システムソフトウェア講義の概要

1. 計算機システムの復習：中央演算処理装置(CPU)，プログラムの実行，主記憶装置，補助記憶装置
2. 時分割処理：プロセス，スレッド，スケジューリング
3. スレッド間の排他制御：フラグ，セマフォ，モニタ，デッドロック
4. デバイス管理，HDDへのアクセス制御
5. 記憶管理：メモリ割り当て，ページング，セグメンテーション
6. 仮想記憶とファイルシステム
7. 演習問題
8. プログラミングシステムの概要，文法とそのクラス，字句解析と正規文法
9. 正規表現からの非決定性オートマトンの生成、決定性オートマトンへの変換
10. 字句解析用オートマトン生成ソフトウエアの実際
11. 構文解析と導出，文脈自由文法の構文解析法：LL構文解析
12. 文脈自由文法の構文解析法：LR構文解析
13. コンパイラ-コンパイラと構文解析の実際
14. 演習問題
15. 講義の総括と試験

高級言語

表 3.1　代表的な高級言語の例

FORTRAN	PL/I
DO 10 I=1,10000 READ*,X IF (X.GT.MAX) MAX=X 10 CONTINUE	L: DO I=1 TO 10000; GET LIST(X); IF X>MAX THEN MAX=X; END L;
BASIC	**COBOL**
10 FOR I=1 TO 10000 20 INPUT X 30 IF X>MAX THEN MAX=X 40 NEXT I	PROCEDURE DIVISION. L. ACCEPT X. IF X> MAX MOVE X TO MAX. GO TO L.
C	**APL**

for(i=1;i<=10000;i++){
scanf("%d", &x);
if(x>max) max=x;
}
	X←3 1 17 5 10 4 9 7 14 6
	⌈/X

PASCAL	**LISP**
for i:=1 **to** 10000 **do** **begin** read(x); **if** x>max **then** max:=x **end**;	(defun what-day(day) (cond ((member day '(月 火 水 木 金)) '平日)((member day '(土 日)) '週末)(t 'エラー)))
ALGOL	**PROLOG**
for i:=1 **until** 10000 **do** **begin** ininteger(x); **if** x>max **then** max:=x **end**;	human(Socrates). mortal(X) :- human(X).

プログラムの作成から実行まで

コンパイラ

プログラムテキスト

実行プログラム

アセン

リンカ

```
program example(output);
 var i, sum : integer
begin
  sum := 0;
  for i :=1 to 100 do
+i;
  writeln(sum)
end.
```

```
main:
.globl  PASCALMAIN
        .type   PASCALMAIN,@function
PASCALMAIN:
.globl  program_init
        .type   program_init,@function
program_init:
        pushl   %ebp
        movl    %esp,%ebp
        subl    $4,%esp
        call    FPC_INITIALIZEUNITS
        movw    $0,_SUM
        movw    $1,_I
        .balign 4,144
.L7:
        movswl  _SUM,%eax
        movswl  _I,%edx
        addl    %eax,%edx
        movw    %dx,_SUM
        cmpw    $100,_I
        jge     .L6
        incw    _I
        jmp     .L7
```

```
457f 464c 0101 0001 0000 0000 0000 0000
0002 0003 0001 0000 8080 0804 0034 0000
b43c 0000 0000 0000 0034 0020 0002 0028
0005 0004 0001 0000 0000 0000 8000 0804
8000 0804 ab00 0000 ab00 0000 0005 0000
1000 0000 0001 0000 b000 0000 3000 0805
3000 0805 0420 0000 0e00 0004 0006 0000
1000 0000 0000 0000 0000 0000 0000 0000
8959 89e3 40c8 e0c1 0102 83e0 f8e4 f8a3
053d 8908 3c0d 0534 8908 481d 0534 9b08
e3db d99b 002d 0530 3108 e8ed a9d0 0000
                . . .
0000 0000 000b 0000 0001 0000 0006 0000
8080 0804 0080 0000 aa80 0000 0000 0000
0000 0000 0010 0000 0000 0000 0011 0000
0001 0000 0003 0000 3000 0805 b000 0000
0420 0000 0000 0000 0000 0000 0004 0000
0000 0000 0017 0000 0008 0000 0003 0000
3420 0805 b420 0000 09e0 0004 0000 0000
0000 0000 0010 0000 0000 0000 0001 0000
0003 0000 0000 0000 0000 0000 b420 0000
001c 0000 0000 0000 0000 0000 0001 0000
                0000 0000
```

