INE5318
Construção de Compiladores

AULA 5: Análise Semântica

Ricardo Azambuja Silveira

INE-CTC-UFSC

E-Mail: silveira@inf.ufsc.br

URL: www.inf.ufsc.br/~silveira

Recuperação de Erros

- Sabemos que ocorre um erro quando:
  - o símbolo corrente da entrada não corresponde ao *terminal* contido no topo da pilha OU
  - o símbolo corrente da entrada não possui produção correspondente a partir do *não-terminal* contido no topo da pilha.
- Tais erros podem ser recuperados através da modalidade do desespero, i.e., através do descarte de símbolos da entrada até a localização de um *token* de sincronização.

Recuperação de Erros: Modalidade Pânico

- A eficiência da recuperação de erros desta forma depende da escolha de um conjunto adequado de *tokens* de sincronização.
- Não existem regras formais para tal escolha que pode ser baseada em técnicas heurísticas
- A recuperação de erros é provável na maioria dos casos embora não possa ser assegurada completamente.

Recuperação de Erros: Modalidade Pânico

- Considerando a ocorrência de um erro durante a expansão do *não-terminal* A:
  - **Técnica 1**:

    Como *tokens* de sincronização usam-se todos os símbolos em folow(A).

    Descartam-se os símbolos de entrada até encontrar-se um elemento de folow(A), quando também descartamos A da pilha

    É provável que a análise sintática possa prosseguir.

Recuperação de Erros: Modalidade Pânico

- **Técnica 2**:

  Como somente os símbolos em folow(A) não são suficiente para a sincronização, adicionam-se os símbolos first das produções que se expandem em A.
  Descartam-se os símbolos de entrada até encontrar-se do conjunto de sincronização, quando também descartamos A.
  É provável que a análise sintática possa prosseguir.

Recuperação de Erros: Modalidade Pânico

- **Técnica 3**:

  Se os símbolos em first(A) são adicionados ao conjunto de sincronização, poderá ser possível retomar a análise a partir de A se um símbolo que figura em first(A) estiver na entrada.

- **Técnica 4**:

  Se o não-terminal A puder gerar $\in$, tal produção pode ser usada como *default*. Pode-se assim adiar a detecção do erro sem haver possibilidade de perde-lo.

Recuperação de Erros: Modalidade Pânico

- **Técnica 5**:

  Se um terminal no topo da pilha não pode ser reconhecido, pode-se remove-lo. Sugere-se a emissão de mensagem de aviso nestes casos.

  A análise provavelmente poderá prosseguir.

  Tal enfoque corresponde a adicionar todos os *tokens* possíveis como símbolos do conjunto de sincronização.

Recuperação de Erros

- Toda e qualquer tentativa de recuperação de erros não deve provocar um laço infinito na análise sintática da entrada. Uma forma de proteção é assegurar-se que a pilha foi encurtada após a ação de correção.
- É sempre questionável a inserção ou alteração de símbolos na pilha quando tais operações “criam” construções inexistentes na linguagem sendo analisada.

Mensagens de Erro

- Todas as mensagens de erro produzidas por um compilador devem ser informativas o suficiente para permitir a rápida localização e correção do erro.
- Sugere-se que cada entrada vazia da tabela sintática preditiva contenha apontadores para rotinas de tratamento de erro onde mensagens apropriadas serão fornecidas conforme o erro que as provocou.

Gramática de Atributos

Gramática livre de contexto na qual cada símbolo possui um conjunto associado de atributos. A cada produção pode estar associada um conjunto de regras semânticas, que podem alterar valores de atributos, emitir código, atualizar a tabela de símbolos, emitir mensagens de erro ou realizar quaisquer outras atividades. Em geradores de analisadores sintáticos estas regras são geralmente descritas em linguagem de programação.

Os atributos são classificados como:

- **Atributos Sintetizados**: O valor do atributo de um nó é computado a partir dos atributos de seus filhos.
- **Atributos Herdados:** O valor do atributo de um nó é computado a partir dos atributos dos irmãos e pais do nó.

