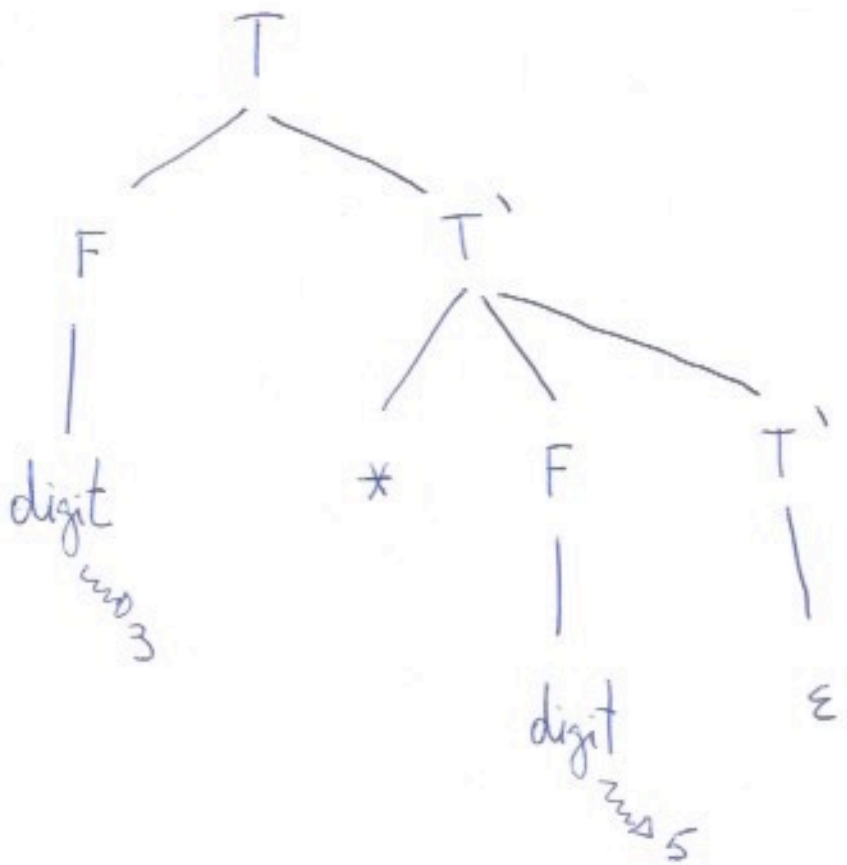
Syntax-Directed Translation (cont.)

- A gramática que vimos na última aula não era adequada para um parser top-down.
- Porquê? Tinha recursividade à esquerda.
- Mas podia ser usada com um parser bottom-up.
- Vamos agora ver um exemplo em que possamos usar um parser top-down.

