CS480
Translators

SDT and Intro to Parsing

Chap. 2

Class Example

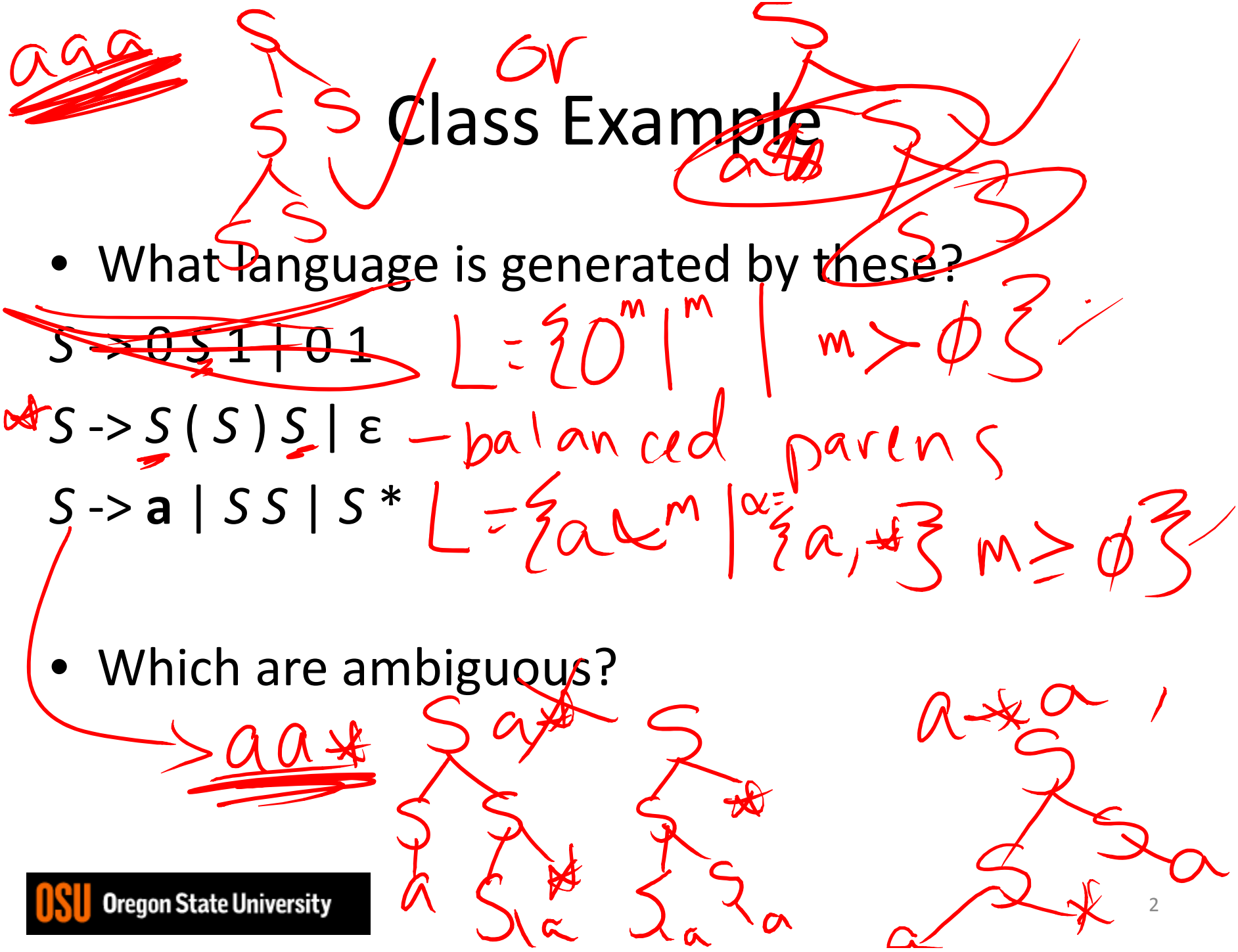
- What language is generated by these?

S -> 0 S 1 | 0 1

S -> S (S) S | ε

S -> **a** | S S | S *

- Which are ambiguous?



