Parsing

Professor: Suman Saha

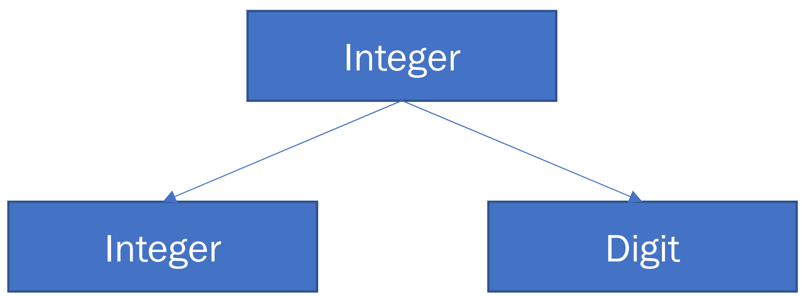
Parse Trees

- Derivation in graphical form
- The root node always contains the start symbol
- Each internal node has as its direct descendants the elements that appear on the right-hand side grammar
- The leaves of the parse tree are always terminal symbol

Parse Trees

- Derivation in graphical form
- The root node always contains the start symbol
- Each internal node has as its direct descendants the elements that appear on the right-hand side grammar
- The leaves of the parse tree are always terminal symbol

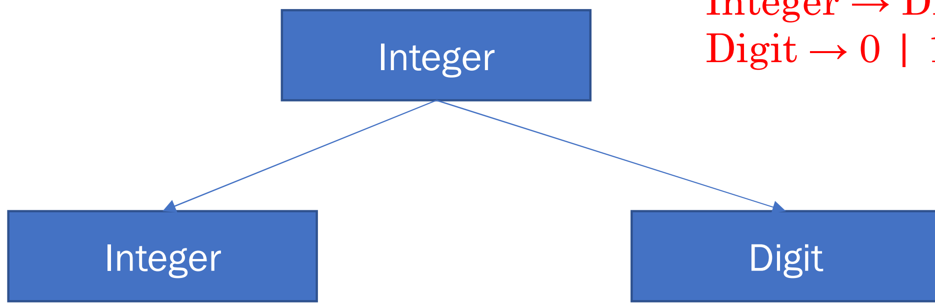
$$\text{Integer} \rightarrow \text{Integer Digit}$$



Parse Trees

Parse Tree of 352

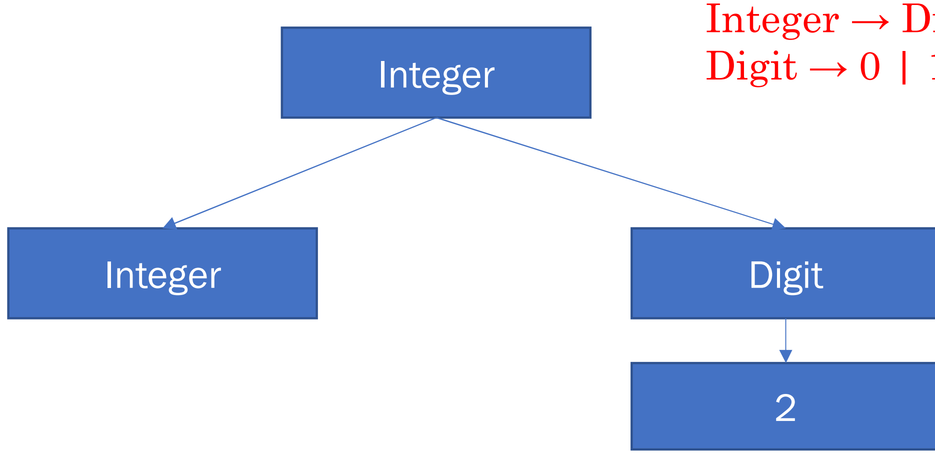
$\text{Integer} \rightarrow \text{Digit} \mid \text{Integer Digit}$
$\text{Digit} \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$



Parse Trees

Parse Tree of 352

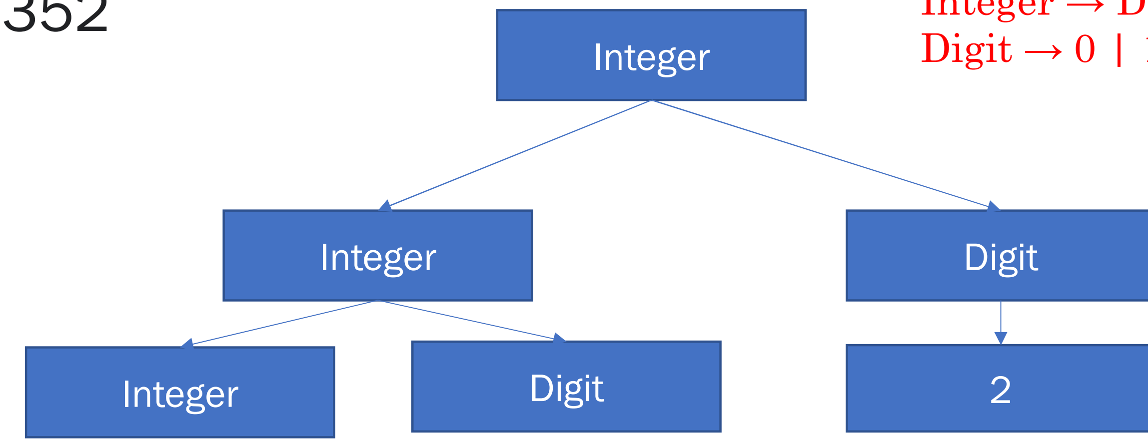
Integer → Digit | Integer Digit
Digit → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9



Parse Trees

Parse Tree of 352

Integer → Digit | Integer Digit
Digit → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

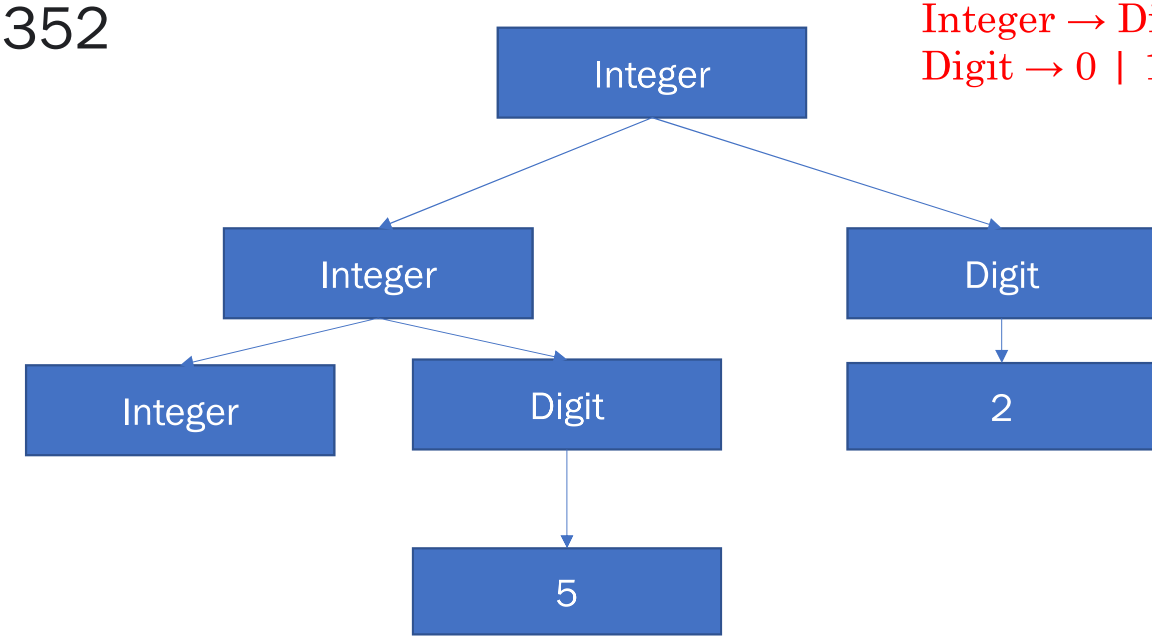


Parse Trees

PennState

Parse Tree of 352

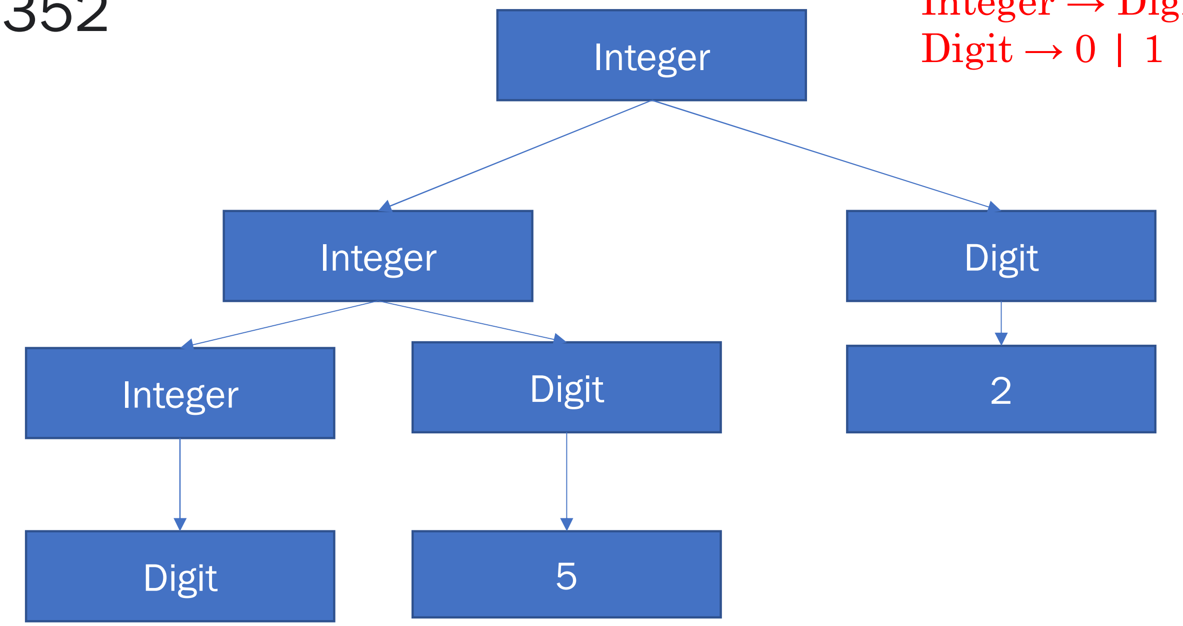
$\text{Integer} \rightarrow \text{Digit} \mid \text{Integer Digit}$
$\text{Digit} \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$



Parse Trees

Parse Tree of 352

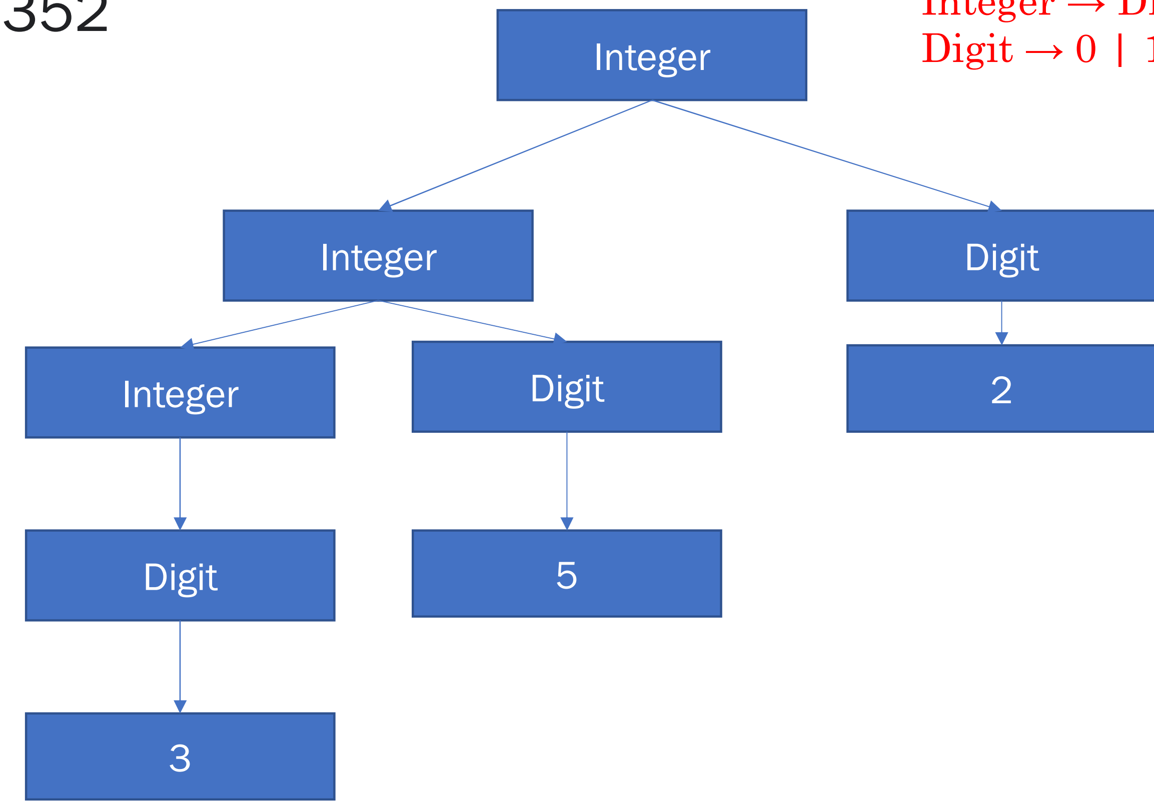
$$\text{Integer} \rightarrow \text{Digit} \mid \text{Integer Digit}$$
$$\text{Digit} \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$$