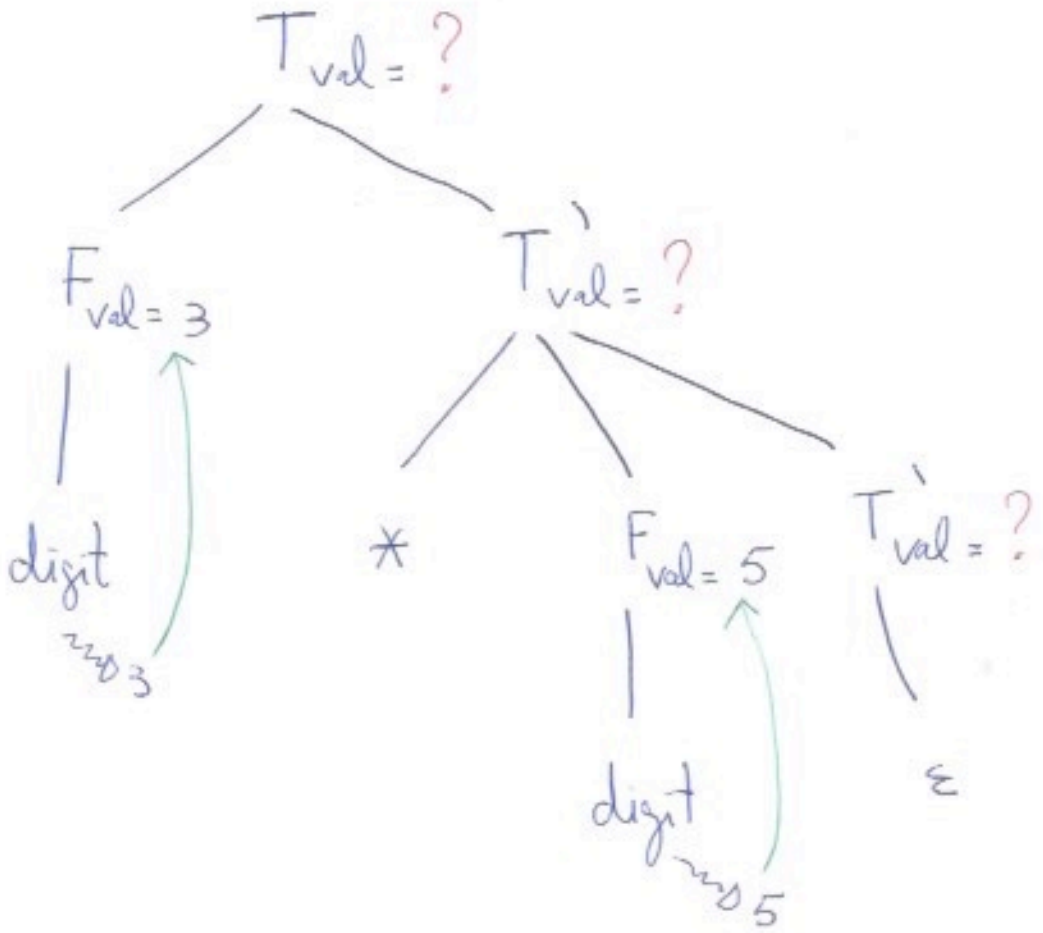
$$T \rightarrow F\,T'$$
$$T' \rightarrow *\,F\,T'$$
$$T' \rightarrow \varepsilon$$
$$F \rightarrow \text{digit}$$

- Vejamos o parsing para o input: 3 * 5



- Será que podemos ter um atributo val associado aos símbolos não terminais, como na gramática da aula passada, e calcular o seu valor ~~dos fatores~~ de baixo para cima?

- Vamos tentar por analogia com a aula passada. (Para já sem especificar regras semânticas...)