Syntax-Directed Translation

- Extend grammar
  - Attributes – rules
  - Translation scheme – actions

Synthesized Attributes

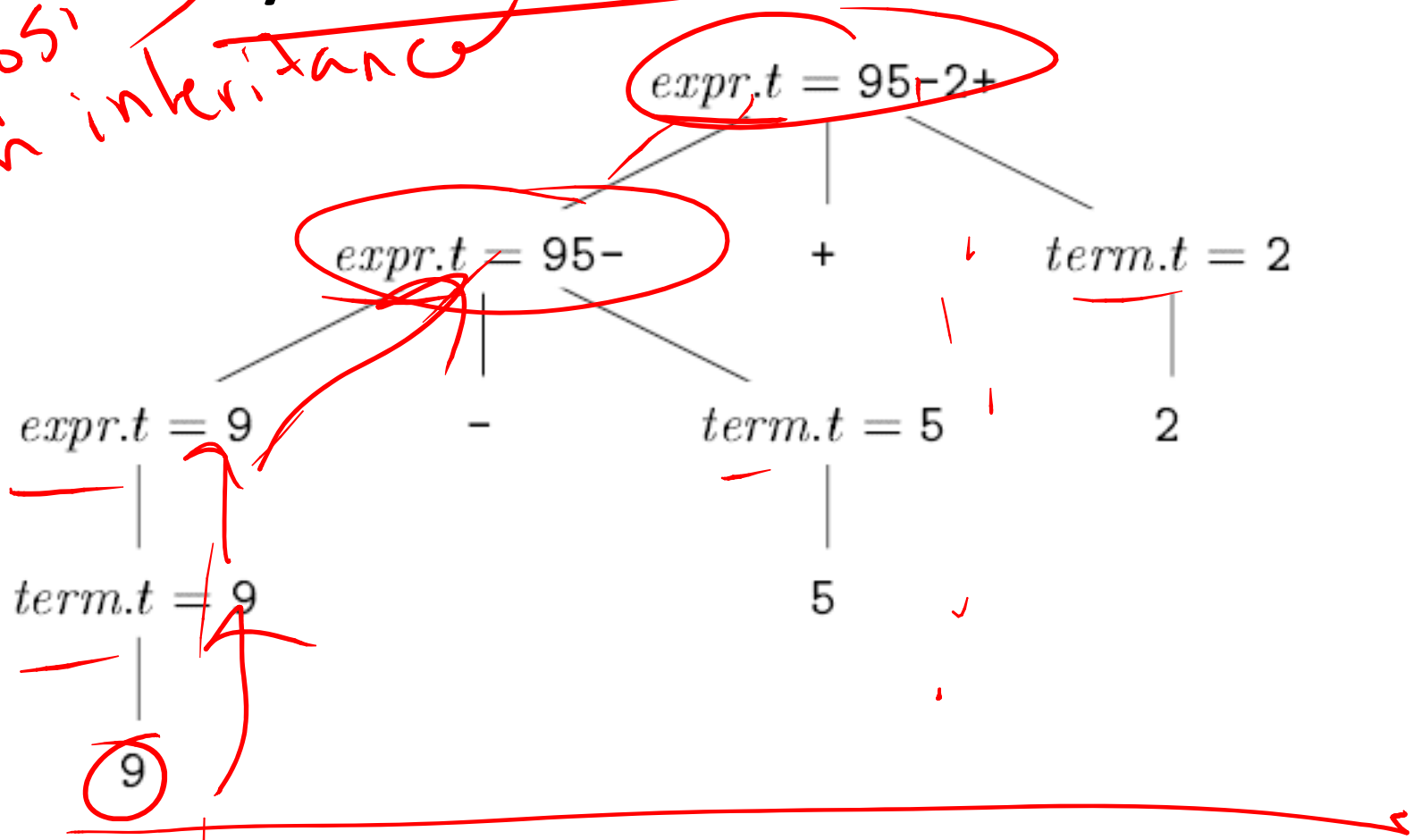


Figure 2.9: Attribute values at nodes in a parse tree

Syntax-Directed Definition

PRODUCTION	SEMANTIC RULES
$expr \rightarrow expr_1 + term$	$expr.t = expr_1.t \,\|\|\, term.t \,\|\|\, '+'$
$expr \rightarrow expr_1 - term$	$expr.t = expr_1.t \,\|\|\, term.t \,\|\|\, '-'$
$expr \rightarrow term$	$expr.t = term.t$
$term \rightarrow 0$	$term.t = '0'$
$term \rightarrow 1$	$term.t = '1'$
...	...
$term \rightarrow 9$	$term.t = '9'$