Parse Trees

Parse Tree of 352

$Integer \rightarrow Digit \mid Integer\ Digit$
$Digit \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$



Parse Trees

- Given grammar

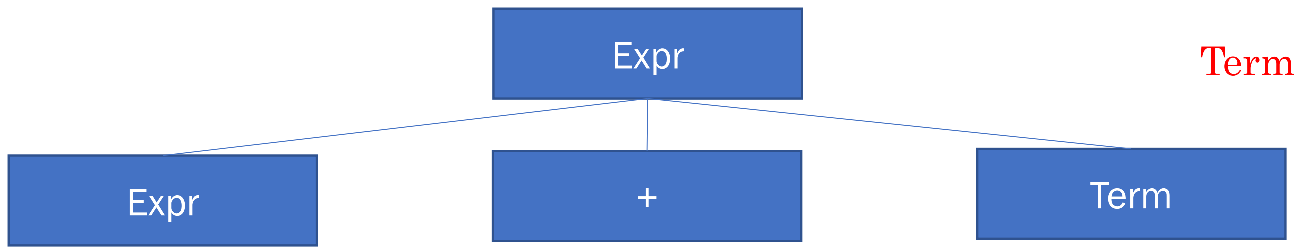
$$
\begin{aligned}
\text{Expr} \rightarrow \quad & \text{Expr} + \text{Term} \\
\mid\ & \text{Expr} - \text{Term} \\
\mid\ & \text{Term}
\end{aligned}
$$

$$\text{Term} \rightarrow \ 0 \mid \ldots \mid 9 \mid (\text{Expr})$$

Parse Trees

Parse of the string $5 - 4 + 3$

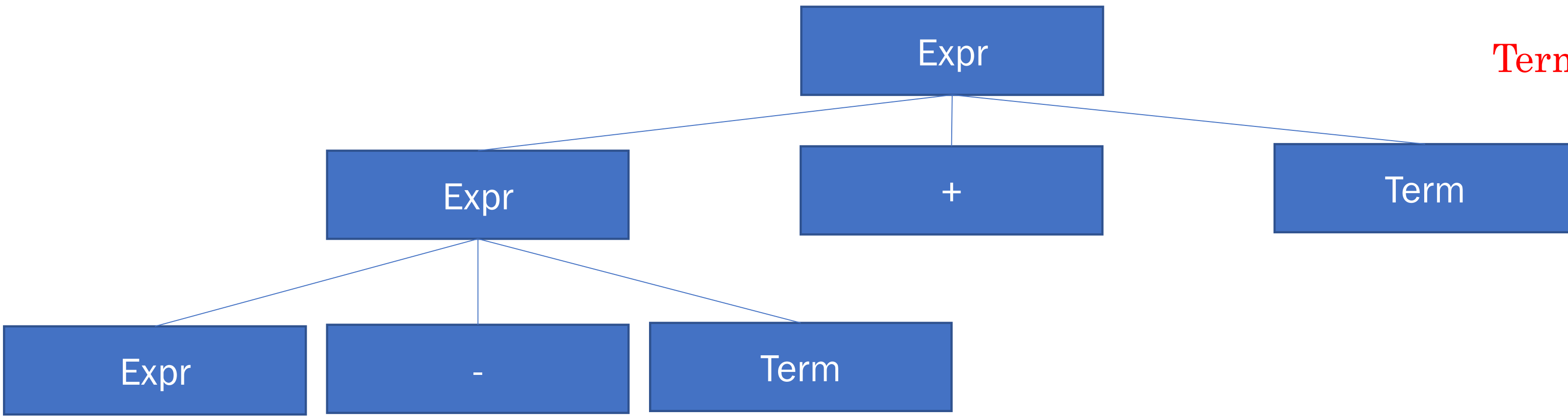
$$\begin{aligned} Expr \rightarrow\ & Expr + Term \\ |\ & Expr - Term \\ |\ & Term \\ Term \rightarrow\ & 0 \mid \ldots \mid 9 \mid (Expr) \end{aligned}$$



Parse Trees

Parse of the string $5 - 4 + 3$

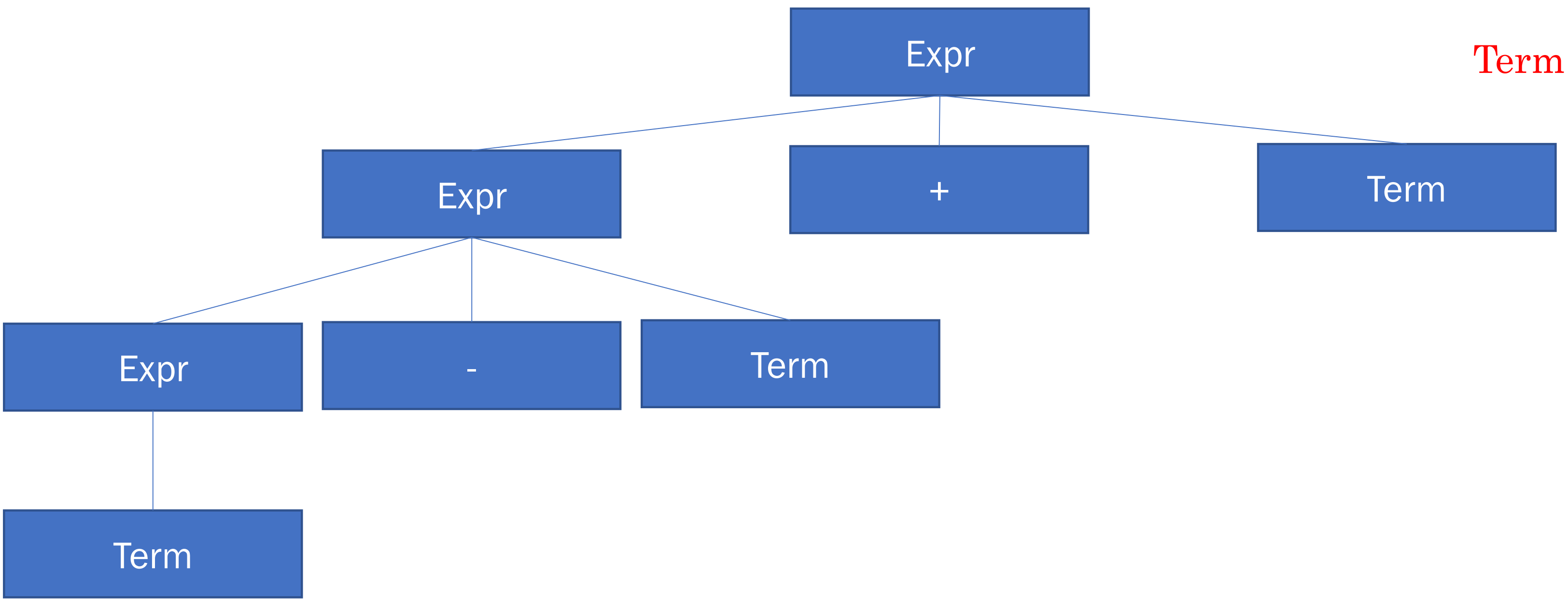
$$
\begin{aligned}
\mathrm{Expr} \rightarrow\ & \mathrm{Expr} + \mathrm{Term} \\
\mid\ & \mathrm{Expr} - \mathrm{Term} \\
\mid\ & \mathrm{Term} \\
\mathrm{Term} \rightarrow\ & 0 \mid \ldots \mid 9 \mid (\mathrm{Expr})
\end{aligned}
$$



Parse Trees

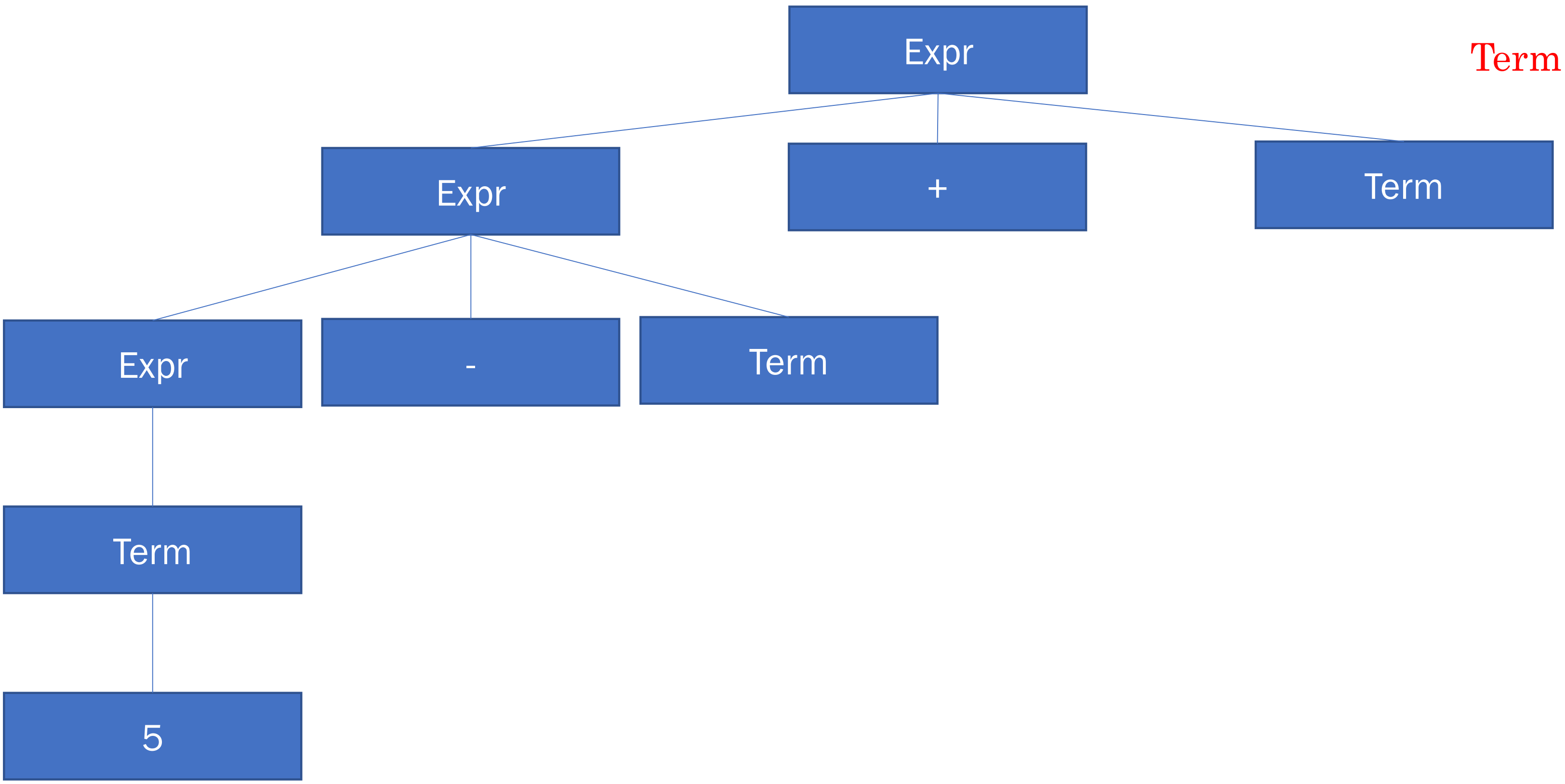
Parse of the string $5 - 4 + 3$

$$
\begin{aligned}
\text{Expr} \rightarrow \quad & \text{Expr} + \text{Term} \\
| \ & \text{Expr} - \text{Term} \\
| \ & \text{Term} \\
\text{Term} \rightarrow \quad & 0 \mid \ldots \mid 9 \mid (\text{Expr})
\end{aligned}
$$