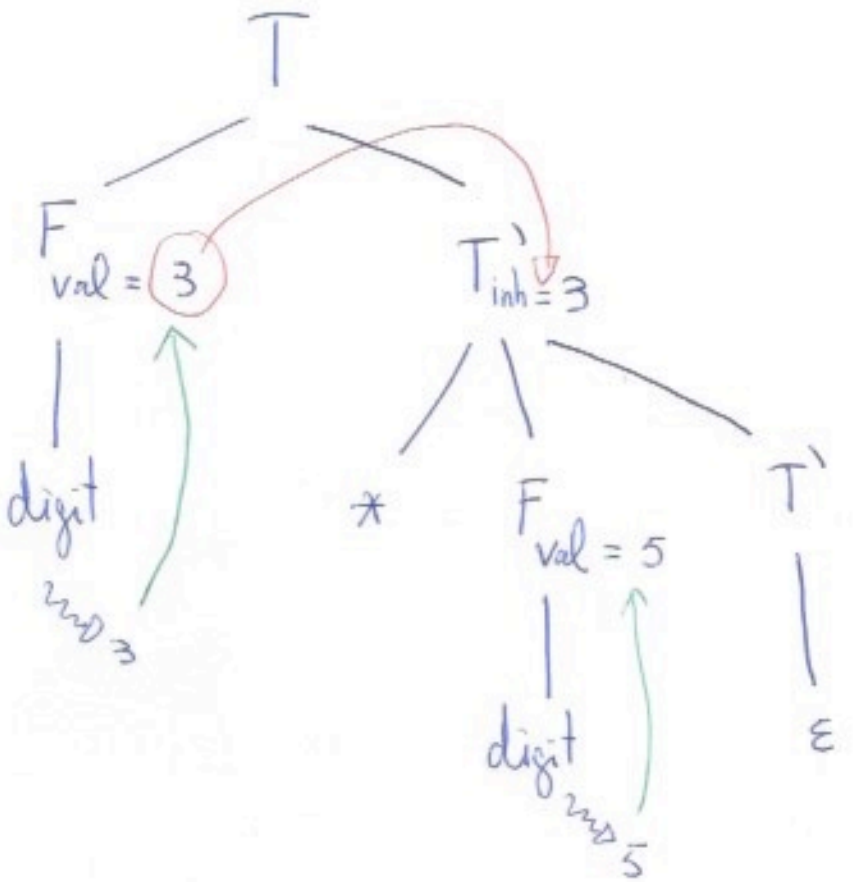


- Ficamos encalhados. Porquê?
- Queremos 3*5 mas o nó associado ao T' (filho de T) vai ter apenas *5 ...

- Pode-se resolver o problema se o valor F_{val} for "passado" para T'.

- Ou seja, T' teria um atributo cujo valor não depende dos nós filhos, mas sim de um nó irmão.

- Um atributo diz-se <u>herdado</u> (em inglês, inherited) se o seu valor depender apenas do nó pai e dos nós irmãos.



- Vejamos uma SDD (retirado do livro do dragão)

Produção	Regras Semânticas
$T \rightarrow F\,T'$	$T'.inh = F.val$ $T.val = T'.syn$
$T' \rightarrow * F\,T_1'$	$T_1'.inh = T'.inh * F.val$ $T'.syn = T_1'.syn$
$T' \rightarrow \varepsilon$	$T'.syn = T'.inh$
$F \rightarrow digit$	$F.val = digit.lexval$

T e F têm um atributo sintetizado : val

T' tem um atributo herdado : inh
e um atributo sintetizado : syn

S de synthesized

- Uma SDD diz-se "S-attributed" se todos os atributos forem sintetizados.

  → é o caso dos exemplos da aula passada.

- Uma SDD diz-se "L-attributed" se cada um dos atributos for: (L de left)

  1) sintetizado, ou

  2) herdado, mas com a seguinte restrição:

     Seja $X_{i.a}$ um atributo herdado de X_i na produção $A \to X_1 X_2 \dots X_n$

     A regra só pode usar:

     a) atributos herdados associados a A.