Figure 2.10: Syntax-directed definition for infix to postfix translation

Syntax-Directed Translation Scheme

rest -> + *term* {print('+')} $rest_1$

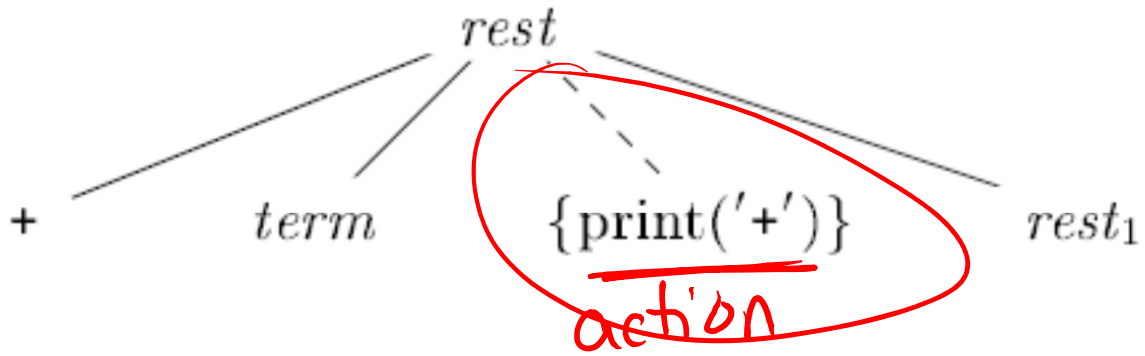


Figure 2.13: An extra leaf is constructed for a semantic action

- How does this differ from synthesized attributes?