Parse Trees

Parse of the string $5 - 4 + 3$

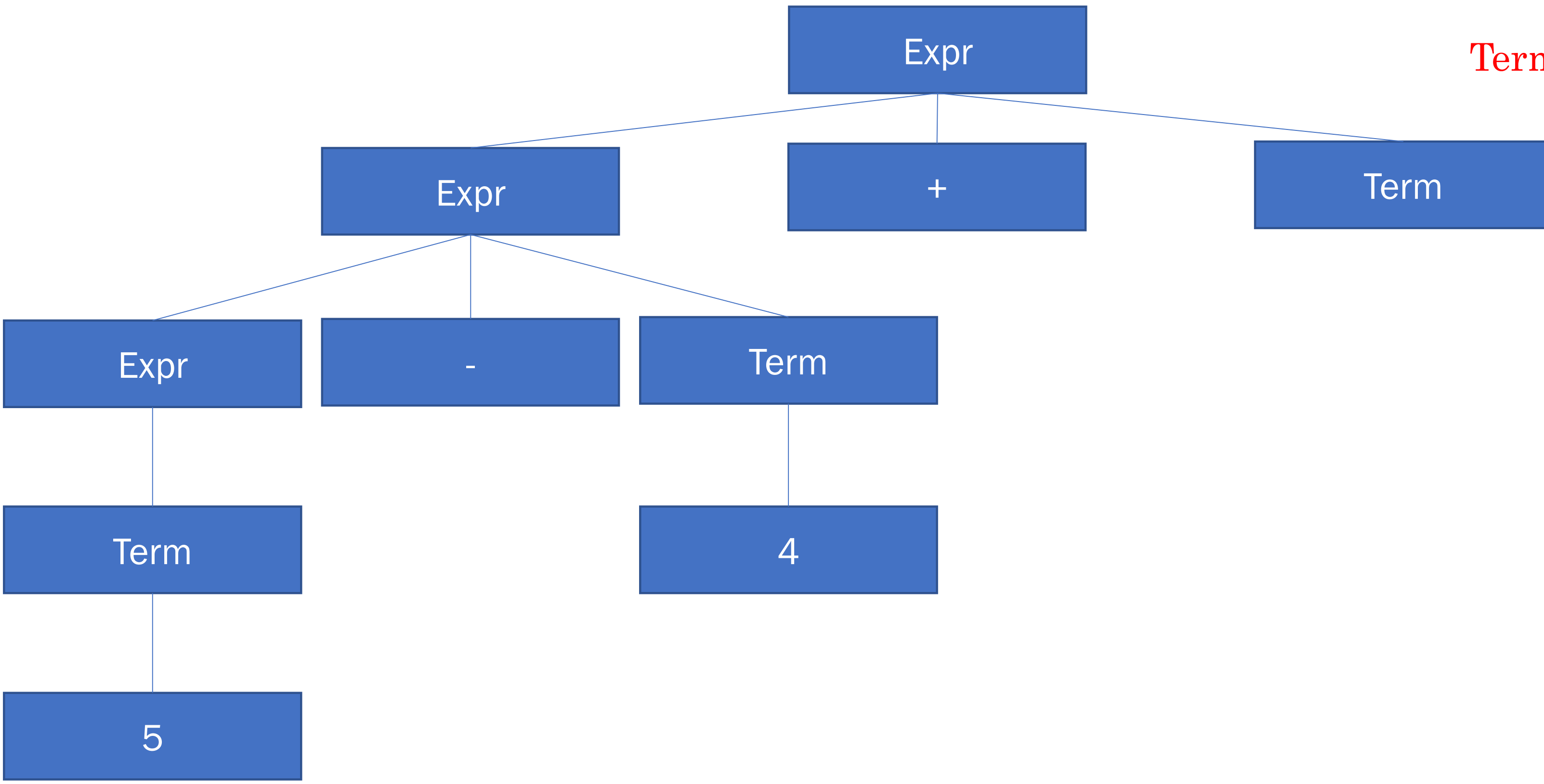


$$
\begin{aligned}
\text{Expr} \rightarrow\; & \text{Expr} + \text{Term} \\
& \mid\ \text{Expr} - \text{Term} \\
& \mid\ \text{Term} \\
\text{Term} \rightarrow\; & 0 \mid \ldots \mid 9 \mid (\text{Expr})
\end{aligned}
$$

Parse Trees

Parse of the string $5 - 4 + 3$

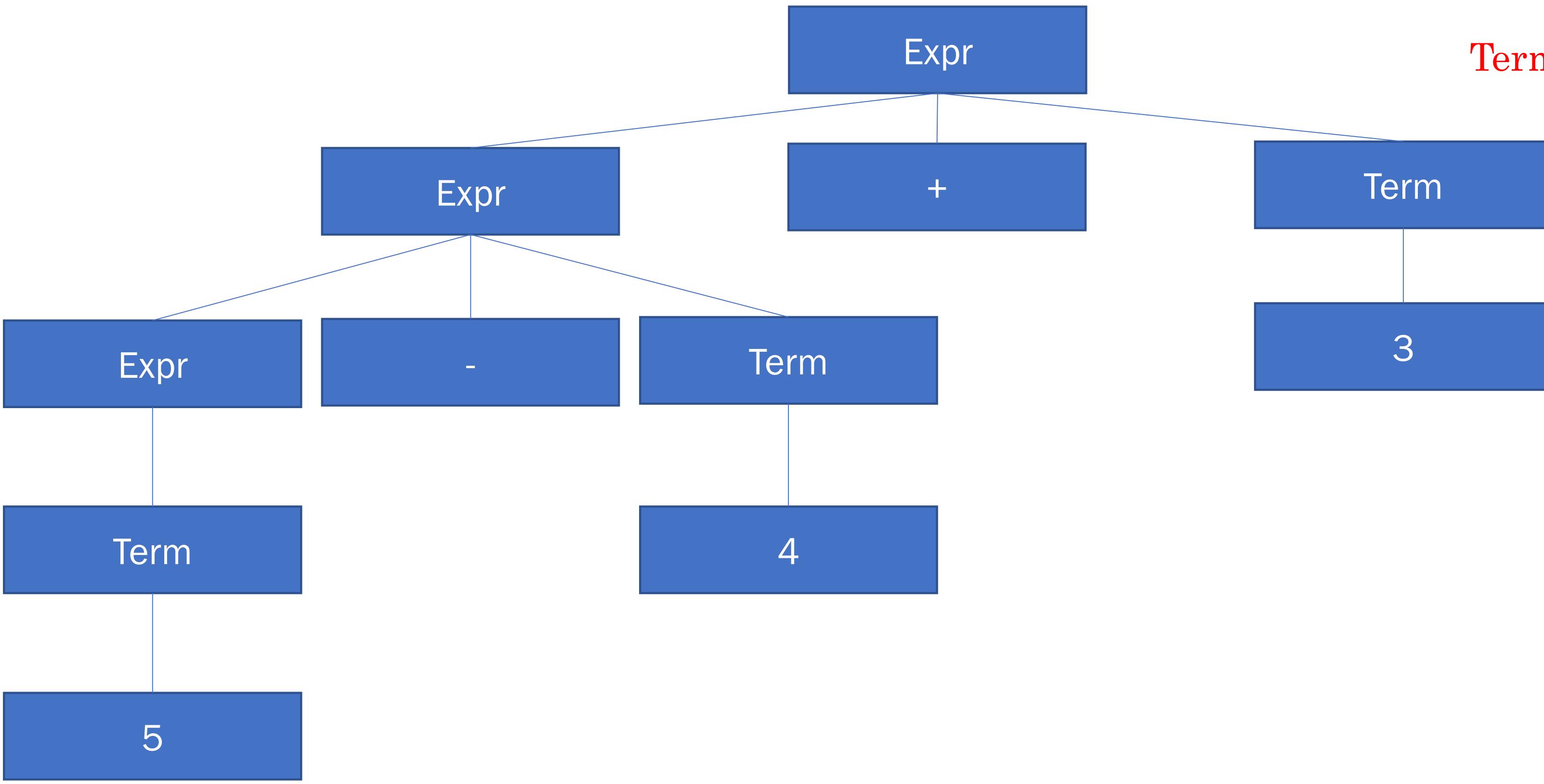
$$
\begin{aligned}
\text{Expr} \rightarrow\ & \text{Expr} + \text{Term} \\
\mid\ & \text{Expr} - \text{Term} \\
\mid\ & \text{Term} \\
\text{Term} \rightarrow\ & 0 \mid \ldots \mid 9 \mid (\text{Expr})
\end{aligned}
$$



Parse Trees

Parse of the string $5 - 4 + 3$

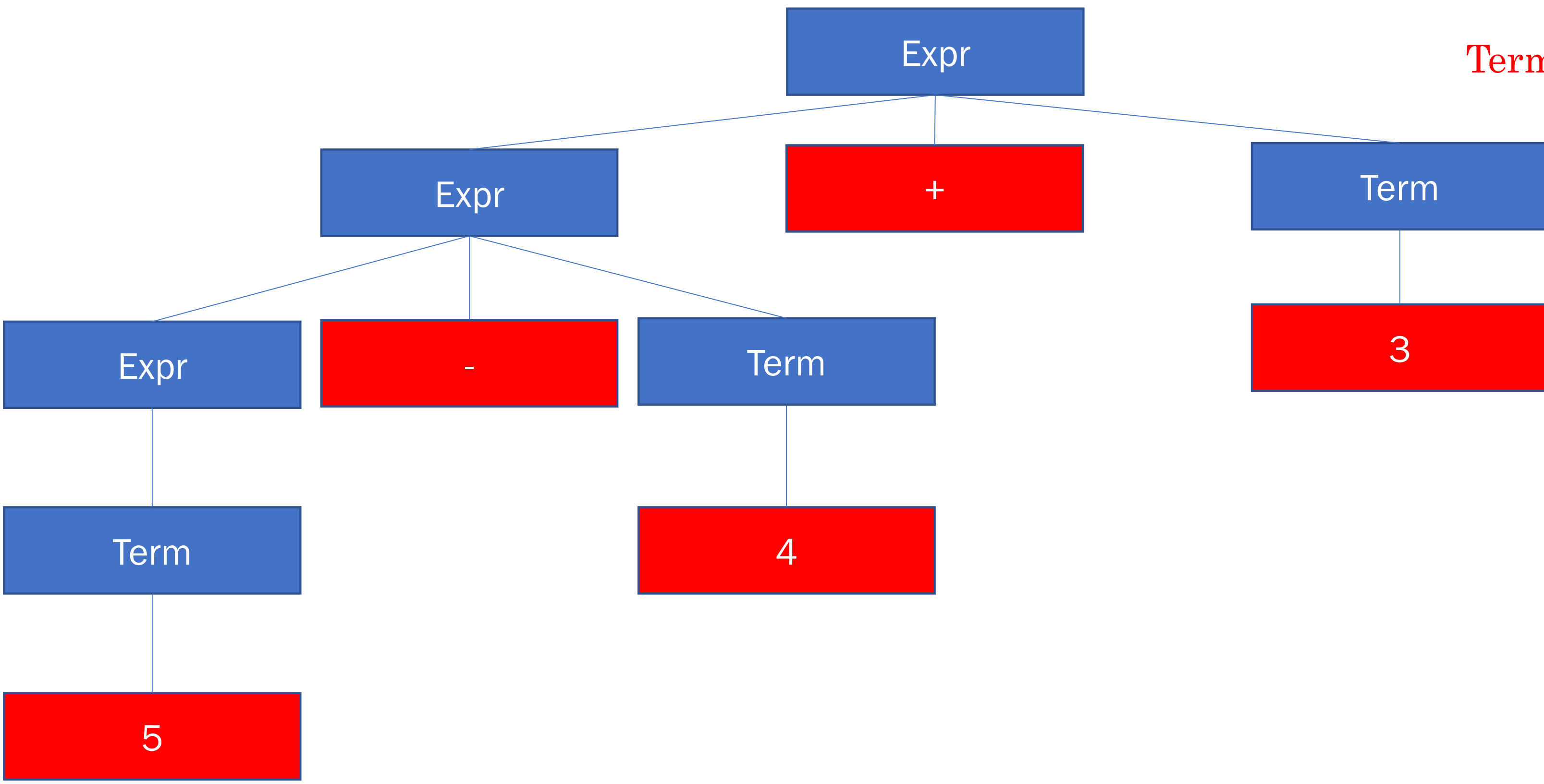
$$
\begin{aligned}
\text{Expr} \rightarrow \quad & \text{Expr} + \text{Term} \\
| \ & \text{Expr} - \text{Term} \\
| \ & \text{Term} \\
\text{Term} \rightarrow \quad & 0 \mid \ldots \mid 9 \mid (\text{Expr})
\end{aligned}
$$



