システムプログラミング実験
コンパイラの作成

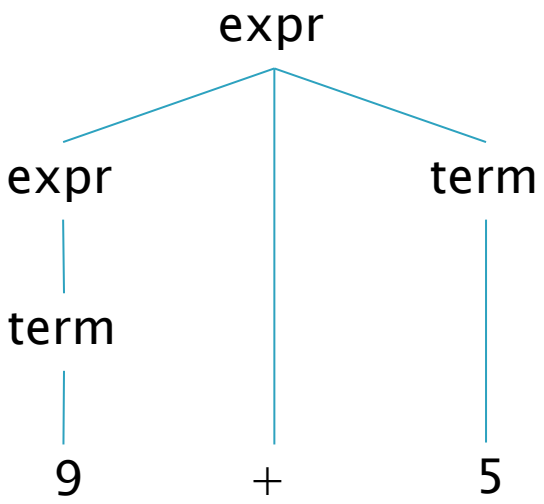
構文解析の補足

生成規則から構文解析ルーチンへ(1/4)

- 次の生成規則を考える

```
expr → expr + term | expr – term | term
term → 0 | 1 | … | 9
```

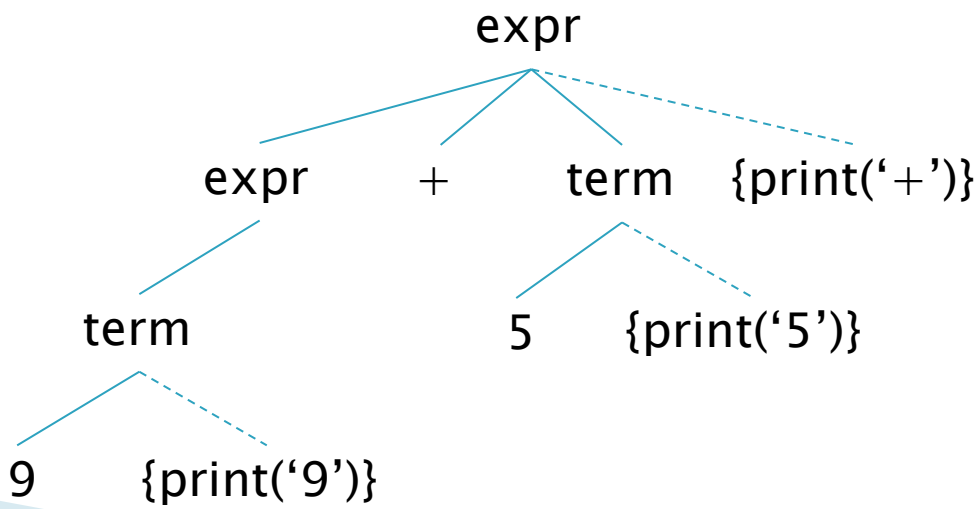
  - 例えば算術式「9 + 5」はこの規則を適用して生成される



生成規則から構文解析ルーチンへ(2/4)

- 後置形に変換するための出力を生成規則に追加

```
expr → expr + term {print('+')}
expr → expr - term {print('-')}
expr → term
term → 0 {print('0')}
term → 1 {print('1')}
:
term → 9 {print('9')}
```



解析木を深さ優先巡回すると,「9+5」の後置形「95+」が出力される

生成規則から構文解析ルーチンへ(3/4)

- exprの生成規則は左再帰の形をしているので再帰下降構文解析が使えない

```
//expr → expr + term {print('+')}の再帰下降構文解析
expr() {
        expr();  //再帰呼出し(無限ループ！)
        match('+');
        term();
        printf('+');
}
```

生成規則から構文解析ルーチンへ(4/4)

- 構文主導翻訳スキーム
  - 新たな非終端記号restを導入して左再帰を除去

```
expr → term rest
rest → + term {print('+')} rest
rest → -  term {print('-')} rest
rest → ϵ
term → 0 {print('0')}
term → 1 {print('1')}
:
term → 9 {print('9')}
```

  - 教科書の図2.22(p.61)のCプログラムはこの規則に従って再帰下降構文解析を実装