Gramática de atributos

Desenvolvida por Knuth, 1968

Gramáticas livre de contexto não tem capacidade para descrever completamente a sintática de linguagens de programação

Mecanismos adicionados a GLC pata tratar algumas informações semânticas relacionadas as formas legais do programa na construção das árvores de análise

Valor primário da gramática de atributos:

- Especificação da semântica estática
- Projeto de compiladores (verificação da semântica estática)

Gramática de atributos

Definição:

Uma *gramática de atributo* é a gramática livre de contexto com as seguintes adições:

- Para cada símbolo gramatical x há um conjunto A(x) de atributos
- Cada regra tem um conjunto de funções que definem certos atributos dos símbolos não-terminais em uma regra
- Cada regre tem um conjunto (possivelmente vazio) de predicados para checar a consistência dos atributos

Gramática de atributos

- Seja a regra X0 -> X1 ... Xn
- Funções na forma S(X0) = f(A(X1), ... A(Xn)) definem, *atributos sintetizados*
- Funções na forma I(Xj) = f(A(X0), ... , A(Xn)), para i <= j <= n, definem *atributos herdados*
- Inicialmente, existem *atributos intrínsecos* nas folhas
- *Exemplo:* expressões na forma id + id
- - id's podem ser tipo int ou real
- - tipos dos dois id's devem ser os mesmos
- - tipos de expressão devem ser o mesmo que o tipo esperado
- *BNF:*
- <expr> -> <var> + <var>
- <var> -> id
- *Attributos:*
- *tipo_efetivo* – sintetizado para <var> e <expr>
- *tipo_esperado* – herdado para <expr>

Gramática de atributos

Regra sintática: <expr> -> <var>[1] + <var>[2]

Regra semantica: <expr>.actual_type ← <var>[1].actual_type

Predicado:

<var>[1].actual_type = <var>[2].actual_type

<expr>.expected_type = <expr>.actual_type

Regra sintática: <var> -> id

Regra semantica: <var>.actual_type ← lookup (id, <var>)

Gramática de atributos

Como os valores dos atributos são computados?

- 1. Se todos os atributos foram herdados, a árvore é decorada em ordem top-down.
- 2. Se todos os atributos foram sintetizados, a árvore é decorada em ordem bottom-up.
- 3. Em muitos casos, os dois tipos de atributos são usados e uma combinação de top-down e bottom-up é usada.

