Chapter 4

Context-free Grammars Part 3

Arithmetic expressions

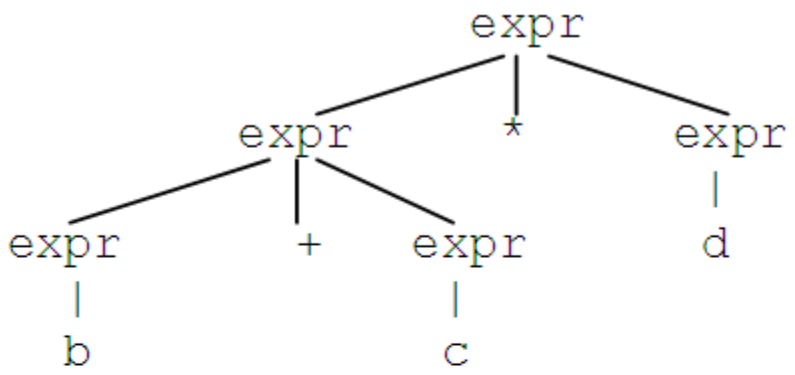
```
b
b+b+b
b+c
b*c +d
 (b+c+d)  *b
b+(c+d)
 (b)
 ((b))
```

Grammar for arithmetic expressions

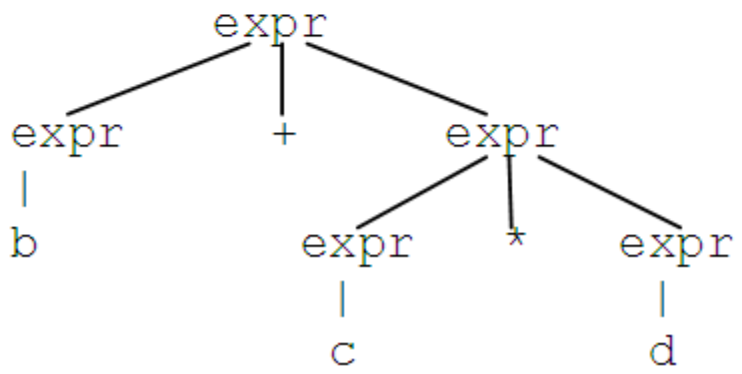
```
expr -> expr "+" expr
expr -> expr "*" expr
expr -> "b"
expr -> "c"
expr -> "d"
expr -> "(" expr ")"
```

But grammar is ambiguous

a)



b)

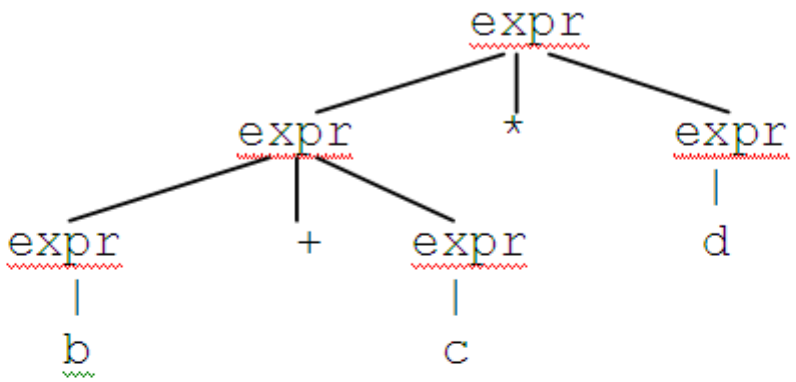


Another grammar

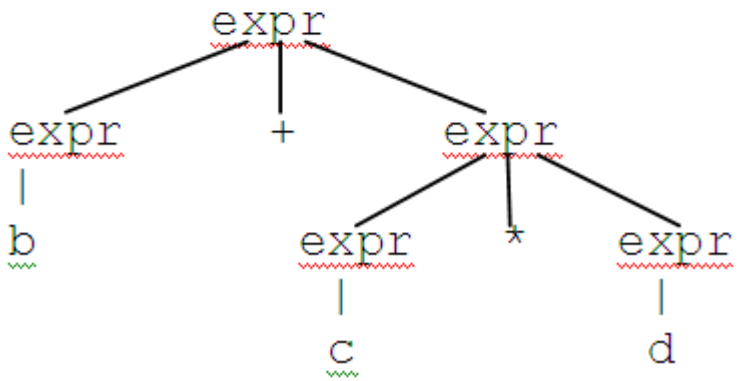
```
expr -> expr "+" expr
expr -> term
term -> term "*" term
term -> "b"
term -> "c"
term -> "d"
term -> "(" expr ")"
```

Ambiguous wrt associativity

a)



b)



