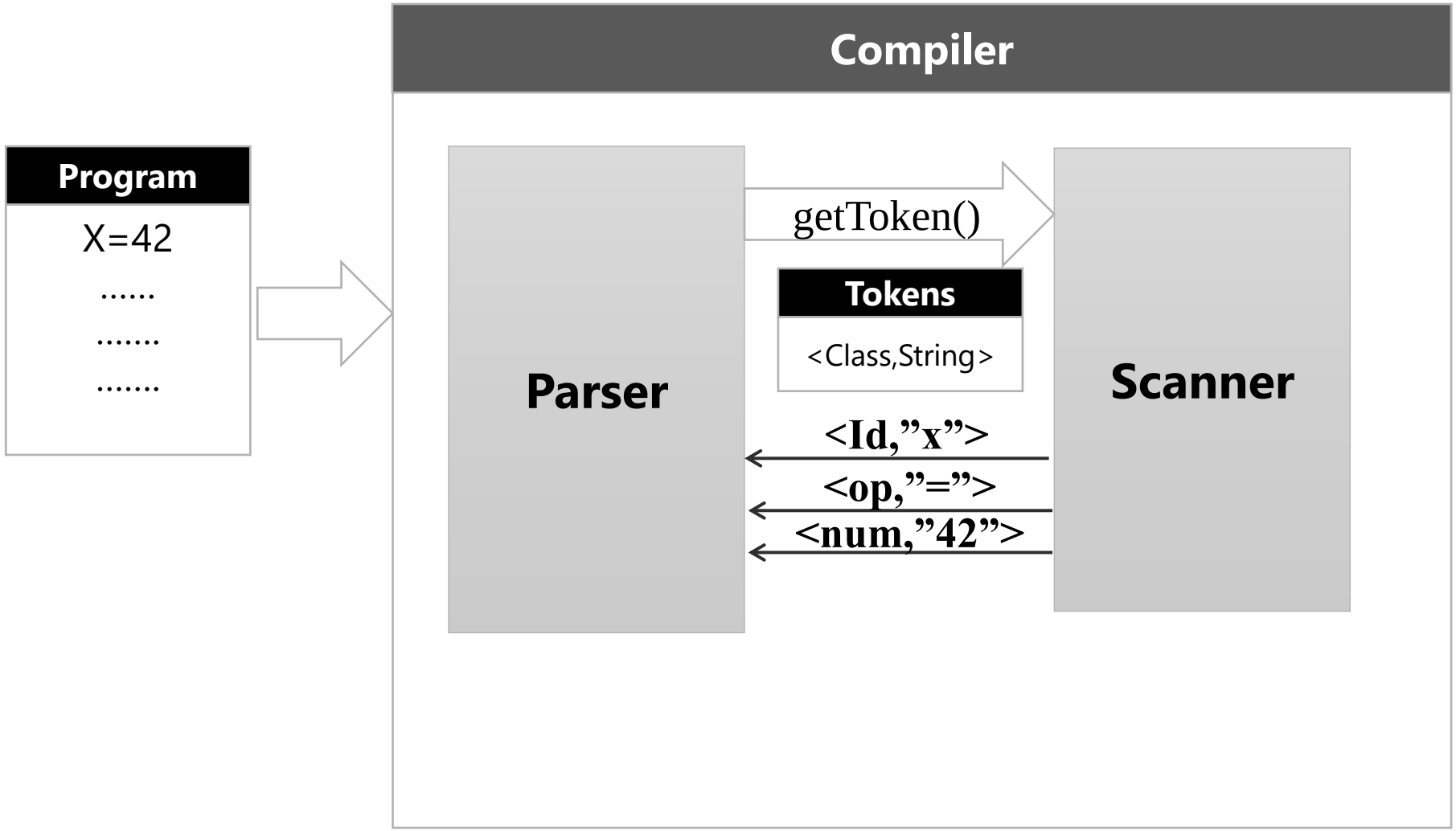
Parser

Lecture Six

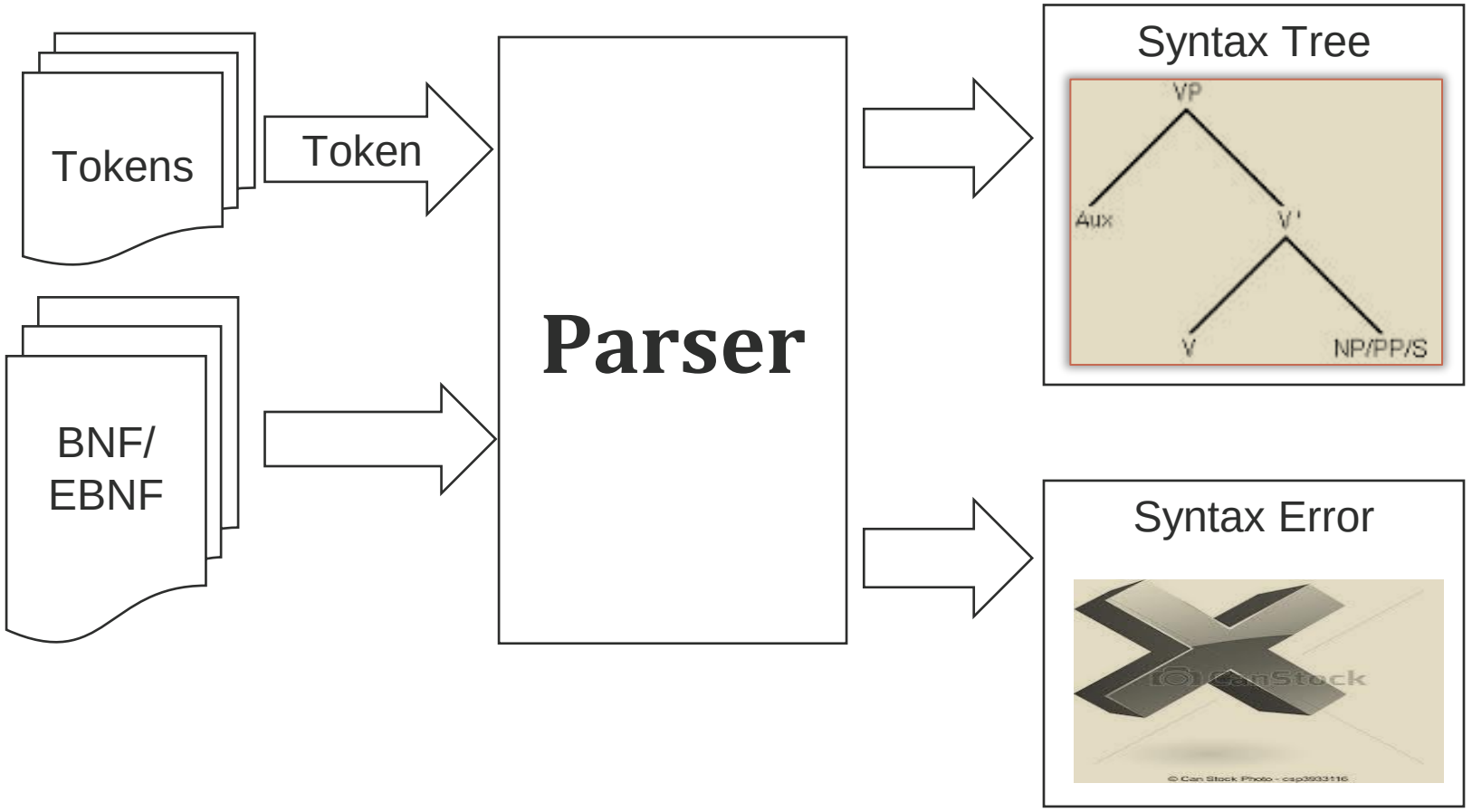
Lecture Topics

- The Parsing Process
- Context-Free Grammars
- Derivations
- Parse Trees
- Abstract Syntax Trees
- Ambiguous Grammars
- Dealing with ambiguity


Compiler
Program
X=42
.....
.....
.....
Parser
getToken()
Tokens
<Class,String>
Scanner
<Id,"x">
<op,"=">
<num,"42">

Parser (syntax Analyzer)

The Parsing Process



- جمله -->جمله فعليه| جمله اسميه|.....
- جمله فعليه--> فعل فاعل| فعل فاعل مفعول|فعل فاعل مفعول1 مفعول2 |.....
- فعل --> شرب |اكل |...
- فاعل-->الولد |البنت