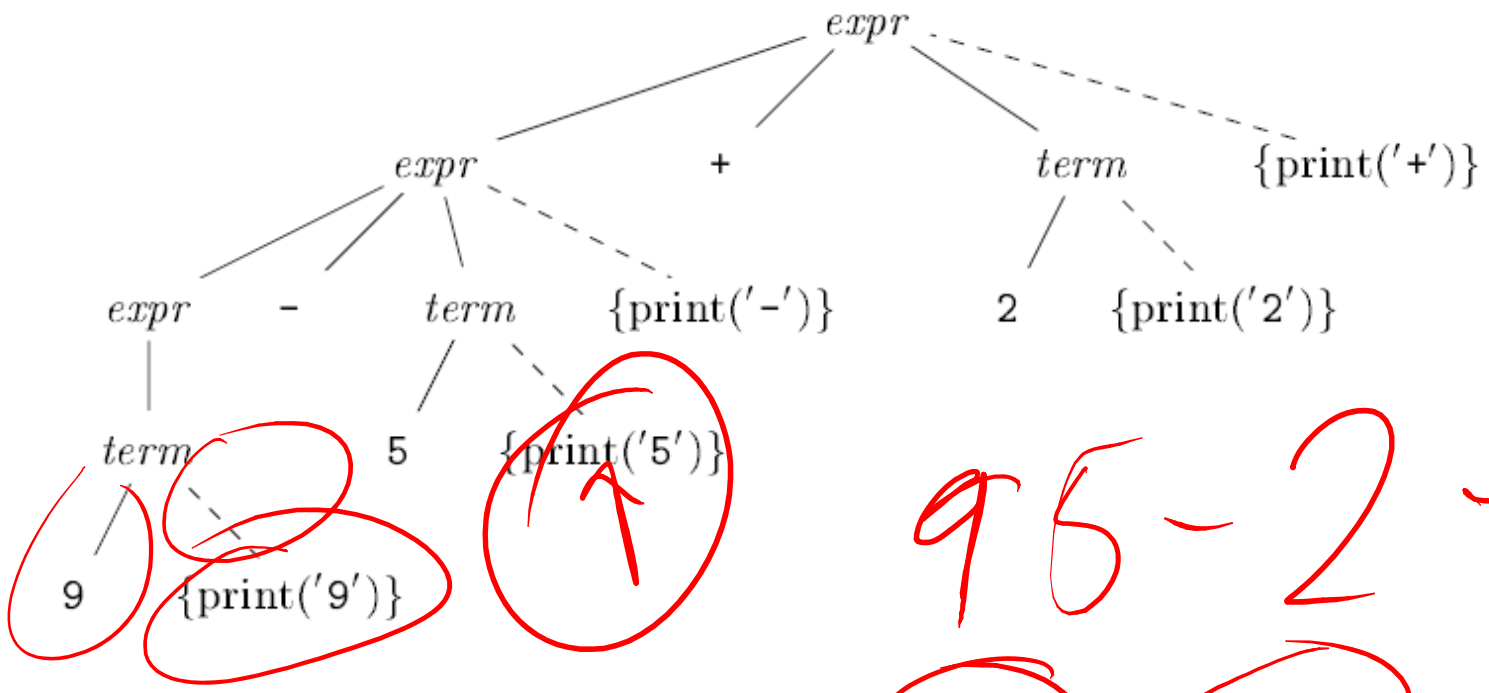


Figure 2.14: Actions translating 9-5+2 into 95-2+

$expr$	$\rightarrow$	$expr_1$ + $term$	{print('+')}
$expr$	$\rightarrow$	$expr_1$ - $term$	{print('-')}
$expr$	$\rightarrow$	$term$	
$term$	$\rightarrow$	0	{print('0')}
$term$	$\rightarrow$	1	{print('1')}
		...	
$term$	$\rightarrow$	9	{print('9')}

Figure 2.15: Actions for translating into postfix notation

What is Parsing?

- Determine how string is generated
- Methods
  - Top-down
  - Bottom-up

terminals

for (; expr ; expr) other

$$
\begin{array}{lcl}
stmt & \rightarrow & \textbf{expr ;} \\
 & | & \textbf{if (expr)} \ stmt \\
 & | & \textbf{for (} optexpr \ \textbf{;} \ optexpr \ \textbf{;} \ optexpr \ \textbf{)} \ stmt \\
 & | & \textbf{other} \\
 & & \\
optexpr & \rightarrow & \epsilon \\
 & | & \textbf{expr}
\end{array}
$$