Parse Trees

Parse of the string $5 - 4 + 3$

$$
\begin{aligned}
\text{Expr} \rightarrow\ & \text{Expr} + \text{Term} \\
\mid\ & \text{Expr} - \text{Term} \\
\mid\ & \text{Term} \\
\text{Term} \rightarrow\ & 0 \mid \ldots \mid 9 \mid (\text{Expr})
\end{aligned}
$$



$$(5\text{-}4)+3 \neq 5 - (4 + 3)$$

Ambiguous Grammar

- A grammar is ambiguous if, for any string
  - it has more than one parse tree, or
  - there is more than one right-most derivation, or
  - there is more than one left-most derivation

  (the three conditions are equivalent)

Ambiguous Grammar

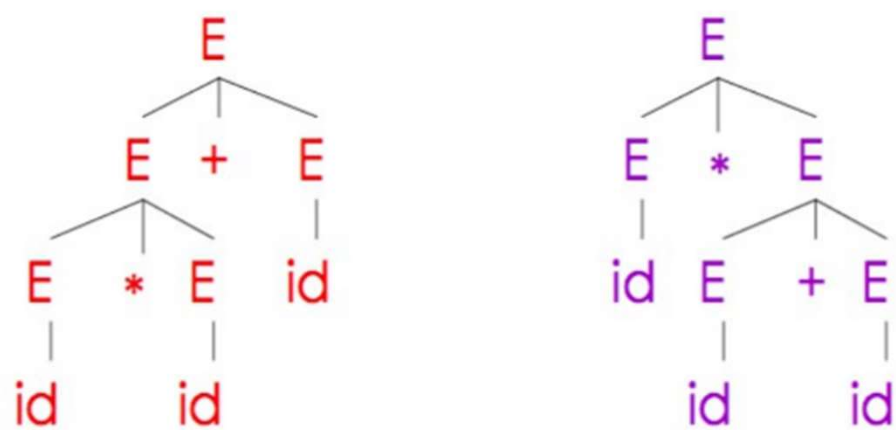
- A grammar is ambiguous if, for any string
  - it has more than one parse tree, or
  - there is more than one right-most derivation, or
  - there is more than one left-most derivation

  (the three conditions are equivalent)

- Unambiguous grammar is preferred
- However, ambiguity may tolerable
- Tradeoff between the size of the grammar and the information it is trying to convey

Ambiguity

- Ambiguity = Program structure is not uniquely defined
- $\mathrm{E} \rightarrow \mathrm{E} + \mathrm{E} \mid \mathrm{E} * \mathrm{E} \mid (\mathrm{E}) \mid \mathrm{id}$
- String $\mathrm{id} * \mathrm{id} + \mathrm{id}$ has two parse trees:



Dangling Else Problem

Given grammar

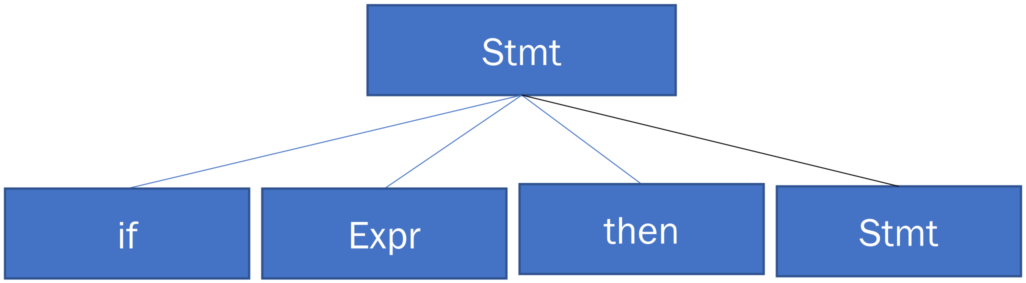
$$\begin{aligned} \text{Stmt} \rightarrow\ & \text{if Expr then Stmt} \\ \mid\ & \text{if Expr then Stmt else Stmt} \\ \mid\ & \text{other} \end{aligned}$$

Example:

