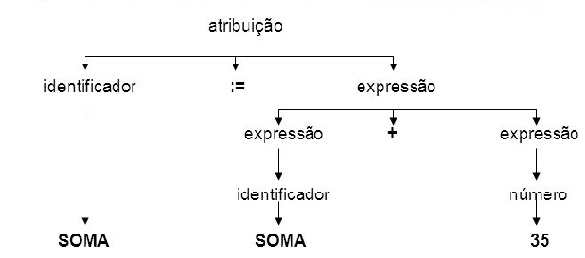
**Conceitos de Análise Sintática e Análise Sintática Descendente**

O que é possível fazer genericamente com a Análise sintática:

Tem a função de combinar a lista de Tokens criando uma Árvore sintática. E ela deve rejeitar todos os Tokens inválidos.



A análise sintática é mais complexa que a Léxica. Pois precisamos de uma linguagem mais avançada.

Abaixo encontra-se suas características principais.

* São linguagens livre de contexto, constituem um conjunto de linguagens que podem ser geradas por gramáticas livres de contexto (GLC), reconhecidas por autômatos de pilha. As linguagens são projetadas a partir da GLC.
* se uma cadeia de átomos (tokens), isto é, o programa já analisado pelo analisador léxico, pode ser gerado por uma gramática.
* Seu objetivo é a construção da árvore sintática ou apenas a decisão se a cadeia fornecida é ou não uma sentença da gramática que define a linguagem.
* Definição baseada em derivação para uma linguagem gerada por uma GLC. Dado uma linguagem qualquer definimos os terminais e os produtores e a partir disso definimos a linguagem.

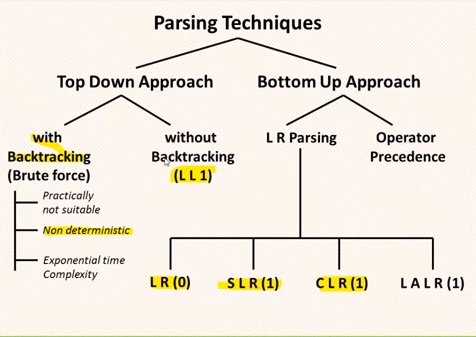
**Derivações**

A derivação é a substituição do conjunto de símbolos não terminais por símbolos terminais começando pelo símbolo inicial, ao final desse processo o resultado é a forma como a linguagem deve assumir.

Durante a derivação devemos aplicar as regras de produção para substituir cada símbolo não terminal por um símbolo terminal, isso permite identificar se certas cadeias de caracteres pertence a linguagem, as regras expandem todas as produções possíveis. Como resultado desse processo temos a árvore de derivação.

**Tipos de derivação:**

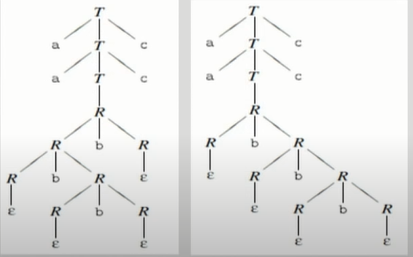
* Top-Down:Examina os símbolos terminais da esquerda para a direita - forma a árvore sintatica de cima para baixo.
* L(eft-to-right) L(eft-most-derivation) => LL
* Bottom-Up: Examina os símbolos terminais da direita para a esquerda - forma a árvore sintatica de baixo para cima
* S(imple) L(eft-to-right) R(ight-most-derivation) => SLR
* L(eft-to-right) R(ight-most-derivation) => LR
* L(ook) A(head) L(eft-to-right) R(ight-most-derivation) => LALR

****

Independente do algoritmo utilizado a derivação deve produzir o mesmo resultado, ou seja, a mesma árvore de derivação, caso o resultado seja diferente temos uma ambiguidade.

**Árvore Sintática**

* As arvores podem ser representadas a partir de um símbolo inicial.
* As folhas devem conter somente símbolos terminais
* Lendo as folhas da esquerda para a direita temos a palavra derivada
* Produtores que levam ao vazio também devem ser representadas.



**A raiz da arvore de derivação é sempre o símbolo inicial, os vértices interiores são os símbolos não terminais, os símbolos terminais são as folhas.**

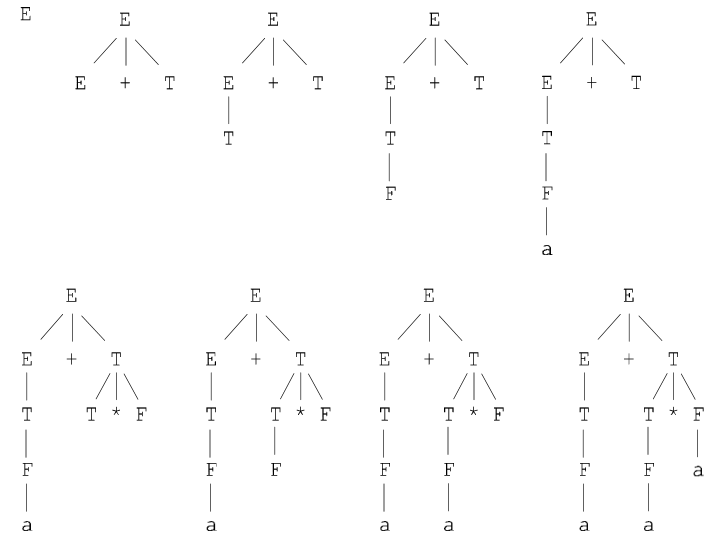
**Conceito de análise sintática descendente**

também chamada de top-down: a árvore de derivação correspondente a x é construída de cima para baixo, da raiz (o símbolo inicial S) para as folhas, onde se encontra x. Nesse tipo de análise, é preciso decidir qual regra A→β 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.

Essas árvores são construídas da esquerda para a direita, pois 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.

Na construção descendente o objetivo é iniciar a análise com uma lista que contém inicialmente apenas o símbolo sentencial; a partir da análise dos símbolos presentes na sentença, busca-se aplicar regras que permitam expandir os símbolos na lista até alcançar a sentença desejada.

Outra caracteristica é obter uma derivação mais à esquerda para uma sentença. Em termos de árvores gramaticais, a construção descendente busca a construção de uma árvore a partir da raiz usando pré-ordem para definir o próximo símbolo não-terminal que deve ser considerado para análise e expansão.



As regras são consideradas na ordem 1 2 4 6 3 4 6 6.

Sendo:

E ⇒ E+T ⇒ T+T⇒ a+T ⇒ a+T\*F ⇒ a+F\*F ⇒ a+a\*F ⇒ a+a\*a

Antes os compiladores usavam dois tipos de analisadores sintáticos. Analisadores baseados em precedência de operadores, sendo adequados à análise de expressões aritméticas. Analisadores do tipo descendentes recursivos implementam a técnica de construção descendente através de um conjunto de rotinas mutuamente recursivas para realizar a análise, sendo normalmente utilizados para outros comandos que não expressões aritméticas.