Unambiguous grammar

```
expr -> expr "+" term
expr -> term
term -> term * factor
term -> factor
factor -> "b"
factor -> "c"
factor -> "d"
factor -> "(" expr ")"
```

Left recursion implies left associativity

Another unambiguous grammar

```
expr -> term termList
termlist -> "+" term termList
termList -> λ
term -> factor factorList
factorList -> "*" factor
factorList
factorList -> λ
factor -> "b"
factor -> "c"
factor -> "d"
factor -> "(" expr ")"
```

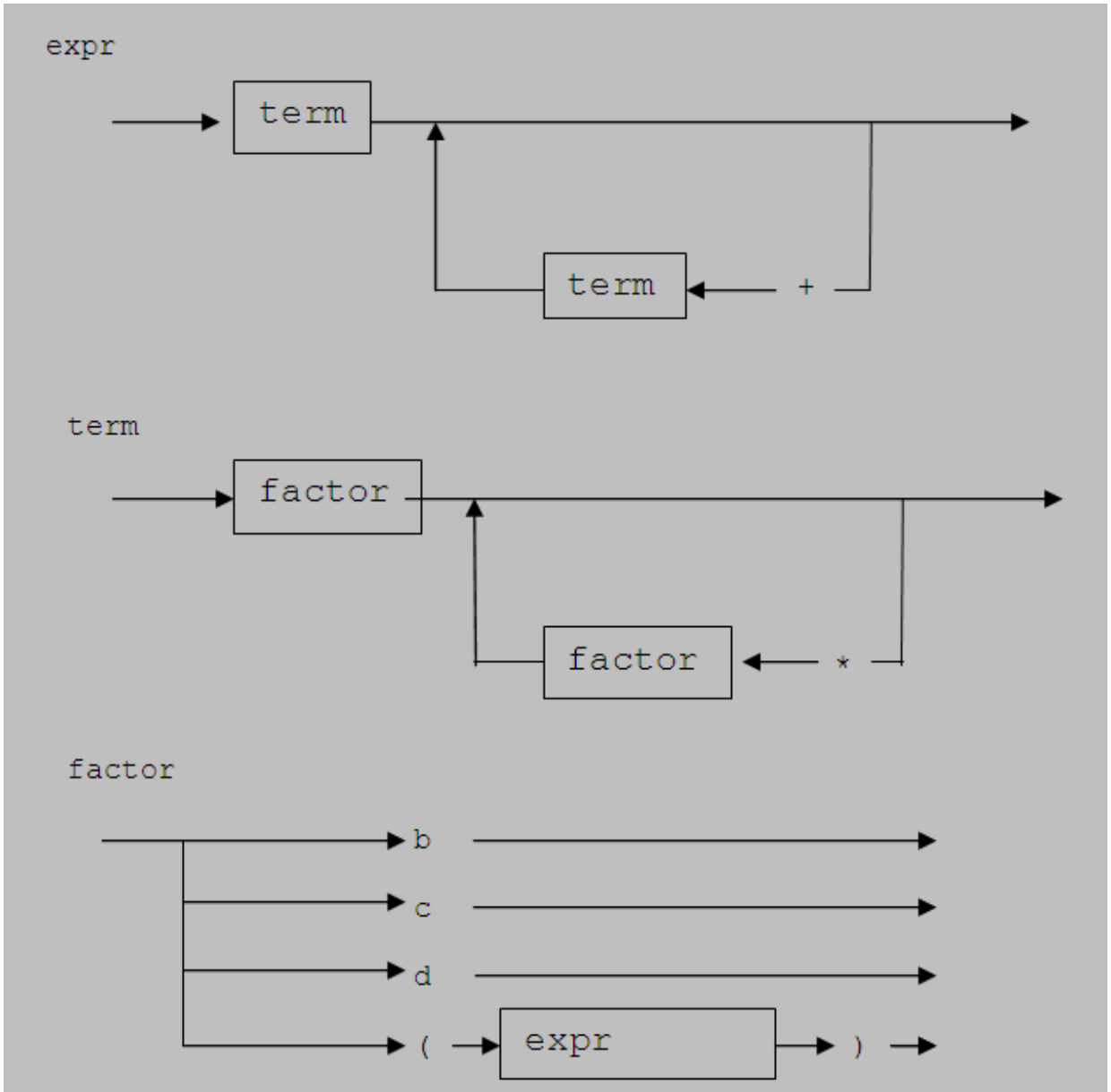
Backus-Naur Form (BNF)

```
expr : term termList
expr : "+" term termList | λ
term : factor factorList
factorList : "*" factor
factorList | λ
factor : "b"|"c"|"d"|"( expr ")"
```