if E1 then

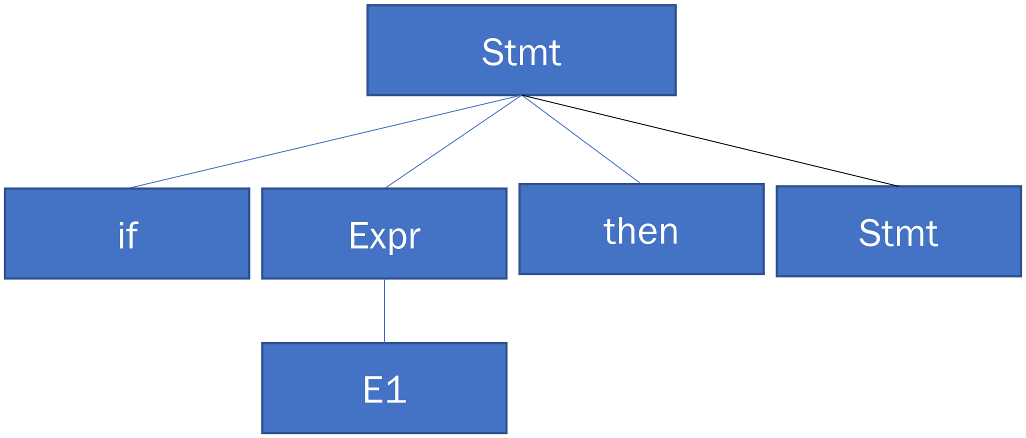
    if E2 then S1 else S2

Dangling Else Problem




```
if E1 then
   if E2 then S1 else S2
```

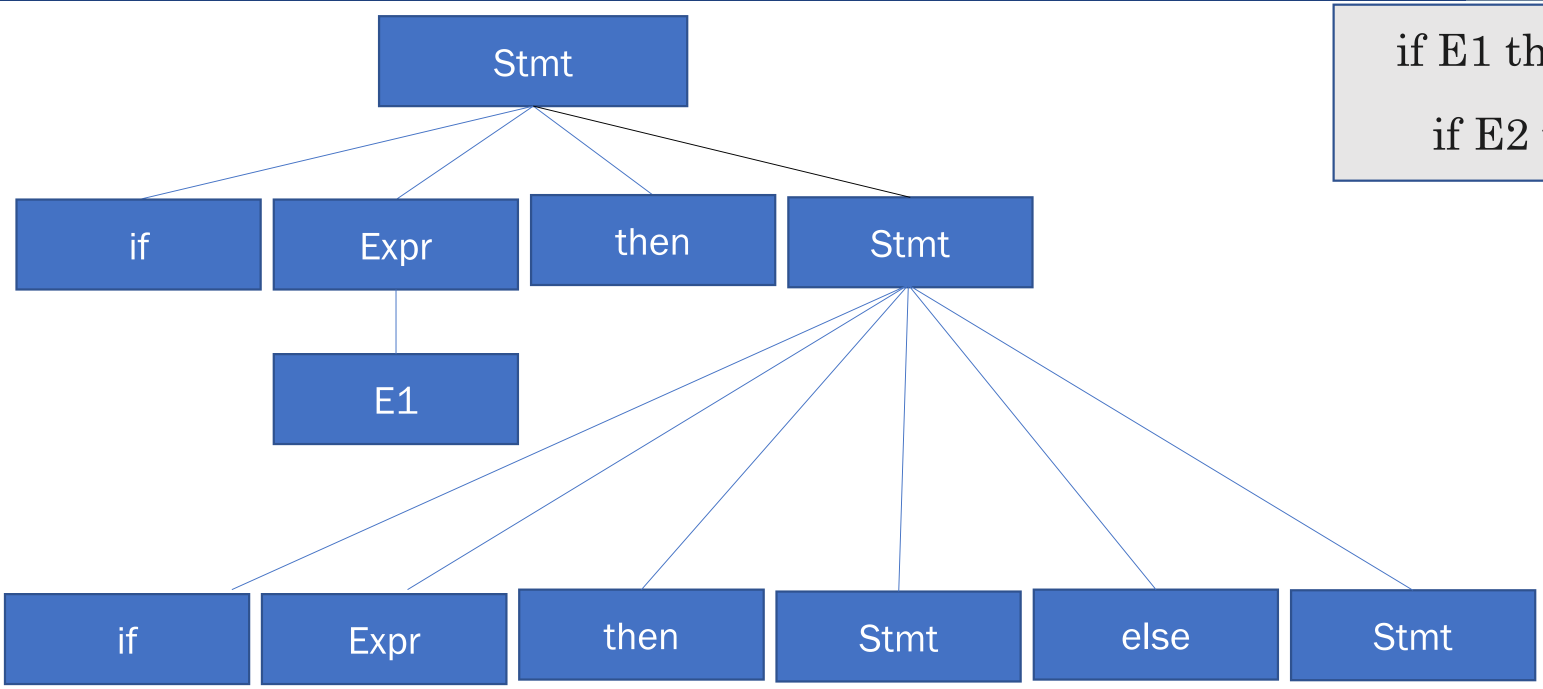
Dangling Else Problem



if E1 then

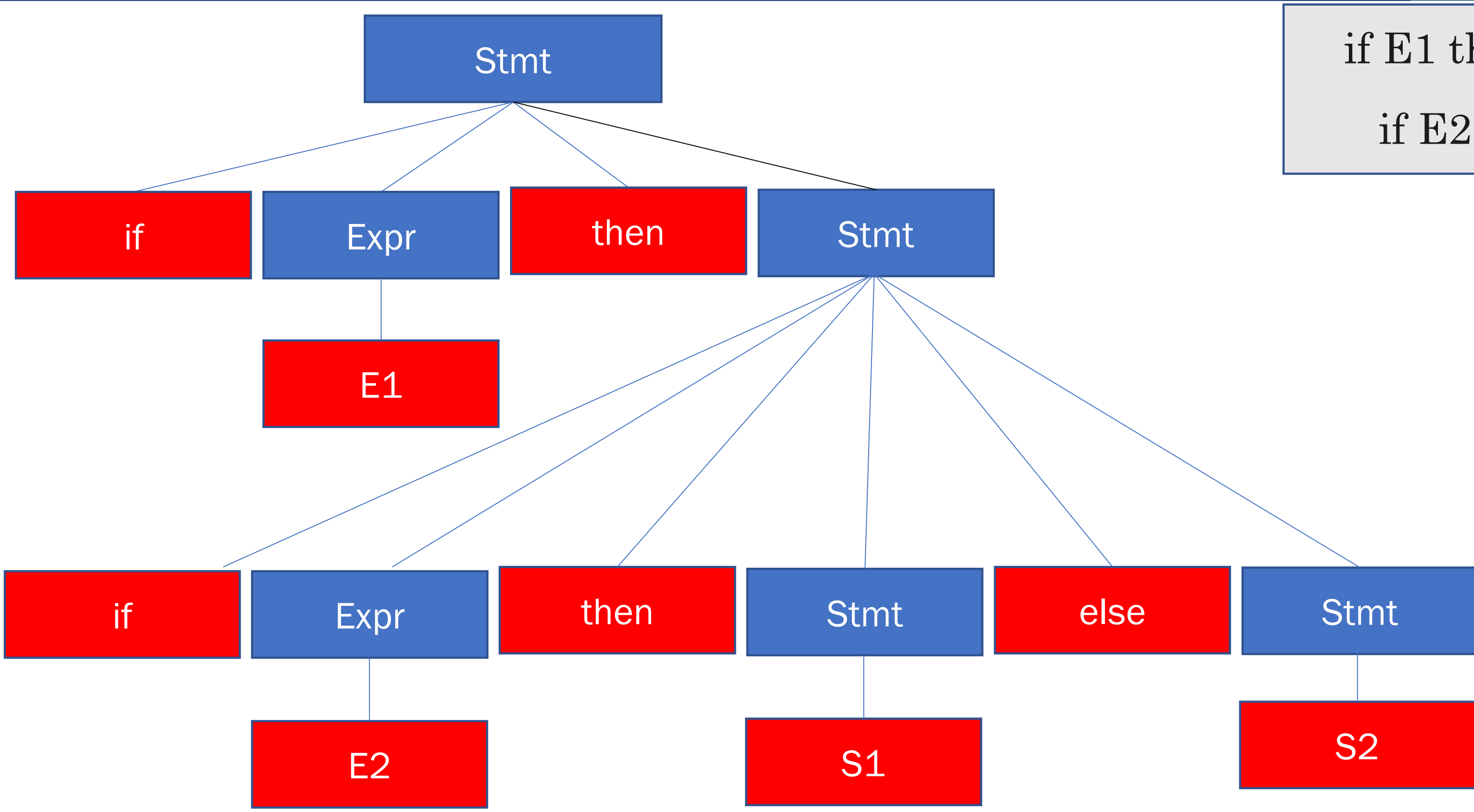
if E2 then S1 else S2

Dangling Else Problem




```
if E1 then
   if E2 then S1 else S2
```

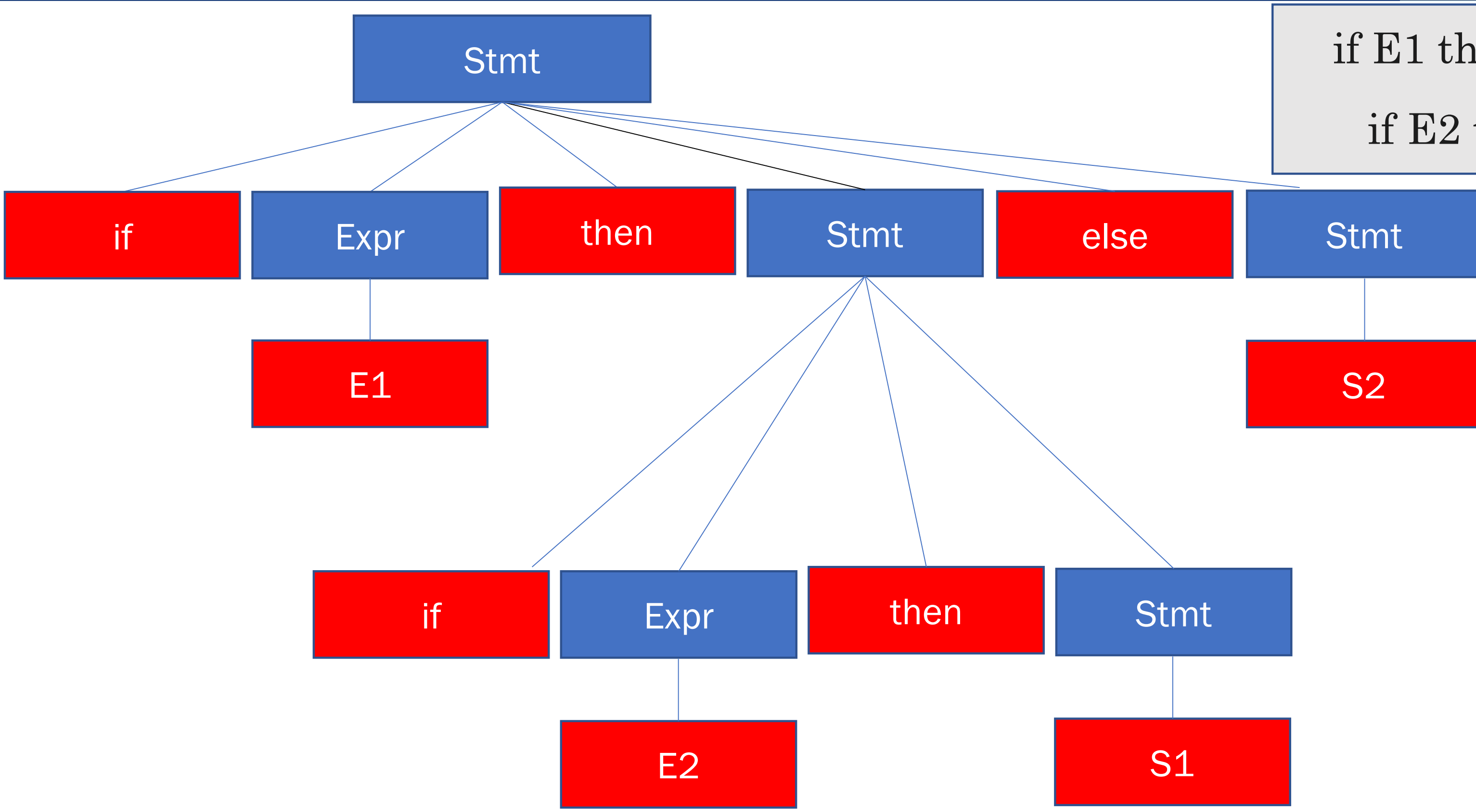
Dangling Else Problem



if E1 then

if E2 then S1 else S2

Dangling Else Problem



if E1 then

    if E2 then S1 else S2

Dealing with Ambiguity

- There are several ways to handle ambiguity
- We will discuss one of them

- Rewriting the grammar

Precedence

Rewriting the grammar

- use a different nonterminal for each precedence
- Start with the lowest precedence

Precedence

Rewriting the grammar

- use a different nonterminal for each precedence
- Start with the lowest precedence

$$E \rightarrow E - E \mid E/ E \mid (E) \mid id$$

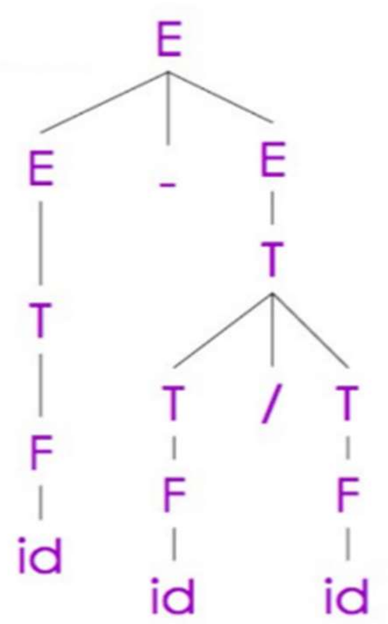
rewrite to

$$E \rightarrow E - E \mid T$$
$$T \rightarrow T / T \mid F$$
$$F \rightarrow id \mid (E)$$

Precedence

Parse tree for id – id / id

$$E \rightarrow E - E \mid T$$
$$T \rightarrow T / T \mid F$$
$$F \rightarrow id \mid (E)$$



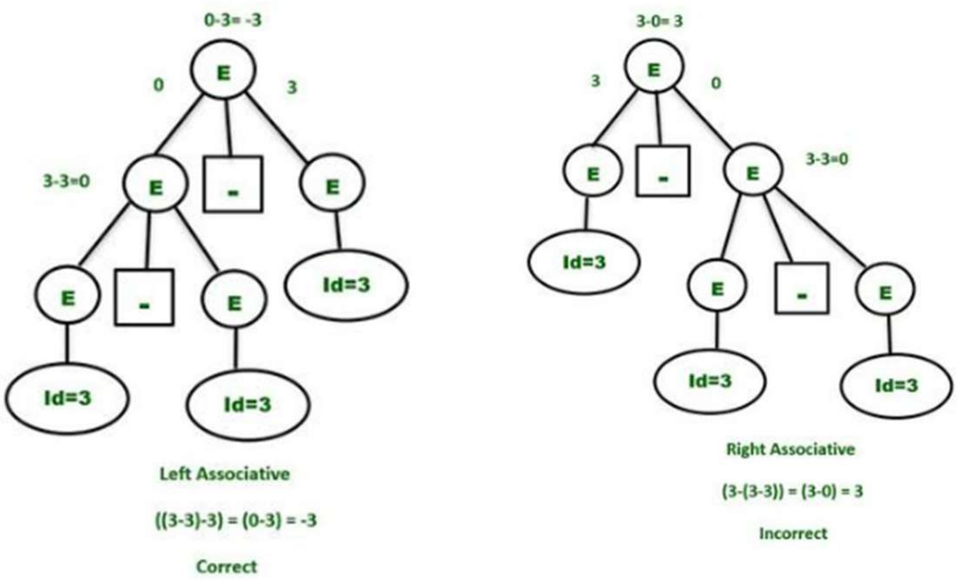
Associativity

- The grammar captures operator precedence, but it is still ambiguous!
  - fails to express that both subtraction and division are left associative;

$E \rightarrow E - E \mid id$

Parse tree for $id - id - id$.

Let's consider a single value of id = 3



Recursion

- Grammar is recursive in nonterminal X if:

  $$X \rightarrow^{+} \ldots X \ldots$$

  $\rightarrow^{+}$ means "in one or more steps, X derives a sequence of symbols that includes an X"

Recursion

- Grammar is recursive in nonterminal X if:

  $X \rightarrow^{+} \ldots X \ldots$

  $\rightarrow^{+}$ means "in one or more steps, X derives a sequence of symbols that includes an X"

- Grammar is left recursive in X if:

  $X \rightarrow^{+} X \ldots$

  In one or more steps, X derives a sequence of symbols that starts with an X

Recursion

- Grammar is recursive in nonterminal X if:

  $\mathrm{X} \rightarrow + \ldots \mathrm{X} \ldots$

  $\rightarrow +$ means "in one or more steps, X derives a sequence of symbols that includes an X"

- Grammar is left recursive in X if:

  $\mathrm{X} \rightarrow + \mathrm{X} \ldots$

  In one or more steps, X derives a sequence of symbols that starts with an X

- A grammar is right recursive in X if:

  $\mathrm{X} \rightarrow + \ldots \mathrm{X}$

  In one or more steps, X derives a sequence of symbols that ends with and X

How to Fix Associativity

- The grammar given above is both left and right recursive in non-terminals exp and term

- To correctly expresses operator associativity:
  - For left associativity, use left recursion
  - For right associativity, use right recursion

Here's the correct grammar

$$
\begin{aligned}
&\mathrm{E} \rightarrow \mathrm{E} - \mathrm{T} \mid \mathrm{T} \\
&\mathrm{T} \rightarrow \mathrm{T} / \mathrm{F} \mid \mathrm{F} \\
&\mathrm{F} \rightarrow \mathrm{id} \mid (\mathrm{E})
\end{aligned}
$$

Abstract Syntax Tree

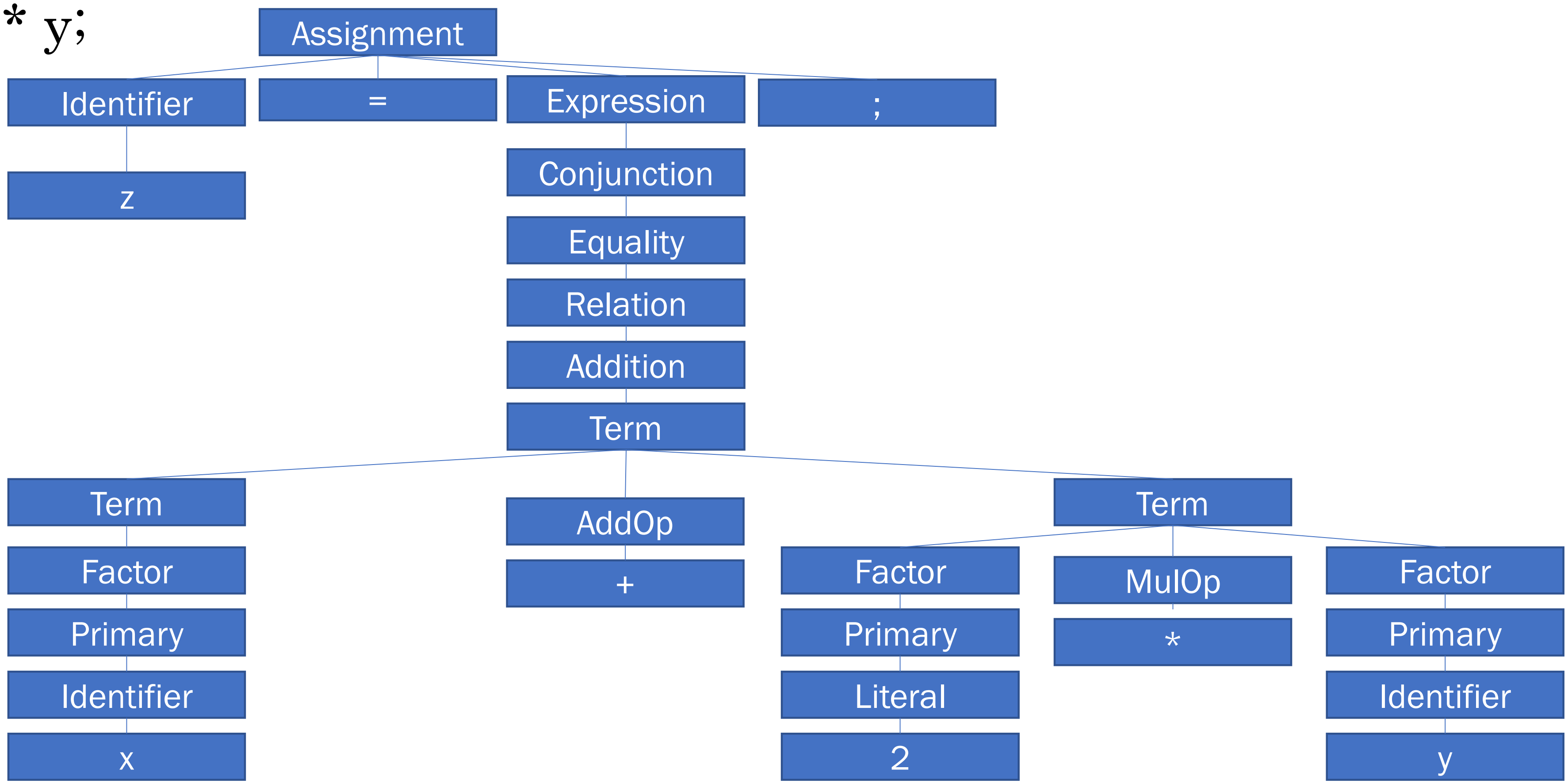
- An abstract syntax tree (AST) is a simplified version of a parse tree. An AST only contains information related to analyzing the source text and ignores extra syntactic information used for parsing text.
- Why do we need alternative?
  - Fewer intermediate nodes and subtrees
  - All information like parse tree but smaller

Transform Parse tree into AST

- Discard all the punctuation, such as semicolon
- Discard all nonterminal which are trivial roots, ones with only a single subtree
- Finally, replace the remaining non-terminals with the operators which are a leaf of one of their immediate subtrees.