Figure 2.16: A grammar for some statements in C and Java

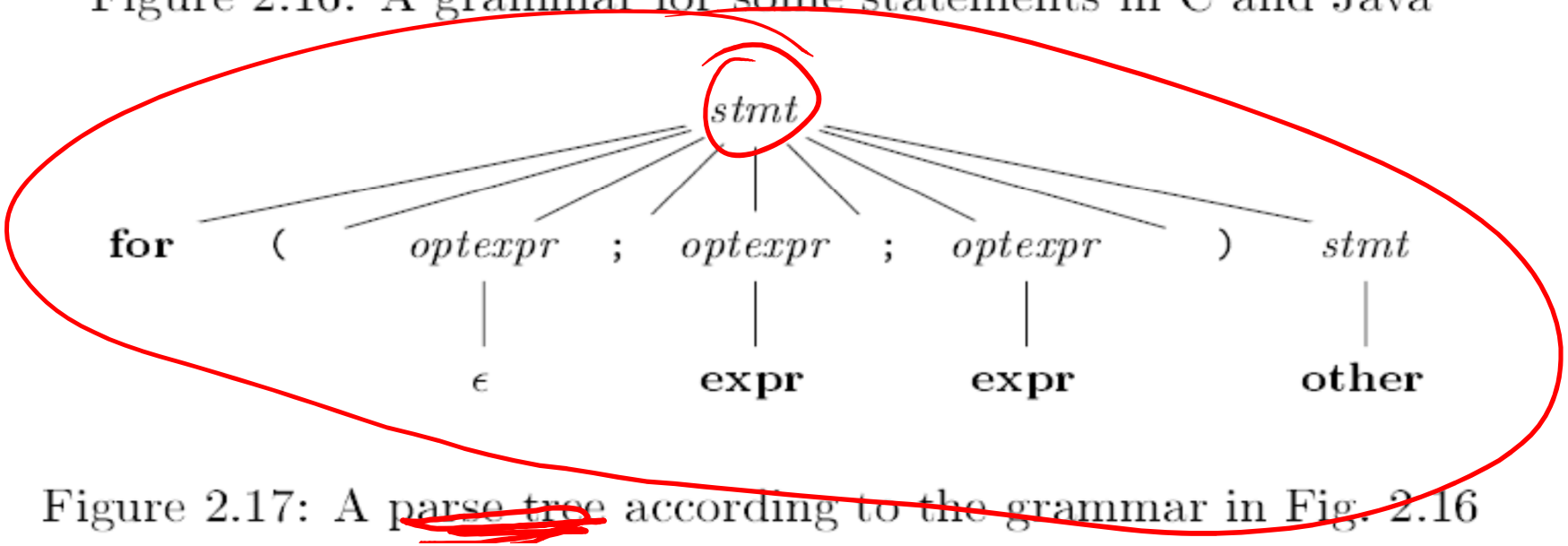
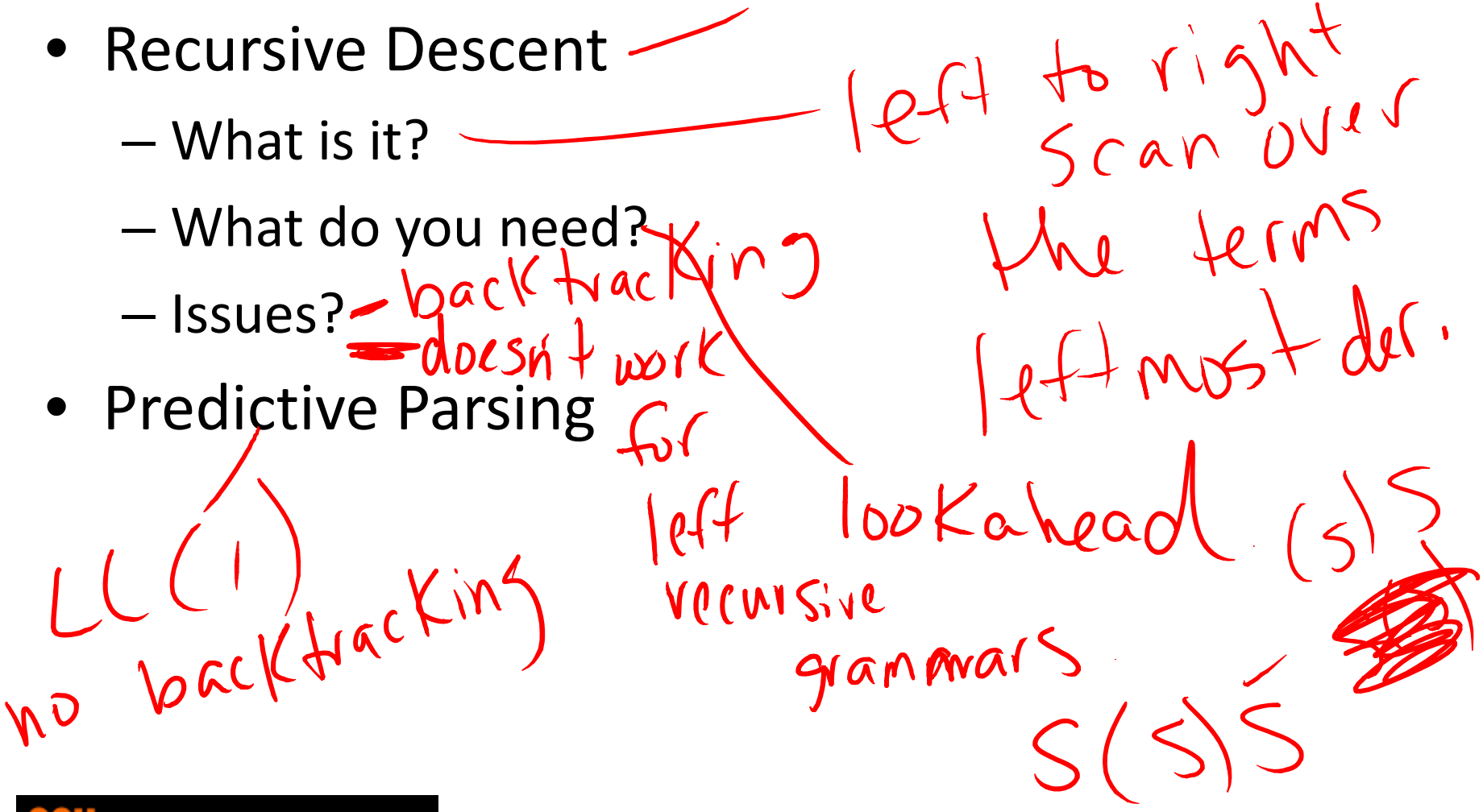


Figure 2.17: A parse tree according to the grammar in Fig. 2.16

Top-down Parsing

- Recursive Descent
  - What is it?
  - What do you need?
  - Issues?
- Predictive Parsing