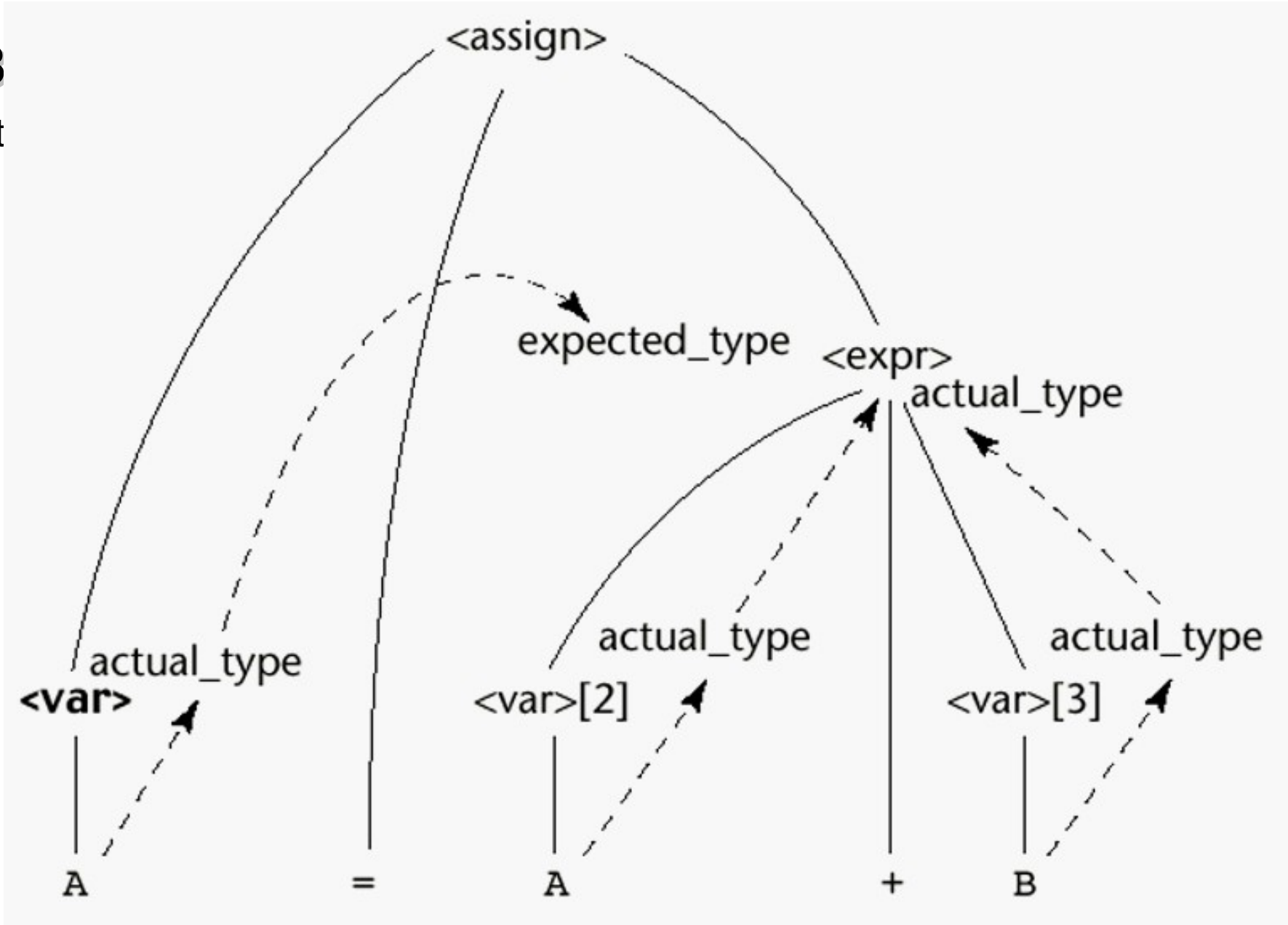
Gramática de atributos

- 1. <expr>.expected_type ← inherited from parent

- 2. <var>[1].actual_type ← lookup (A, <var>[1])
- <var>[2].actual_type ← lookup (B, <var>[2])
- <var>[1].actual_type =? <var>[2].actual_type

- 3. <expr>.actual_type ← <var>[1].actual_type
- <expr>.actual_type =? <expr>.expected_type

Figure 3.8
The flow of att



TRADUÇÃO DIRIGIDA POR SINTAXE

- Cada símbolo tem um conjunto de atributos associado, particionados em dois: atributos sintetizados e herdados.
- Um atributo pode representar uma palavra, um número, um tipo ou uma posição na memória
- O valor de um atributo é definido por uma regra semântica associada à produção:
  - um atributo sintetizado é computado a partir dos atributos dos filhos daquele nodo.
  - um atributo herdado é computado a partir dos atributos dos irmãos ou dos pais daquele nodo.