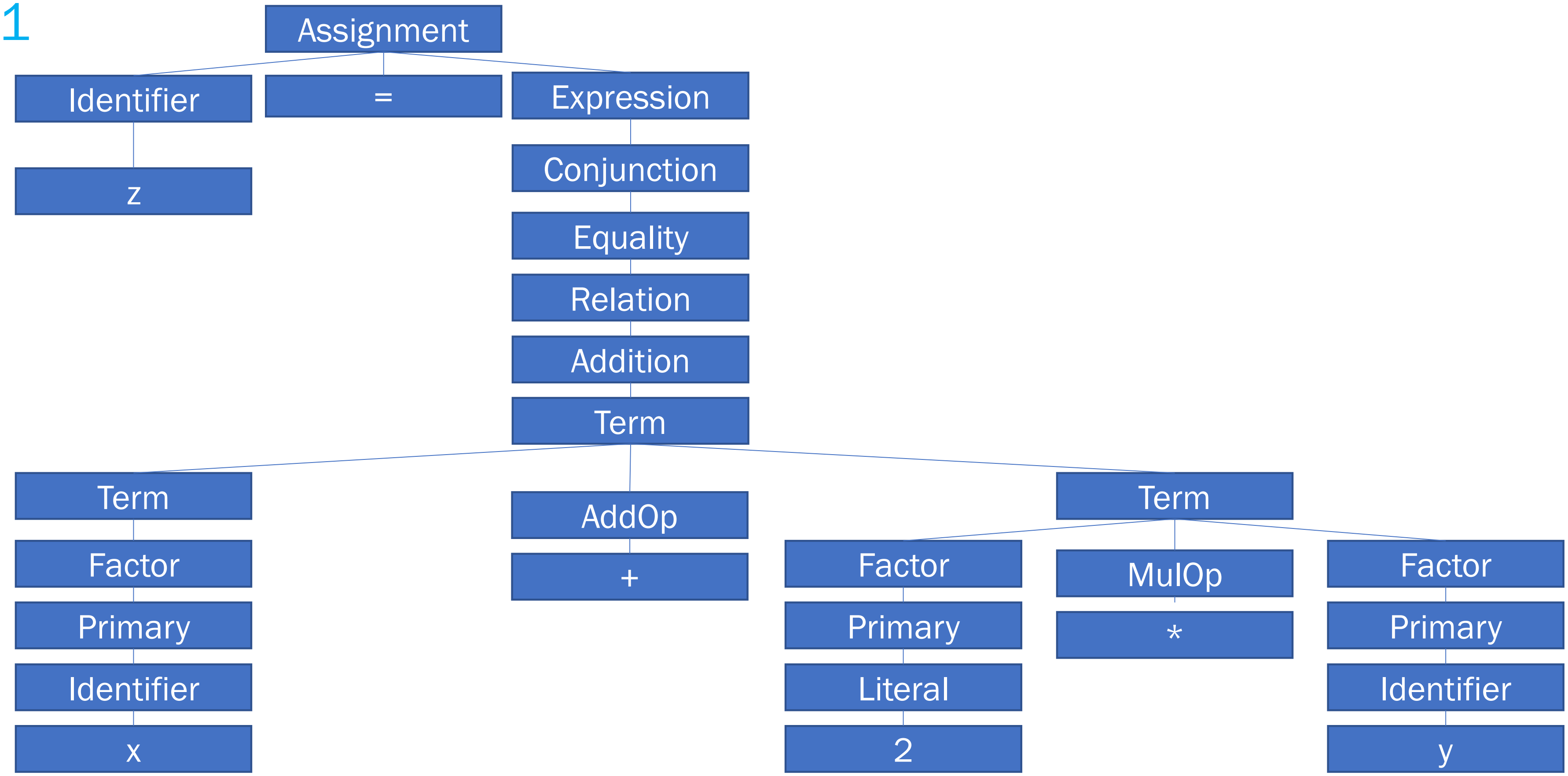
Example (Parse Tree)

- z = x + 2 * y;



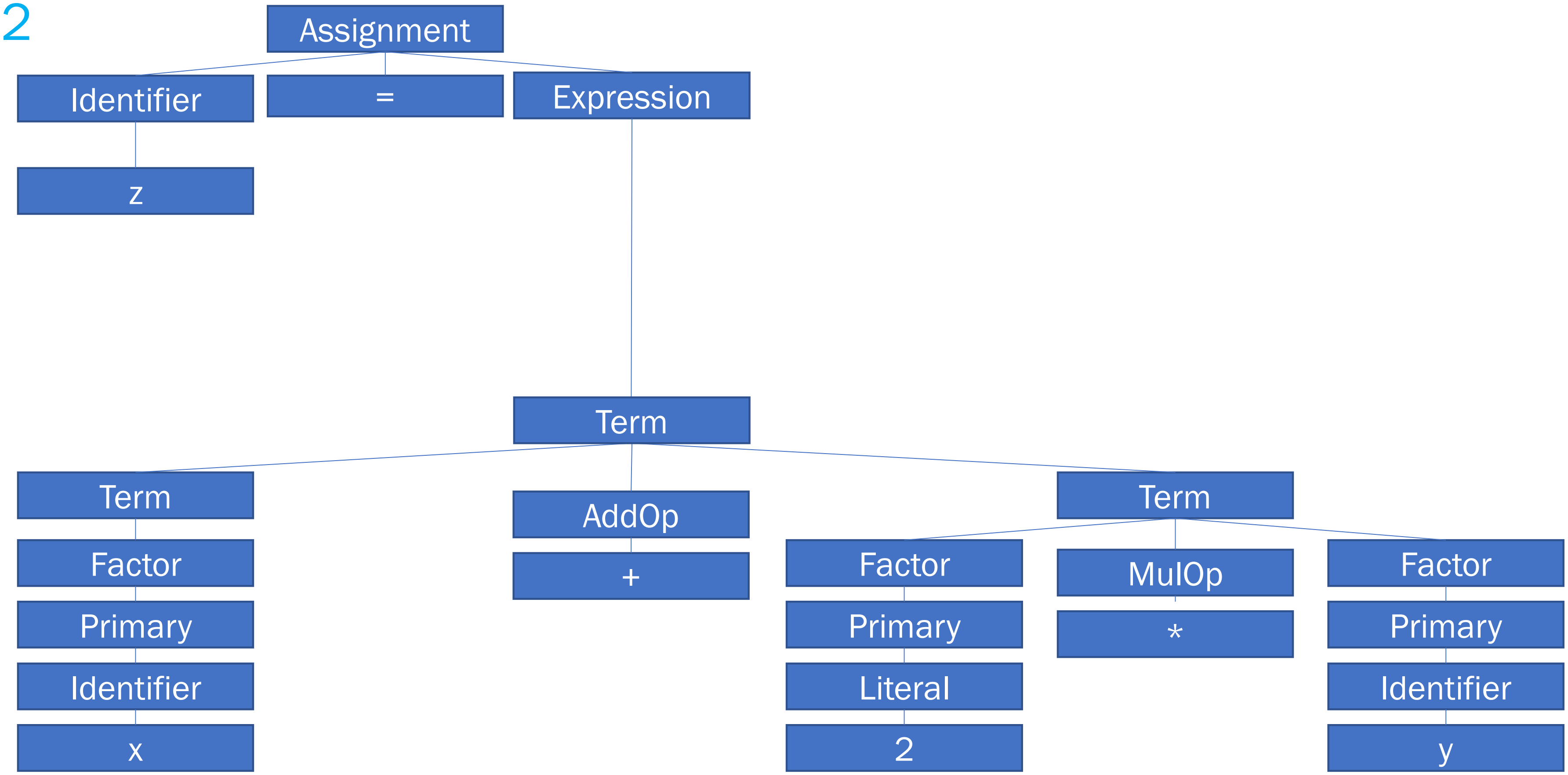
Example (Parse Tree)

Apply Rule 1



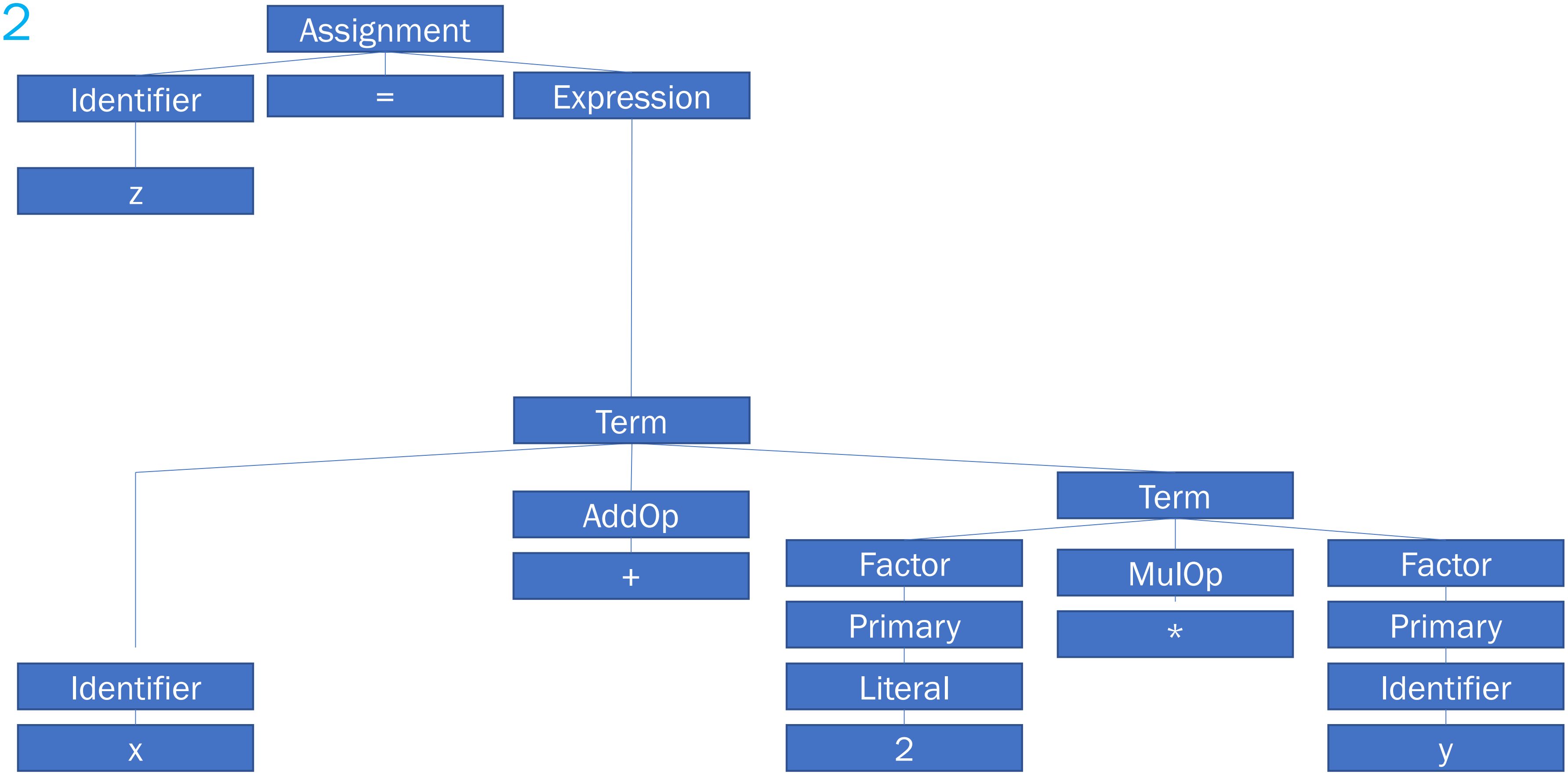
Example (Parse Tree)