プログラムの例（Loop）

- １から１００までの和を求めるプログラム。

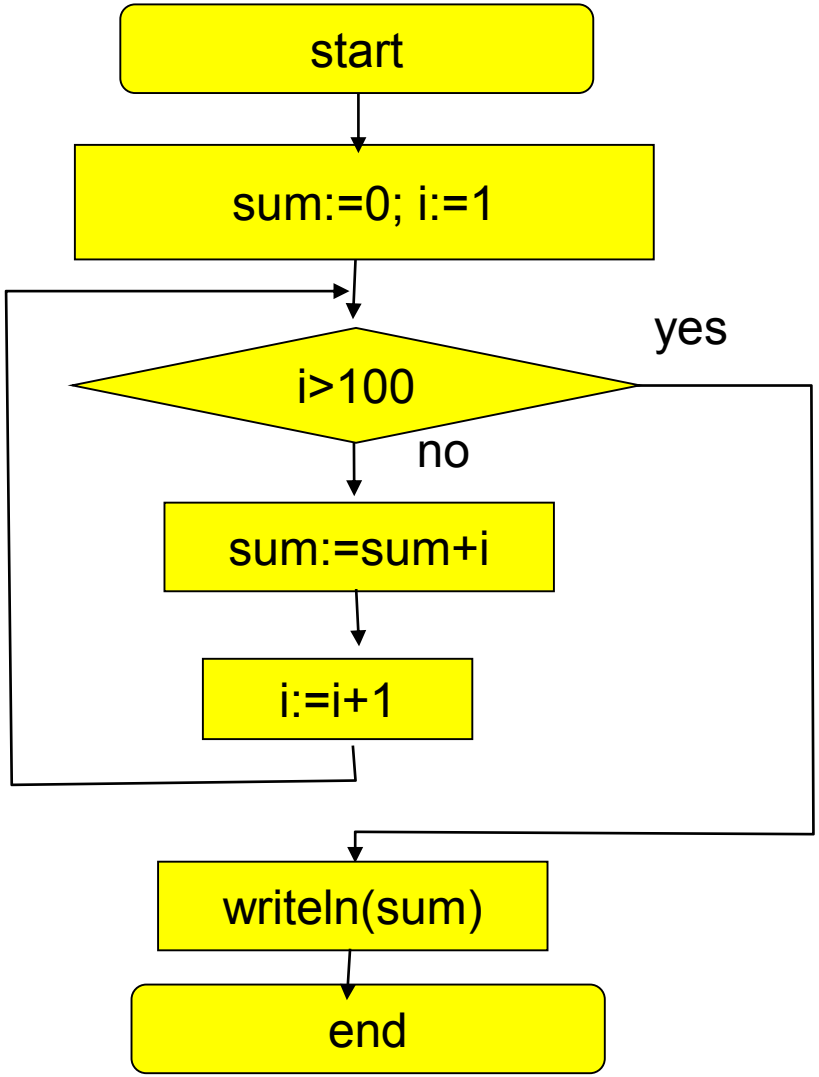
```
program example(output);
 var i, sum : integer;
begin
  sum := 0;
  for i :=1 to 100 do
                sum := sum +i;
  writeln(sum)
end.
```

結果を格納する変数の初期化

この部分をi=1から
i=100まで繰り返す

結果を画面に出力

フローチャートによる計算の記述




```
program example(output);
 var i, sum : integer;
begin
  sum := 0;
  for i :=1 to 100 do
                sum := sum +i;
  writeln(sum)
end.
```

プログラムの例(if 文)

- 最大値を求めるプログラム。

```
program example(input,output);
 var i, x, max: integer;
begin
   x := 0; max := 0;
   for i := 1 to 10 do
   begin
      read(x);
      if x > max then max := x
   end;
   writeln('maximum=',max)
end.
```