Grammar

sentence –> <subject> <verb-phrase> <object>

subject –> This | Computers | I

verb-phrase –> <adverb> <verb> | <verb>

adverb –> never

verb –> is | run | am | tell

object –> the <noun> | a <noun> | <noun>

noun –> university | world | cheese | lies

leftmost derivation

- sentence –> <subject> <verb-phrase> <object>
- sentence–> This <verb-phrase> <object>
- sentence–> This <verb> <object>
- sentence–> This is <object>
- sentence–> This is a <noun>
- sentence–> This is a university

sentence –> <subject> <verb-phrase> <object>

subject –> This | Computers | I

verb-phrase –> <adverb> <verb> | <verb>

adverb –> never

verb –> is | run | am | tell

object –> the <noun> | a <noun> | <noun>

noun –> university | world | cheese | lies

The Parsing Process

- Input: *tokens*
- Output: A *syntax tree*
- lexical structure of tokens is specified by regular expressions, the syntax of programs is specified by the *grammar rules* of a *context-free grammar*.
- Parser must distinguish between valid and invalid programs (strings of tokens).
- (1 + + 2) + 3
- The parser verifies that the string can be generated by the grammar for the source language

Regular Languages

- The languages used till now is Regular Language
- A language is *regular* if it is accepted by some DFA.

  Examples:
  - The strings that represent floating point numbers in Java.
- Some Languages cannot be represented by DFA. They are not regular
- any language that requires **unbounded counting** cannot be regular

  Examples:
  - L1 = $\{0^n1^n \mid n \geq 1\}$: The set of strings consisting of one or more 0's followed by the same name of 1's.
  - L2 =$\{(^n)^n \mid n>0\}$:The language of balanced parentheses. e.*g*. $(), ()(), (()), (()()), \ldots$

Context-Free Grammars (CFG)

```
1. Prog  → Stm ; Prog | Stm
2. Stm   → AsStm | IfStm
3. AsStm → Var = Exp
4. IfStm → if Exp then AsStm
5. VorC  → Var | Const
6. Exp   → VorC | VorC + VorC
7. Var   → [a-z]
8. Const → [0-9]
```

CFG Example

Legal

```
a = 1; b = a + 4;
z = 1;
if b + 3 then z = 2
```

Illegal

```
a = x < 20
b = a + 4 + 5 ;
z = 1
if (a == 33) z < 2 ;
```

BNF Grammar

```
Prog  → Stm ; Prog | Stm
Stm   → AsStm | IfStm
AsStm → Var = Exp
IfStm → if Exp then AsStm
VorC  → Var | Const
Exp   → VorC | VorC + VorC
Var   → [a-z]
Const → [0-9]
```