TRADUÇÃO DIRIGIDA POR SINTAXE

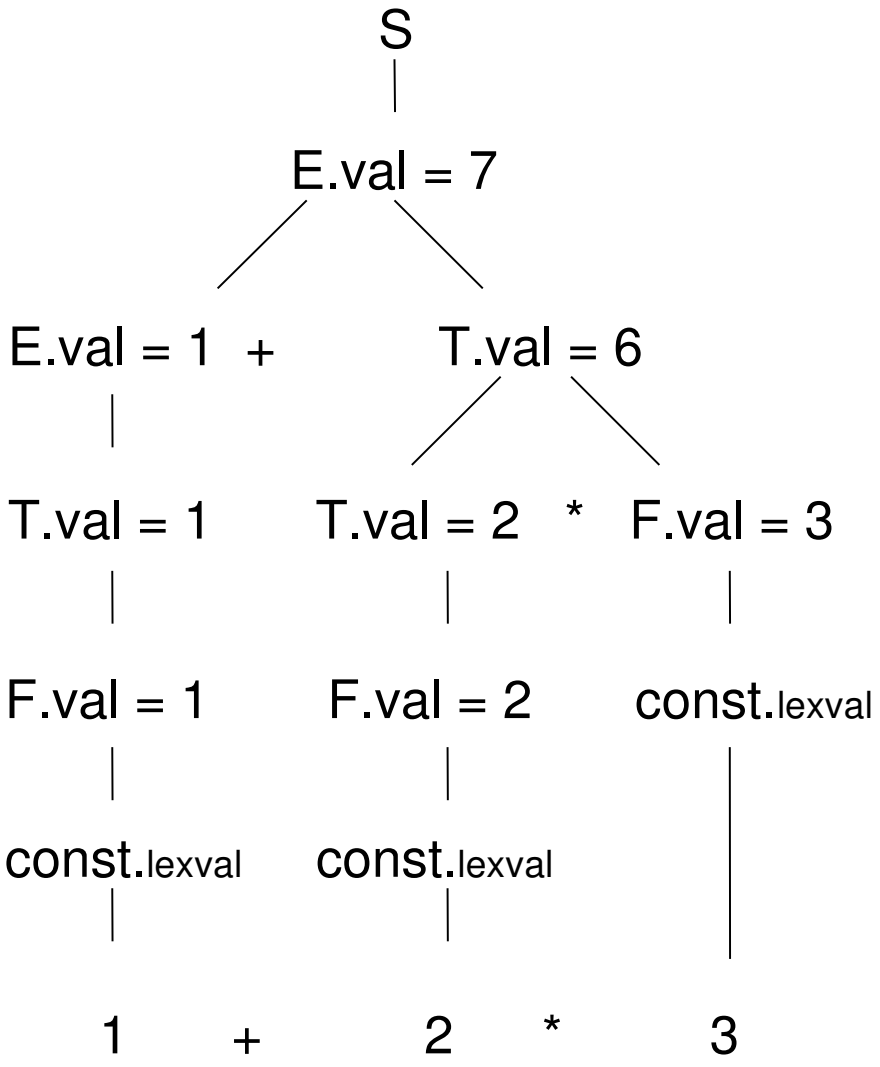
- Regras semânticas estabelecem dependências entre atributos que são representadas por um grafo. A partir do grafo de dependências, é derivada uma ordem de avaliação para as regras semânticas.
- Uma árvore de derivação que apresenta os valores dos atributos de cada nodo é denominada árvore de derivação “decorada”.

TRADUÇÃO DIRIGIDA POR SINTAXE

- Uma gramática de atributos é uma definição dirigida por sintaxe na qual as funções nas regras semânticas não têm efeitos colaterais.
- Assume-se que terminais tenham apenas atributos sintetizados, assim como o símbolo inicial da gramática. Um exemplo de definição dirigida por sintaxe é apresentado na Tabela a seguir

Gramática de Atributos

Produção	Regra Semântica
$S \rightarrow E$	{Imprimir (E.val)}
$E \rightarrow E_1 + T$	{E.val = E_1.val + T.val}
$E \rightarrow T$	{E.val = T.val}
$T \rightarrow T_1 * F$	{T.val = T_1.val * F.val}
$T \rightarrow F$	{T.val = F.val}
$F \rightarrow (E)$	{F.val = E.val}
$F \rightarrow$ ***const***	{F.val = ***const***.*lexval*}