Apply Rule 2



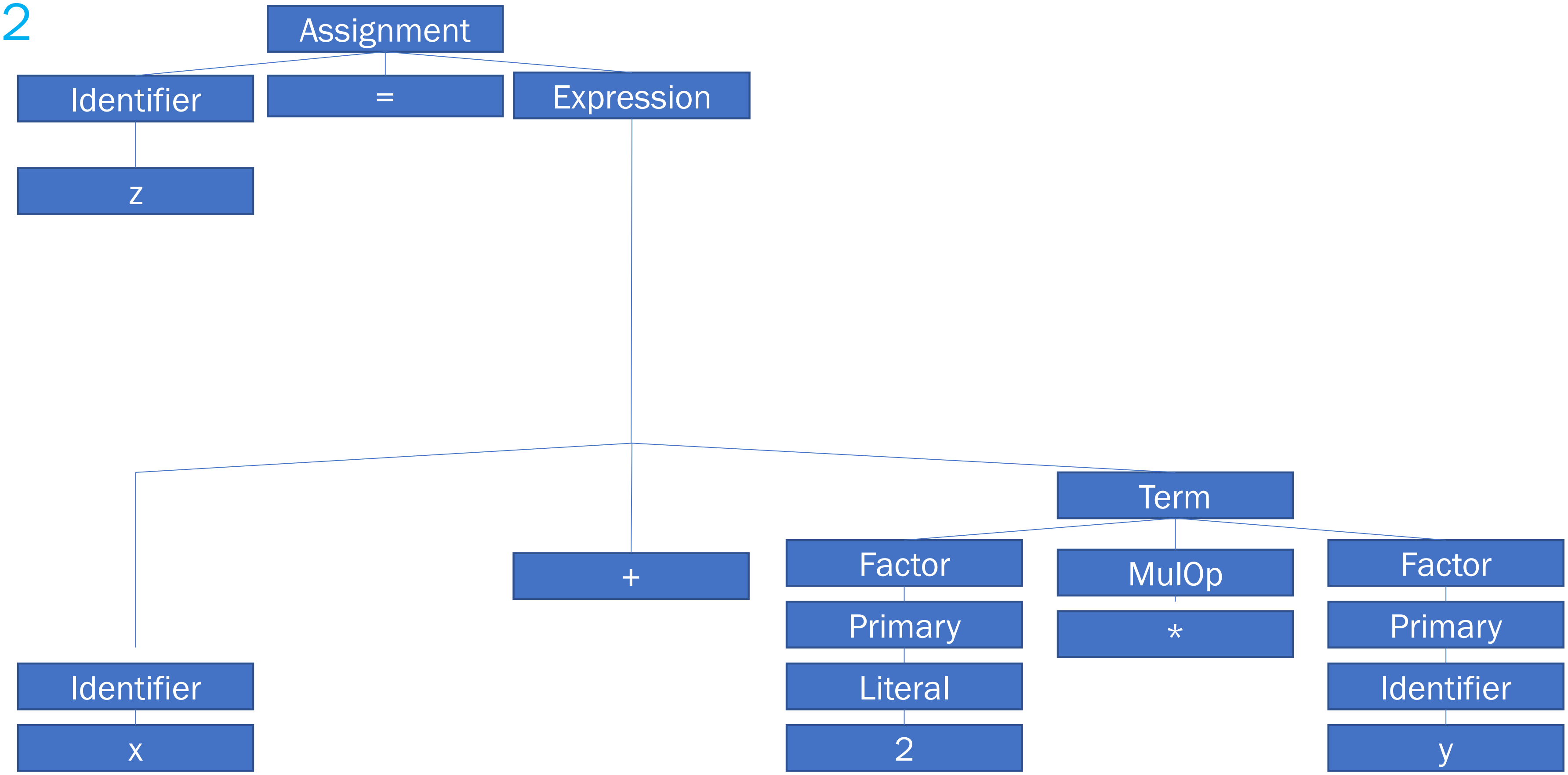
Example (Parse Tree)

Apply Rule 2



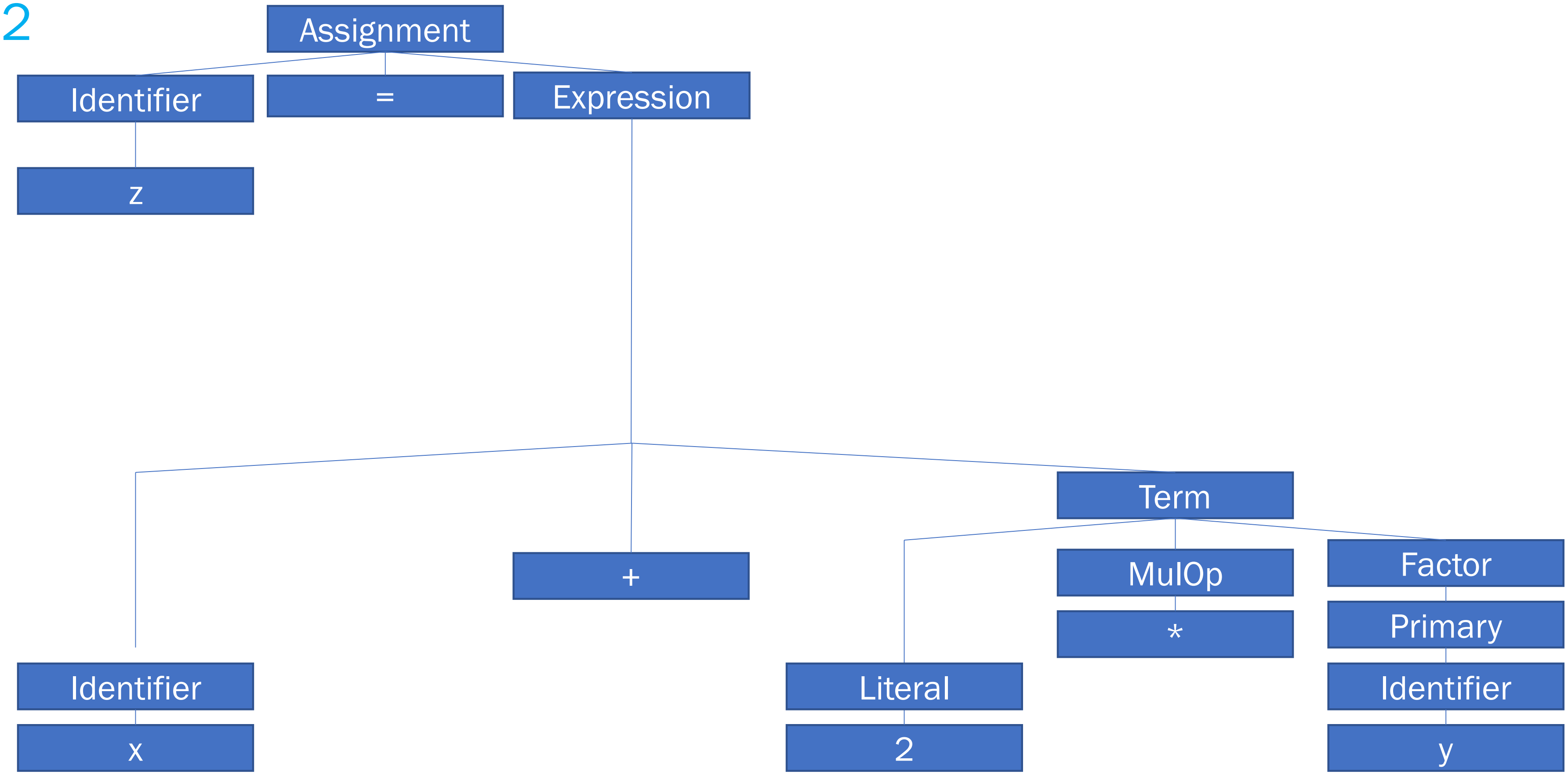
Example (Parse Tree)

Apply Rule 2



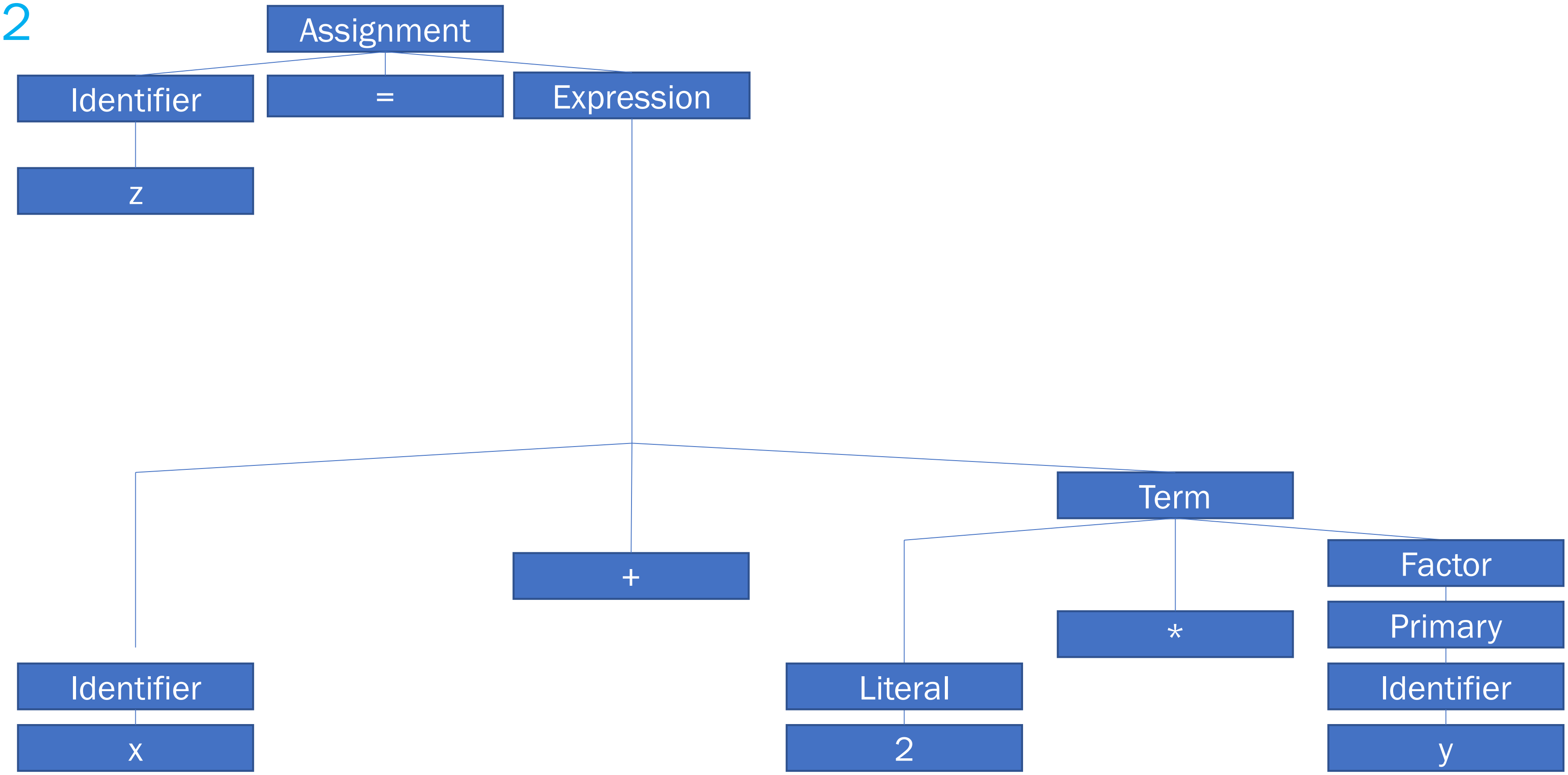
Example (Parse Tree)