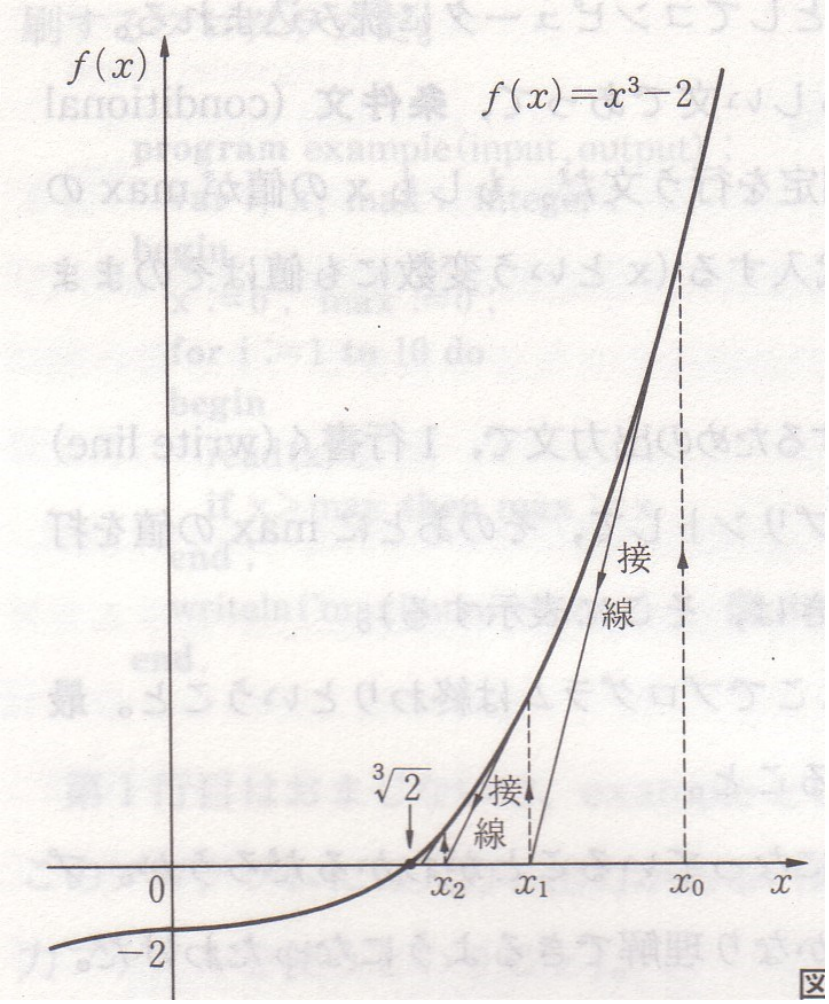
結果を格納する変数の初期化

入力された数値をxに格納する

最大値の更新

結果を画面に出力

プログラムの例(数値計算)




```
program Newton(output);
  var x, x1 : real;
begin
  x := 1.0;
  repeat
    x1 := x;
    x := (2.0/3.0) * (x1+1.0/sqr(x1))
  until abs(x-x1) < 1.0e-11;
  writeln('cubic root of 2=',x)
end.
```

