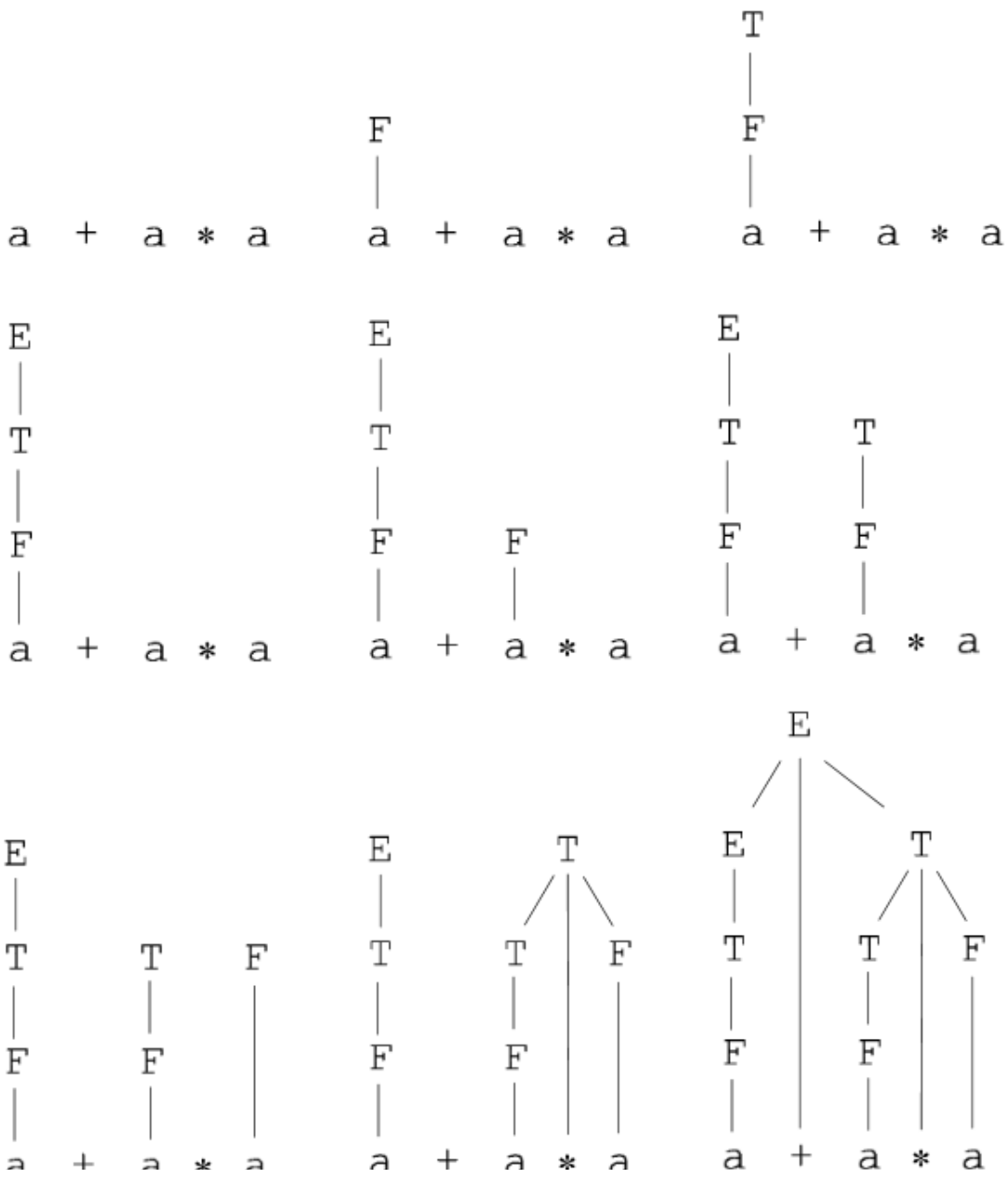
Na análise descendente:



As regras são consideradas na ordem 1 2 4 6 3 4 6 6, a mesma ordem em que as regras são usadas na derivação esquerda:

```
E ⇒ E+T ⇒ T+T⇒ a+T ⇒ a+T*F ⇒ a+F*F ⇒ a+a*F ⇒ a+a*a
```

Com a análise ascendente, por outro lado, as regras são identificadas na ordem 6 4 2 6 4 6 3 1, neste caso, a ordem das regras corresponde à derivação direita, invertida:



```
a+a*a ⇐ F+a*a ⇐ T+a*a ⇐ E+a*a ⇐ E+F*a ⇐ E+T*a
```

Ou seja:

```
E ⇒ E+T ⇒ E+T*F ⇒ E+T*a ⇒ E+F*a ⇒ E+a*a ⇒ T+a*a ⇒ F+a*a ⇒ a+a*a
```

Referências:

- Análise Sintática
- Compiladores - Análise Sintática
- Construção de compiladores/Análise sintática

Compartilhar

Melhore esta resposta Seguir

editada 13/04/17 às 12:59

Comunidade Bot
1

respondida 3/02/17 às 13:56

Taisbevalle
9.000 6 26 59