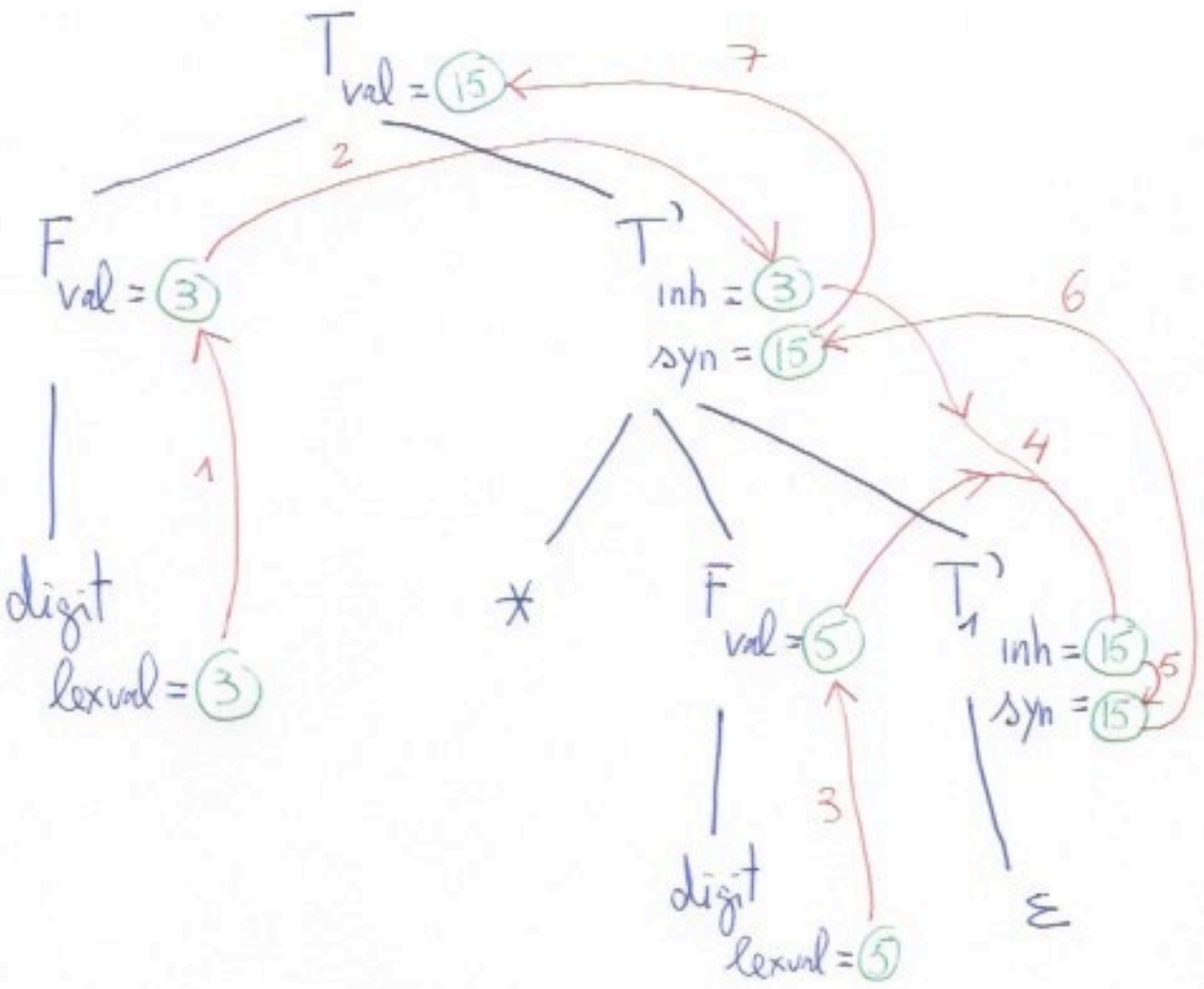
     b) atributos (herdados ou sintetizados) associados a símbolos à esquerda de X_i.

- Ou seja, para ser L-attributed, os atributos herdados só podem obter os seus valores através de atributos que já estejam calculados.

- A SDD que está na folha 5 é L-attributed.

- Quando assim é, é possível fazer um parser top-down preditivo que calcula o valor dos atributos à medida que faz o parsing.

- Exemplo. Input: 3*5



- Na página anterior, está indicado a vermelho a ordem pela qual os atributos seriam calculados.

Implementação (ler secção 5.5.1 do livro, pag 338)

- Como o parser é LL(1), sabemos sempre qual a produção a usar mediante o símbolo de lookahead.
- Temos uma função A() para cada símbolo não terminal A.
- A função A() recebe atributos herdados como argumentos de entrada, e retorna os atributos sintetizados do símbolo A.
- O corpo da função A() deve guardar em variáveis locais, os valores de atributos necessários para calcular os atributos mencionados na produção associada.

Exemplo em pseudocódigo para T'

```
T'() {
    if ( currToken == TIMES) {
        match (TIMES);
        F();
        T'();
    }
}
```

→ "parsing normal"

```
T'( T'inh ) {
    if ( currToken == TIMES ) {
        match (TIMES)
        Fval = F();
        T1'inh = T'inh * Fval ;
        T1'syn = T'(T1'inh) ;
        return T1'syn ;
    }
    else {
        T'syn = T'inh;
        return T'syn;
    }
}
```

→ "parsing extendido"

Fval
T_1'inh
T_1'syn
T'syn

→ vars locais