x1²を表している

プログラムの例(数値計算)解説

$f(x)=x^3-2$ の $x=x_1$ での接線の方程式 $g(x)$ を求める

$$g(x)=f'(x_1)(x-x_1)+f(x_1)$$
$$=3x_1^2(x-x_1)+x_1^3-2$$
$$=3x_1^2x-2x_1^3-2$$

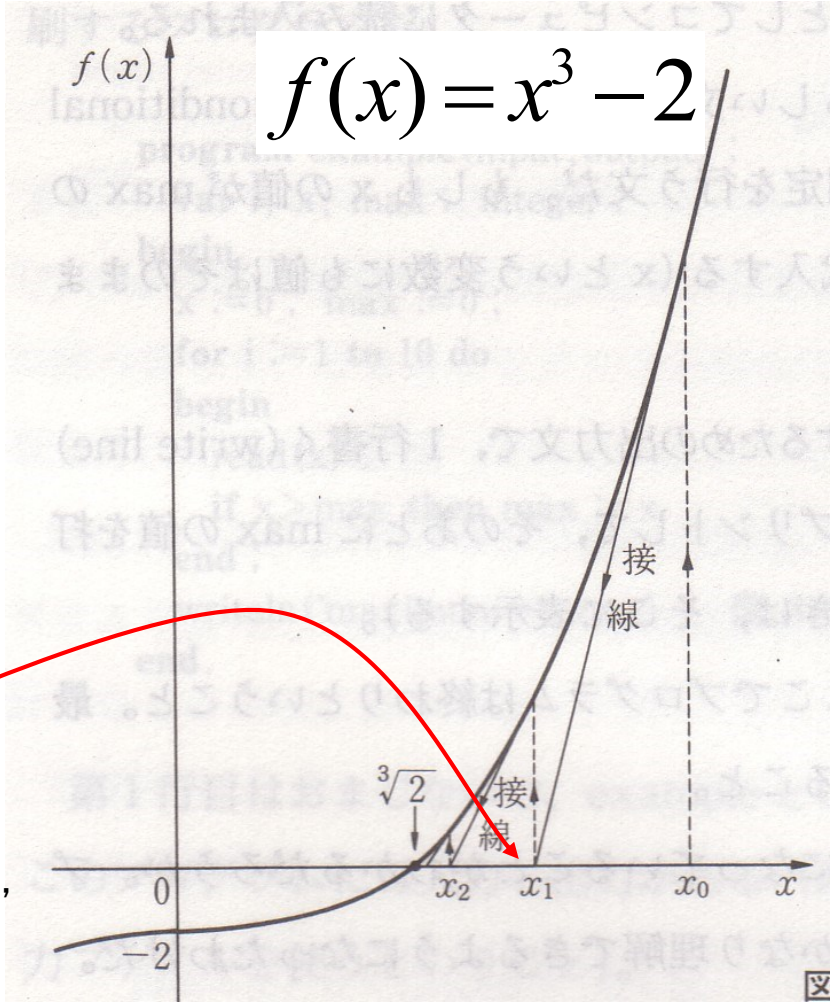
接線が x 軸と交わる位置は,

$$g(x)=3x_1^2x-2x_1^3-2=0$$

から

$$x=\frac{2}{3}\left(x_1+\frac{1}{x_1^2}\right)$$

この x を新たな x_1 として, 上の計算を繰り返すと,
答えが求められる.



プログラミング言語処理系

高級言語の処理系

インタープリタ

プログラムを逐次解釈実行する.

コンパイラ

プログラムを機械語に変換する.