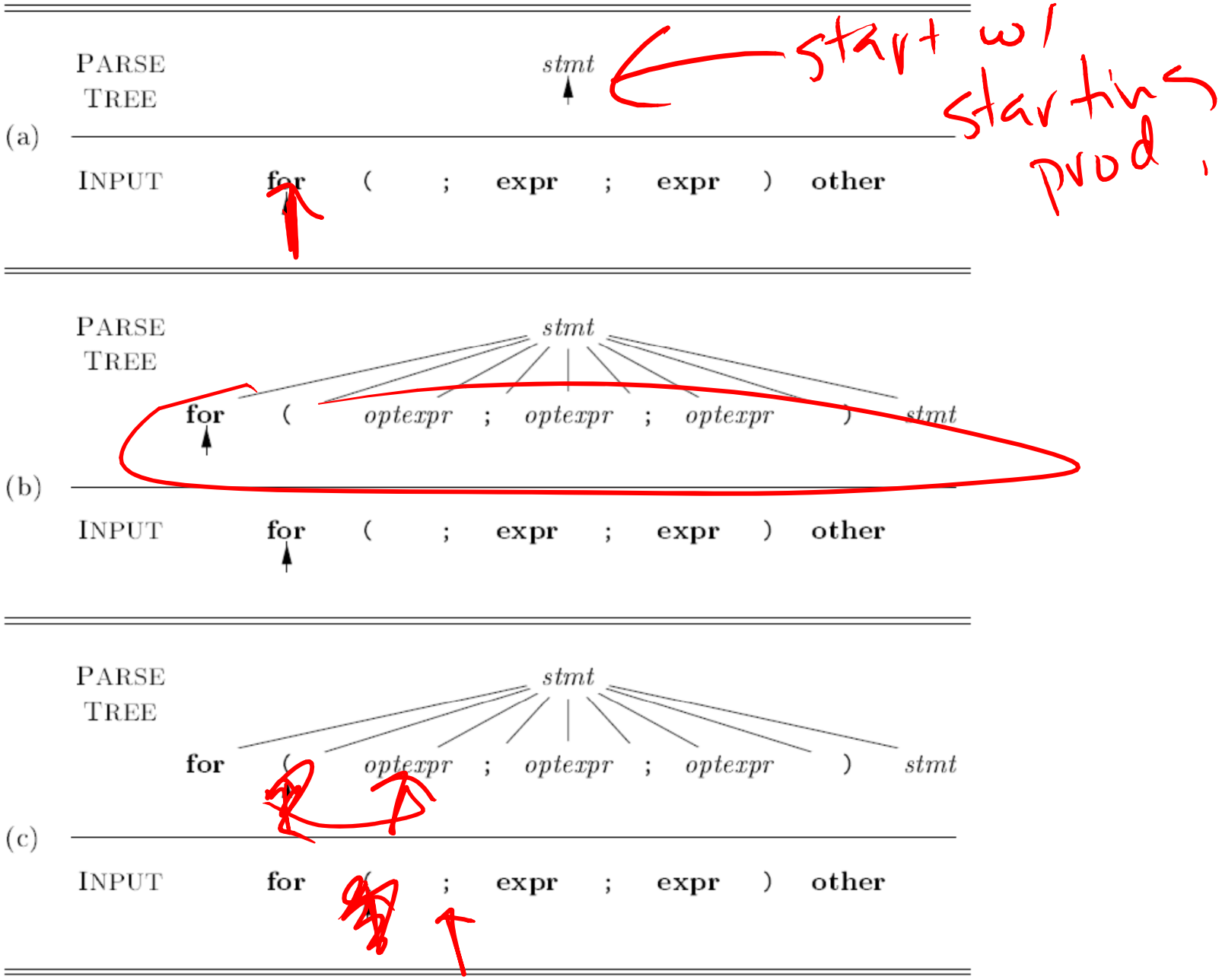


Figure 2.18: Top-down parsing while scanning the input from left to right

```
void stmt() {
        switch ( lookahead ) {
        case expr:
                match(expr); match(';'); break;
        case if:
                match(if); match('('); match(expr); match(')'); stmt();
                break;
        case for:
                match(for); match('(');
                optexpr(); match(';'); optexpr(); match(';'); optexpr();
                match(')'); stmt(); break;
        case other;
                match(other); break;
        default:
                report("syntax error");
        }
}

void optexpr() {
        if ( lookahead == expr ) match(expr);
}

void match(terminal t) {
        if ( lookahead == t ) lookahead = nextTerminal;
        else report("syntax error");
}
```