Extended BNF

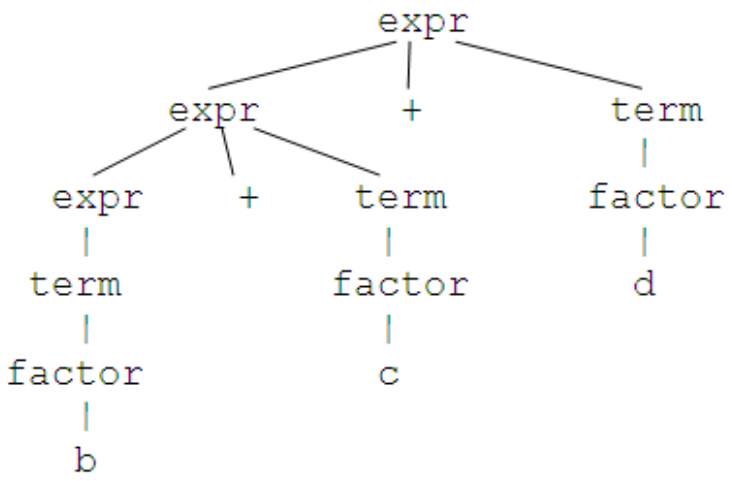
```
expr : term ("+" term")*
term: factor ("*" factor)*
factor: "b"|"c"|"d"|"(" expr ")"
```

Syntax diagrams



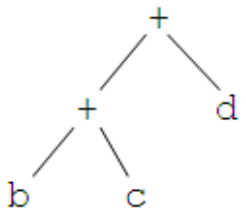
Abstract syntax tree

a)



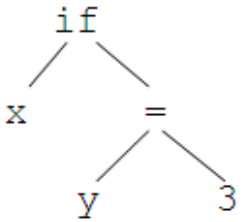
Parse tree for b+c+d

b)



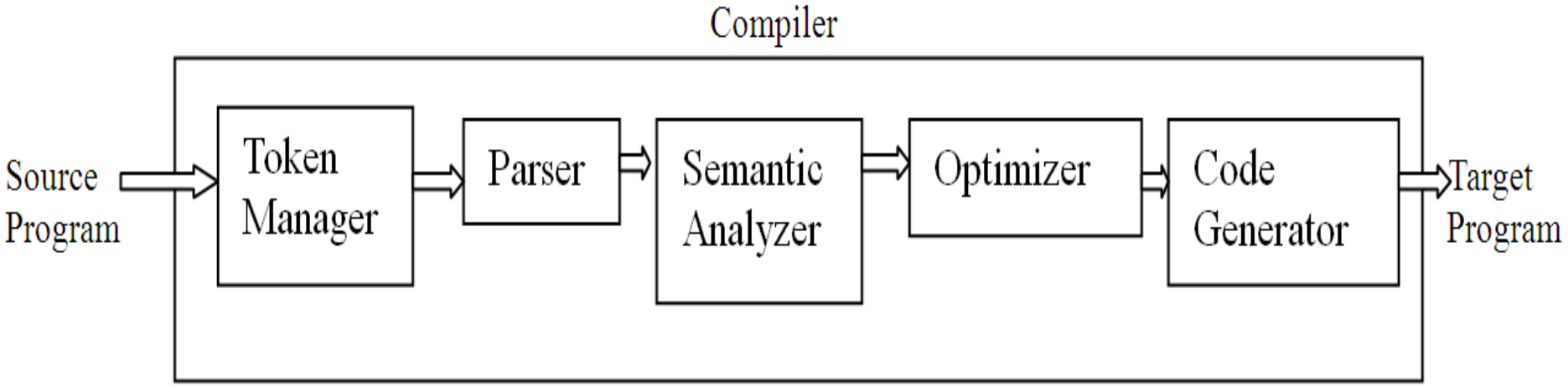
Abstract syntax tree for b+c+d

c)



Abstract syntax tree for `if (x) y = 2;`

Parts of a traditional compiler



Non-contracting grammar

Right side of each production at least as long as left side.

S -> bS
S -> c

Essentially non-contracting grammar

Start symbol can go to lambda if it does not appear on the right side of any production.

S -> b
S -> λ

Converting CFG to essentially non-contracting grammar

S -> bS

S -> λ

Eliminate lambda productions

S -> bS

S -> b

Add λ if in original language:

S' -> S

S' -> λ

S -> bS

S -> b

Pumping lemma for CFLs

If L is a CFG and $z \in L$ and is long enough, then there exists u, v, w, x, and y such that z = uvwxy where

$|vx| > 0$

$|vwx| < p$ for constant p that depends on only on L.

$uv^iwx^iy \in L$ for all $i \geq 0$

PAIRED violates pumping lemma

PAIRED = $\{ b^i c^i \mid i \geq 0 \}$

$b^p c^p$ cannot be parsed into uvwxy so that v and x can be pumped.