Derivation

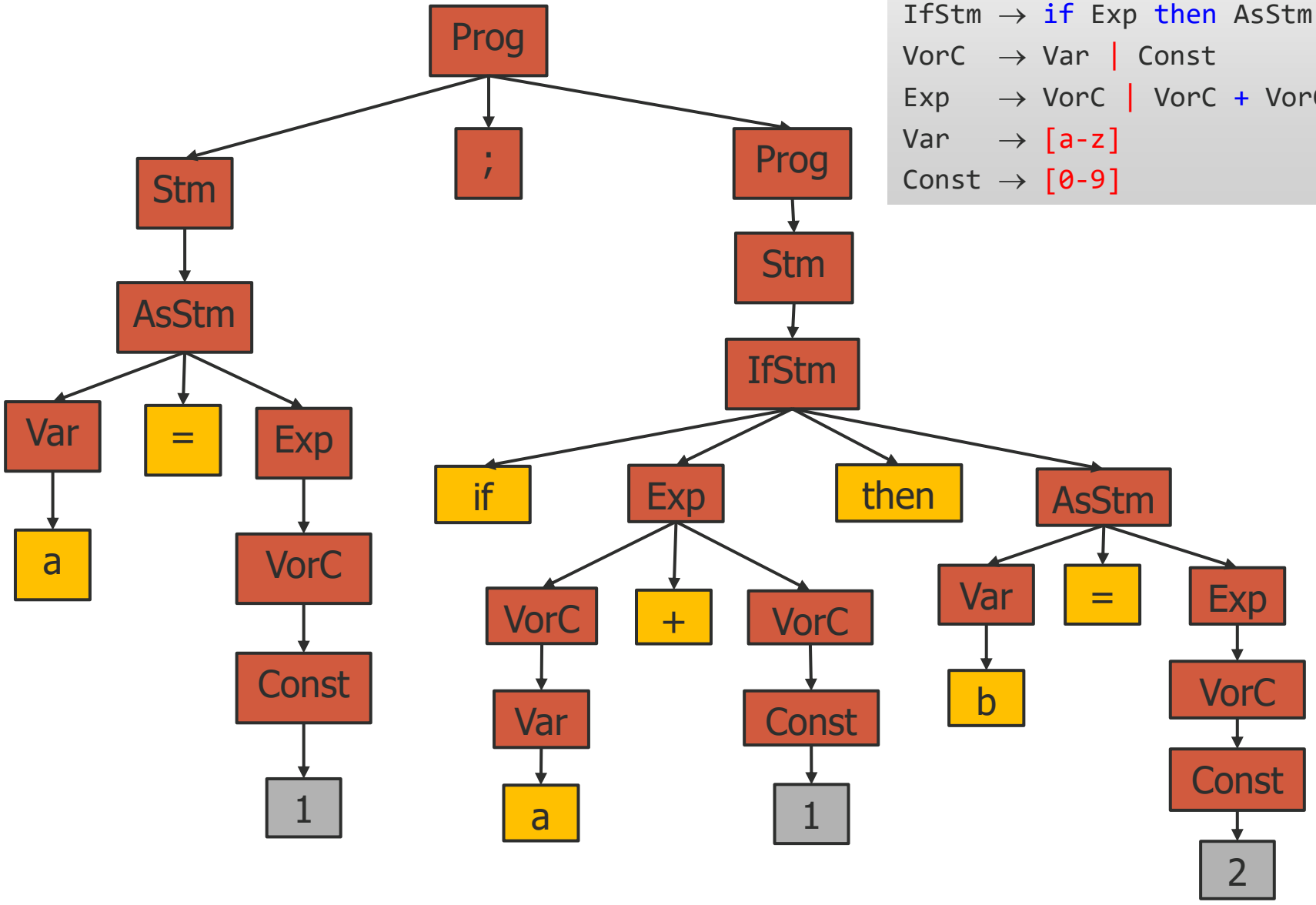
```
Prog  → Stm ; Prog | Stm
Stm   → AsStm | IfStm
AsStm → Var = Exp
IfStm → if Exp then AsStm
VorC  → Var | Const
Exp   → VorC | VorC + VorC
Var   → [a-z]
Const → [0-9]
```

```
Prog
=> Stm ; Prog
=> AsStm ; Prog
=> Var = Exp ; Prog
=> a = Exp ; Prog
=> a = VorC ; Prog
=> a = Const ; Prog
=> a = 1 ; Prog
=> a = 1 ; Stm
=> a = 1 ; IfStm
=> a = 1 ; if Exp then AsStm
=> a = 1 ; if VorC + VorC then AsStm
=> a = 1 ; if Var + VorC then AsStm
```

```
=> a = 1 ; if a + VorC then AsStm
=> a = 1 ; if a + Const then AsStm
=> a = 1 ; if a + 1 then AsStm
=> a = 1 ; if a + 1 then Var = Exp
=> a = 1 ; if a + 1 then b = Exp
=> a = 1 ; if a + 1 then b = VorC
=> a = 1 ; if a + 1 then b = Const
=> a = 1 ; if a + 1 then b = 2
```

Parse Tree

```
Prog  → Stm ; Prog | Stm
Stm   → AsStm | IfStm
AsStm → Var = Exp
IfStm → if Exp then AsStm
VorC  → Var | Const
Exp   → VorC | VorC + VorC
Var   → [a-z]
Const → [0-9]
```



Context Free Grammar(CFG)

Context free grammar G = (N, T, S, P)

- N : set of Non-terminal symbols
- T : set of Terminal symbols
- S : Start symbol (S ∈ N)
- P : set of Production rules $\{\alpha \rightarrow \beta | \ \alpha \in N \wedge \beta \in (N \cup T)^*\}$
- $N \cap T = \emptyset$

Note :

- $(N \cup T)^* = \{N \cup T \cup \varepsilon\}$
- ::= may be used instead of $\rightarrow$

CFG Example

L1= { 0^n1^n | n ≥ 1}

S → 01

S → 0S1

- Nonterminal= {S}.
- Terminals = {0, 1}.
- Start symbol = S.
- Productions =

  S → 01

  S → 0S1

CFG Example

L2 ={$(^n)^n$ | n>0}:The language of balanced parentheses.

S → (S)

S → ε

- N= {S}.
- T= {(,)}.
- Start symbol = S.
- Productions =

  S → (S)

  S → ε

CFG Example

$exp \rightarrow exp\ op\ exp \mid \mathbf{(}\ exp\ \mathbf{)} \mid \mathbf{number}$

$op \rightarrow \mathbf{+} \mid \mathbf{-} \mid \mathbf{*}$

- N= {exp,op}.
- T= {(,),number,*,-,+}.
- Start symbol = exp.
- Productions =

$exp \rightarrow exp\ op\ exp \mid (\ exp\ ) \mid \texttt{number}$

$op \rightarrow + \mid - \mid *$