Tipos de esquemas de tradução

- S-atribuídos: contém apenas atributos sintetizados (*bottom-up*)
- ações semânticas são dispostas à direita das produções

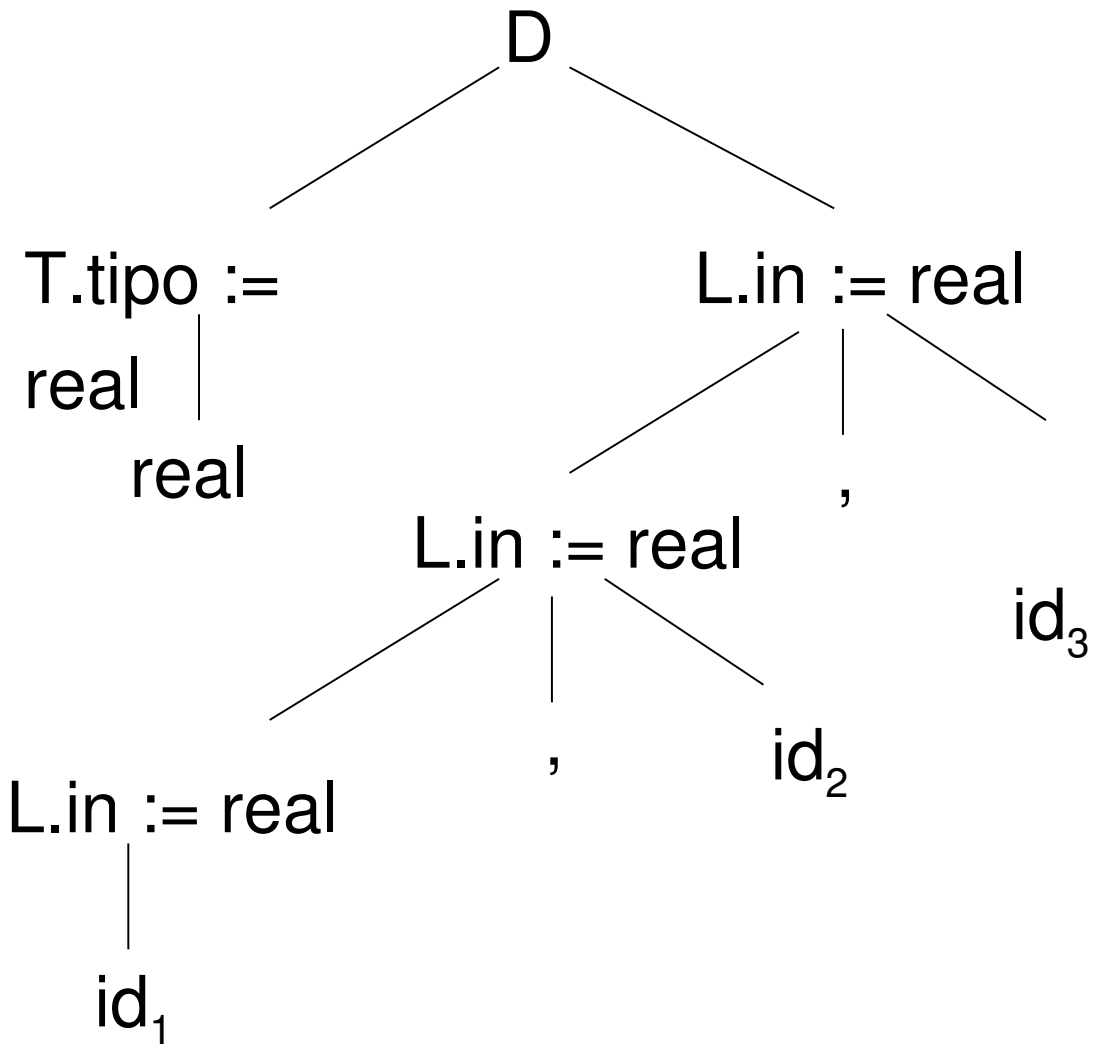
T→ int T.tipo := inteiro

- L-atribuídos: restringem o uso de atributos herdados

D → T {L.in := T.tipo} L

Produção	Regras Semânticas
$D \rightarrow T\ L$	L.in := T.tipo
$T \rightarrow int$	T.tipo := inteiro
$T \rightarrow real$	T.tipo := real
$L \rightarrow L_1, id$	L_1.in := L.in incluir_tipo(id, entrada, L.in)
$L \rightarrow id$	incluir_tipo(id, entrada, L.in)

Árvore de derivação

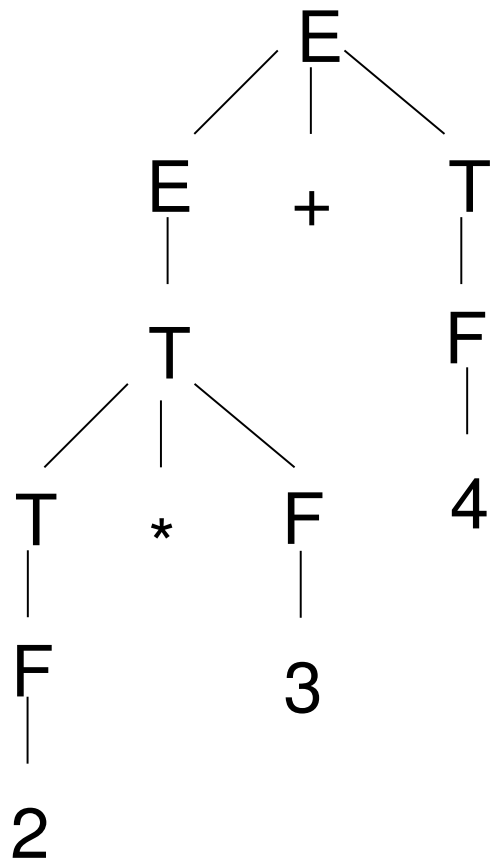


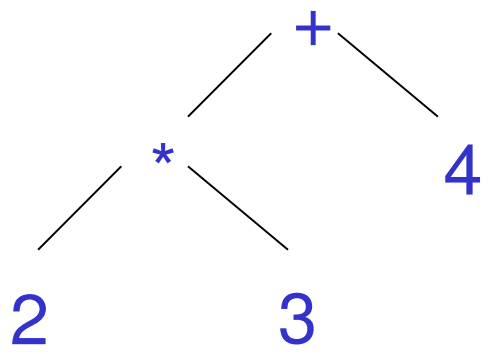
Produção	Regras Semânticas
D → T L	L.in := T.tipo
T → int	T.tipo := inteiro
T → real	T.tipo := real
L → L_1, id	L_1.in := L.in incluir_tipo(id, entrada, L.in)
L → id	incluir_tipo(id, entrada, L.in)

Árvores de Sintaxe

- Forma condensada da árvore de derivação, na qual somente os operandos aparecem nas folhas, enquanto que os operadores aparecem em nodos interiores da árvore

Árvore de derivação e árvore de sintaxe para a sentença 2*3+4





Exemplo

Produções	Regras semânticas
$E \rightarrow E_1 + T$	E.ptr := geranodo("+", E_1.ptr, T.ptr)
$E \rightarrow E_1 - T$	E.ptr := geranodo("-", E_1.ptr, T.ptr)
$E \rightarrow T$	E.ptr := T.ptr
$T \rightarrow (E)$	T.ptr := E.ptr
$T \rightarrow id$	T.ptr := gerafolha("id", id.nome)
$T \rightarrow num$	T.ptr := gerafolha("num", num.lexval)

Árvore sintática p/a expressão *x + 5 - y*