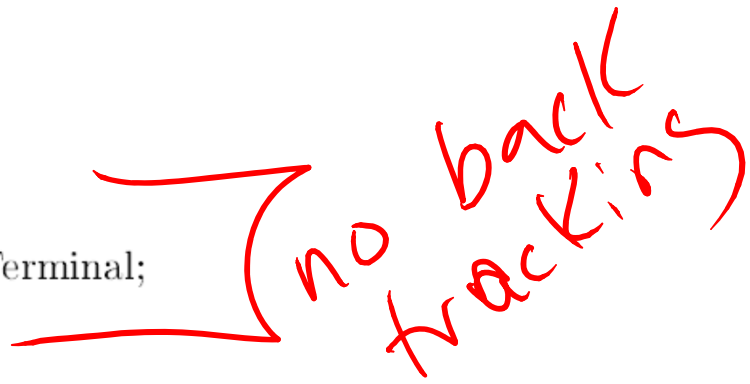


Figure 2.19: Pseudocode for a predictive parser

Predictive Parsing

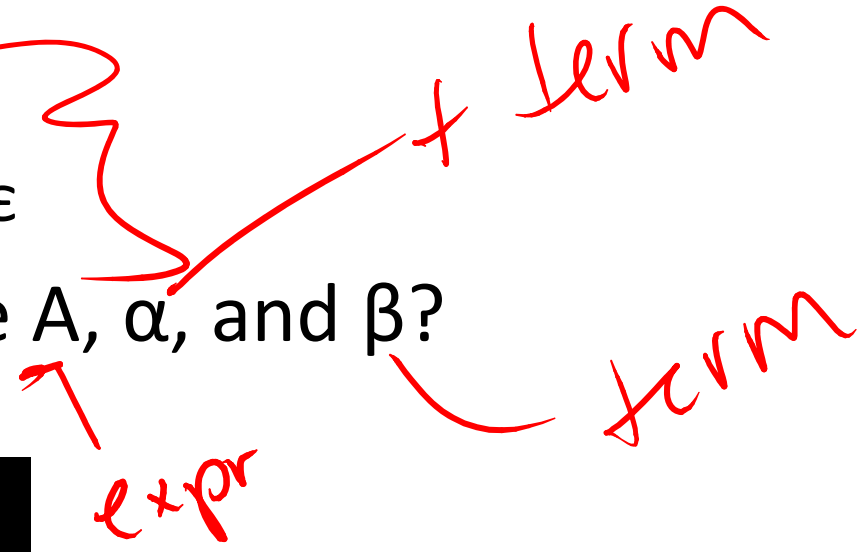
- Relies on:
  - α is string in grammar
  - FIRST(α) is set of terminals that appear first
  - If α generates ε, then ε is in FIRST(α)
  - If A-> α | β , then FIRST(α) and FIRST(β) disjoint
- Example:
  - FIRST(*stmt*) = {**expr, if, for, other**}
  - FIRST(**expr ;**) = {**expr**}

Looping Forever

- Left Recursion
- Rewrite:
  - *expr -> expr + term | term*
  - *A -> Aα | β*

  Vs.

  - *A -> βR*
  - *R -> αR | ε*
- What is the A, α, and β?



Infix to Postfix/What's wrong?

$$
\begin{array}{lcll}
expr & \rightarrow & expr + term & \{ \text{ print}('+') \ \} \\
 & | & expr - term & \{ \text{ print}('-') \ \} \\
 & | & term & \\
 & & & \\
term & \rightarrow & 0 & \{ \text{ print}('0') \ \} \\
 & | & 1 & \{ \text{ print}('1') \ \} \\
 & & \ldots & \\
 & | & 9 & \{ \text{ print}('9') \ \}
\end{array}
$$

Figure 2.21: Actions for translating into postfix notation

How is this different/same?

$$
\begin{array}{lcl}
expr & \rightarrow & term\ rest \\
rest & \rightarrow & +\ term\ \{\ \mathrm{print}('+')\ \}\ rest \\
 & | & -\ term\ \{\ \mathrm{print}('-')\ \}\ rest \\
 & | & \epsilon \\
term & \rightarrow & 0\ \{\ \mathrm{print}('0')\ \} \\
 & | & 1\ \{\ \mathrm{print}('1')\ \} \\
 & & \ldots \\
 & | & 9\ \{\ \mathrm{print}('9')\ \}
\end{array}
$$

Figure 2.23: Translation scheme after left-recursion elimination

Reading/Assignments

- Continue Milestone 1
- Finish Chap. 2