コンパイラの概要

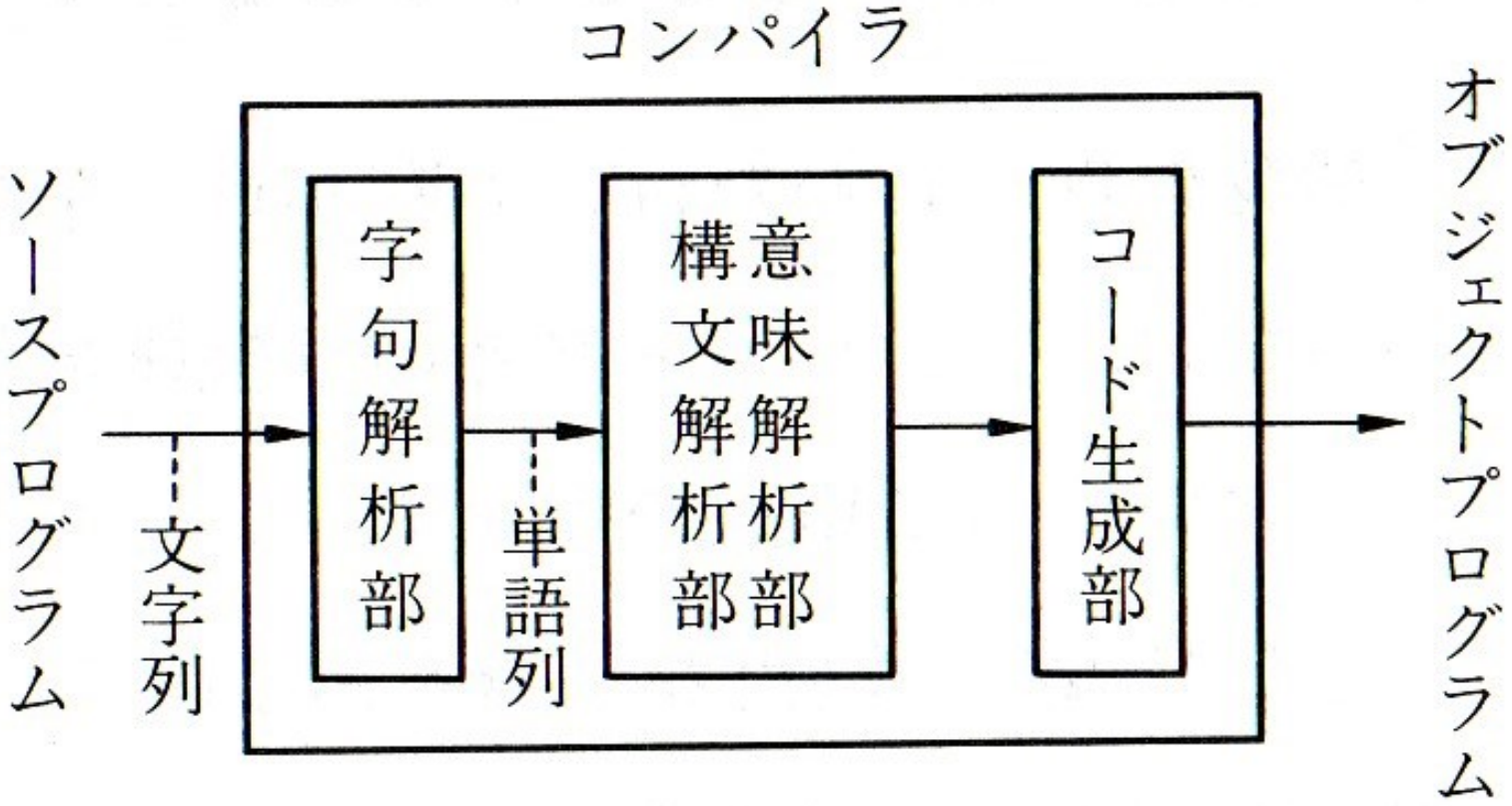


図 7.1　コンパイラの基本構造

文法と言語
ー正規表現とオートマトンー

和田俊和

資料保存場所

http://vrl.sys.wakayama-u.ac.jp/~twada/syspro/

言語は，単なる記号の並び．．．ではない

- 言語はある規則を満足する記号列（文）の集合
  - 例：　日本語，英語，C言語，その他
  - 「ex@p蛇Wx労z$壺-^ofD魔」は上記言語に属さない
  - 「while (A<100) A=A+1;」 はC言語に属する「文」

- ある言語に属する文は無数に存在する．（無限集合）
- 文は形式的に定義可能→文法の必要性

なんで，こんなことを学ぶのか？

- 言語には，「文法」という規則がある．
- この規則を知らずに，文を書くことも読むこともできない．
  - 現実に，プログラミング言語の「正しい文法」を知らない学生は，許される文と許されない文の区別がつかない．
- プログラミング言語処理系では，実際に構文解析や字句解析が行われており，これを理解しなければ，情報の基礎を学んでいるとは言えない．