Apply Rule 2



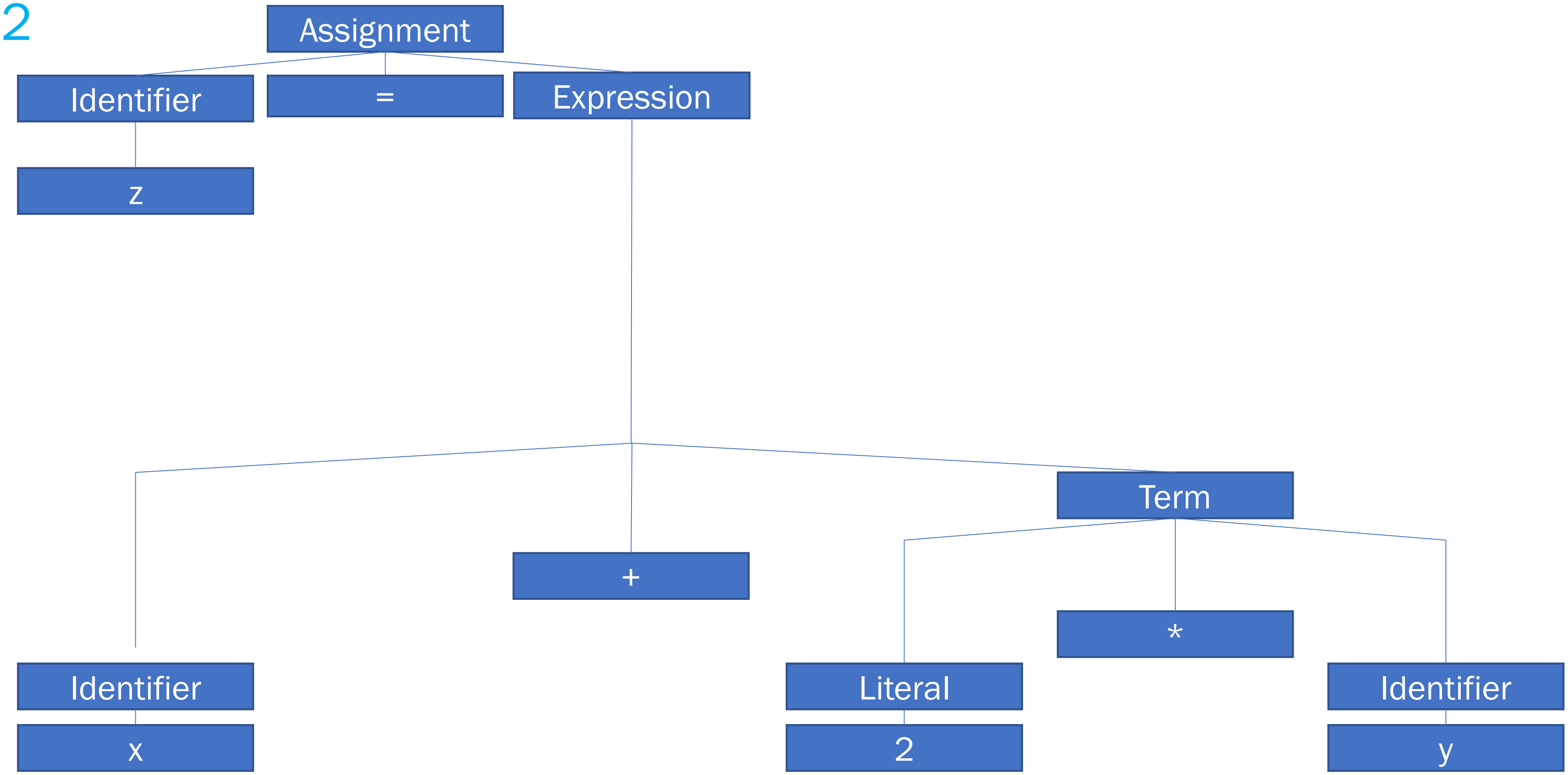
Example (Parse Tree)

Apply Rule 2



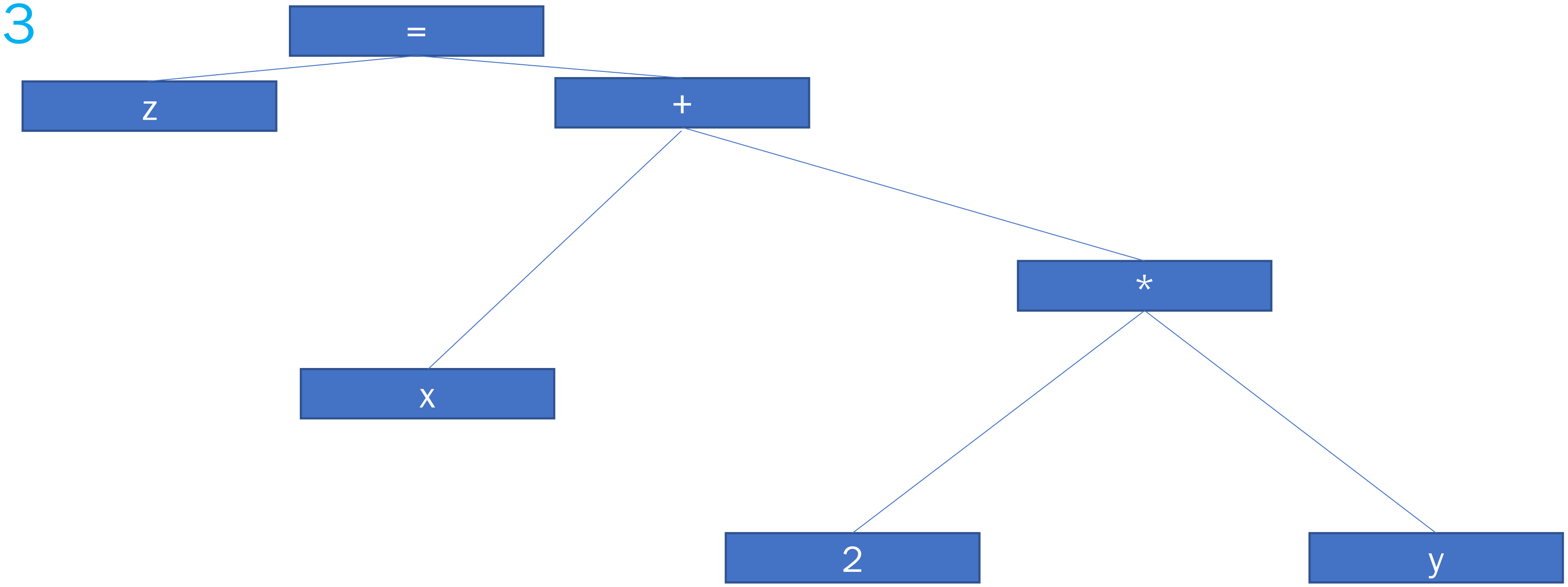
Example (Parse Tree)

Apply Rule 2



Example (AST)

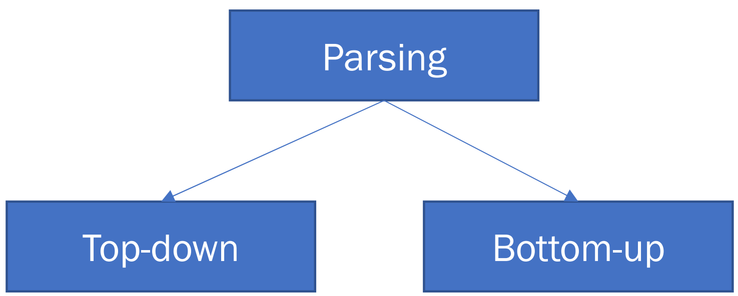
Apply Rule 3



Parsing

- Top-down: the parser starts constructing the parse tree from the start symbol and then tries to transform the start symbol to the input

- Bottom-up: parsing with the input symbols and tries to construct the parse tree up to the start symbols

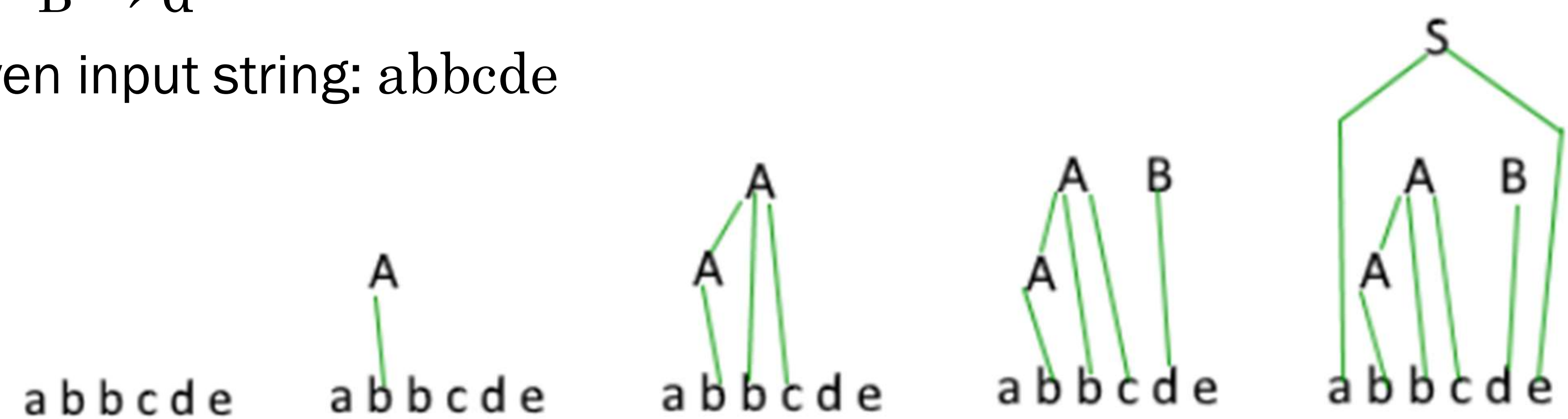


Bottom-up Parsing (Example)

- Say, we have grammar

$$S \rightarrow aABe$$
$$A \rightarrow Abc \mid b$$
$$B \rightarrow d$$

- Given input string: abbcde



Bottom-up Parsing (Example)

- Say, we have grammar

$$E \rightarrow T + E \mid T$$

$$T \rightarrow \text{int} * T \mid \text{int} \mid (E)$$

- Given input string: $\text{int} * \text{int} + \text{int}$

Bottom-up Parsing (Example)

- Say, we have grammar

$$E \rightarrow T + E \mid T$$
$$T \rightarrow \text{int} * T \mid \text{int} \mid (E)$$

- Given input string: int * int + int

$$\text{int} * \text{int} + \text{int}$$
$$\text{int} * T + \text{int}$$

Bottom-up Parsing (Example)

- Say, we have grammar

$$E \rightarrow T + E \mid T$$
$$T \rightarrow int * T \mid int \mid (E)$$

- Given input string: int * int + int

$$int * int + int$$
$$int * T + int$$
$$T + int$$

Bottom-up Parsing (Example)

- Say, we have grammar

$$E \rightarrow T + E \mid T$$

$$T \rightarrow \text{int} * T \mid \text{int} \mid (E)$$

- Given input string: int * int + int

$$\text{int} * \text{int} + \text{int}$$

$$\text{int} * T + \text{int}$$

$$T + \text{int}$$

$$T + T$$

Bottom-up Parsing (Example)

- Say, we have grammar

$$E \rightarrow T + E \mid T$$

$$T \rightarrow \text{int} * T \mid \text{int} \mid (E)$$

- Given input string: int * int + int

$\text{int} * \text{int} + \text{int}$

$\text{int} * T + \text{int}$

$T + \text{int}$

$T + T$

$T + E$

Bottom-up Parsing (Example)

- Say, we have grammar

  $E \to T + E \mid T$

  $T \to \text{int} * T \mid \text{int} \mid (E)$

- Given input string: $\text{int} * \text{int} + \text{int}$

  $\text{int} * \text{int} + \text{int}$

  $\text{int} * T + \text{int}$

  $T + \text{int}$

  $T + T$

  $T + E$

  E

Bottom-up Parsing (Exercise)

- Say, we have grammar

$$\text{Integer} \rightarrow \text{Digit} \mid \text{Integer Digit}$$
$$\text{Digit} \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$$

- Given input string: 24567

Reading and Exercises

Reading

- Chapter: 2.2 (Michael Scott Book)

Exercises

- Exercises: 2.9 and 2.10 (Michael Scott Book)