形式言語理論

- 文法は,
  - P:生成規則
  - S:出発記号
  - N:非終端記号の集合
  - T:終端記号の集合

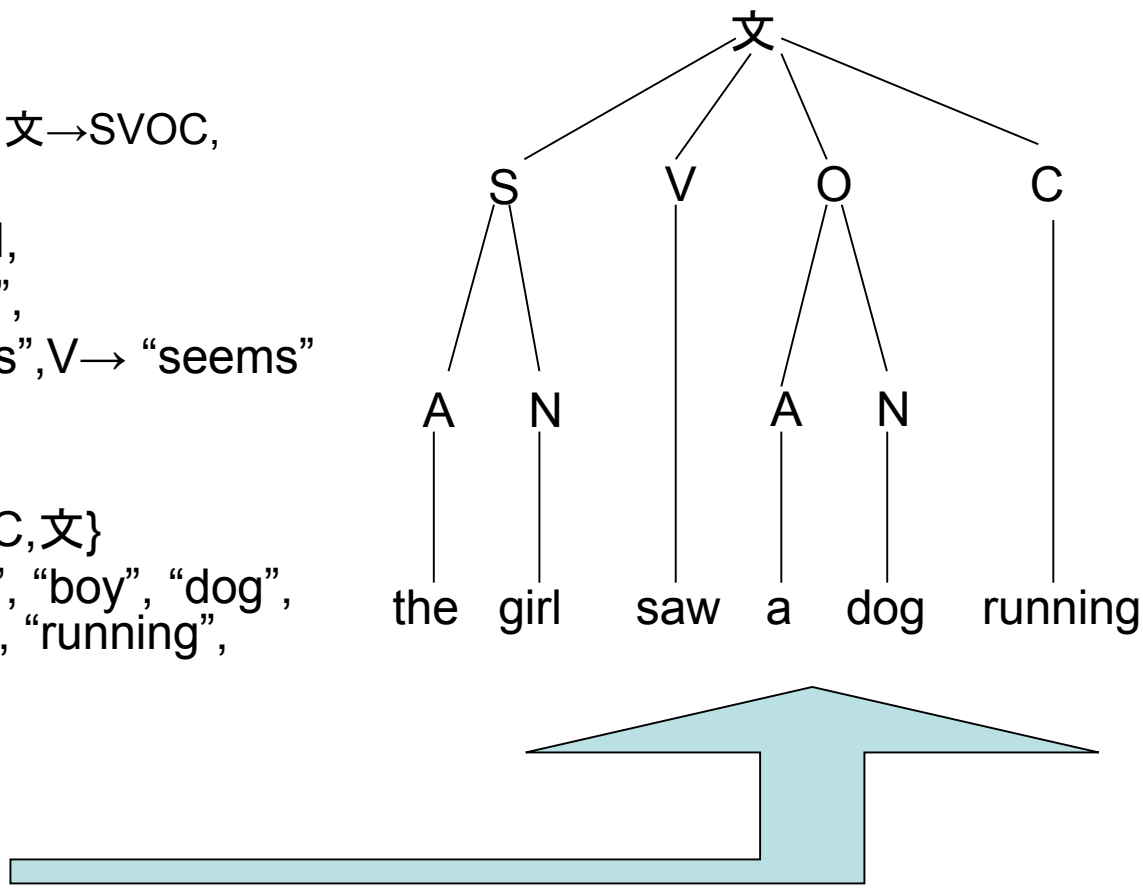
  の4つの組で表される.

  G＝{P，S，N，T}

文法の例1

- 生成規則　P={
  文→SV，文→SVC，文→SVO，文→SVOC,
  A→“the”，A→ “a”,
  S→AN, S→N, O→AN, O→N,
  N→“girl”, N→“boy”, N→“dog”,
  V→“saw”,V→“runs”,V→“bites”,V→ “seems”
  C→“running”, C→“sick”}
- 出発記号　S＝文
- 非終端記号　N＝ {A,S,N,V,O,C,文}
- 終端記号　T＝ {“a”, “the”, “girl”, “boy”, “dog”, “saw”, “runs”, “bites”, “seems”, “running”, “sick”}

対応する言語の例,
the girl saw a dog running
the dog runs
the dog bites a girl
the boy seems sick



導出，言語（ここは我慢して聞いて）

- V を有限個の記号から成る空でない集合とする.
- V に含まれる記号 $x_1, x_2, x_3 \cdots$ を並べた長さ1以上の記号列を V 上の「語」と呼び，語全体をV^+で表す.
- また$V^* = V^+ \cup \{\varepsilon\}$とする.
- $(T \cup N)^*$の要素 uに生成規則を何回か適用することによってvが生成されることを，「導出」と呼び，$u \xrightarrow{*} v$ と表す.
- $L(G) = \{x \in T^* \mid S \xrightarrow{*} x\}$で表される V^*の部分集合を文法 $G = (T, N, P, S)$ が生成する言語と呼ぶ.

文法とそのクラス

- タイプ0文法

$$s \rightarrow t$$

但し，$s \in (T \cup N)^+$, $t \in (T \cup N)^*$

- タイプ1文法（文脈依存文法）

$$mAn \rightarrow mtn$$

但し，$m,n \in (T \cup N)^*$, $t \in (T \cup N)^+$, $A \in N$

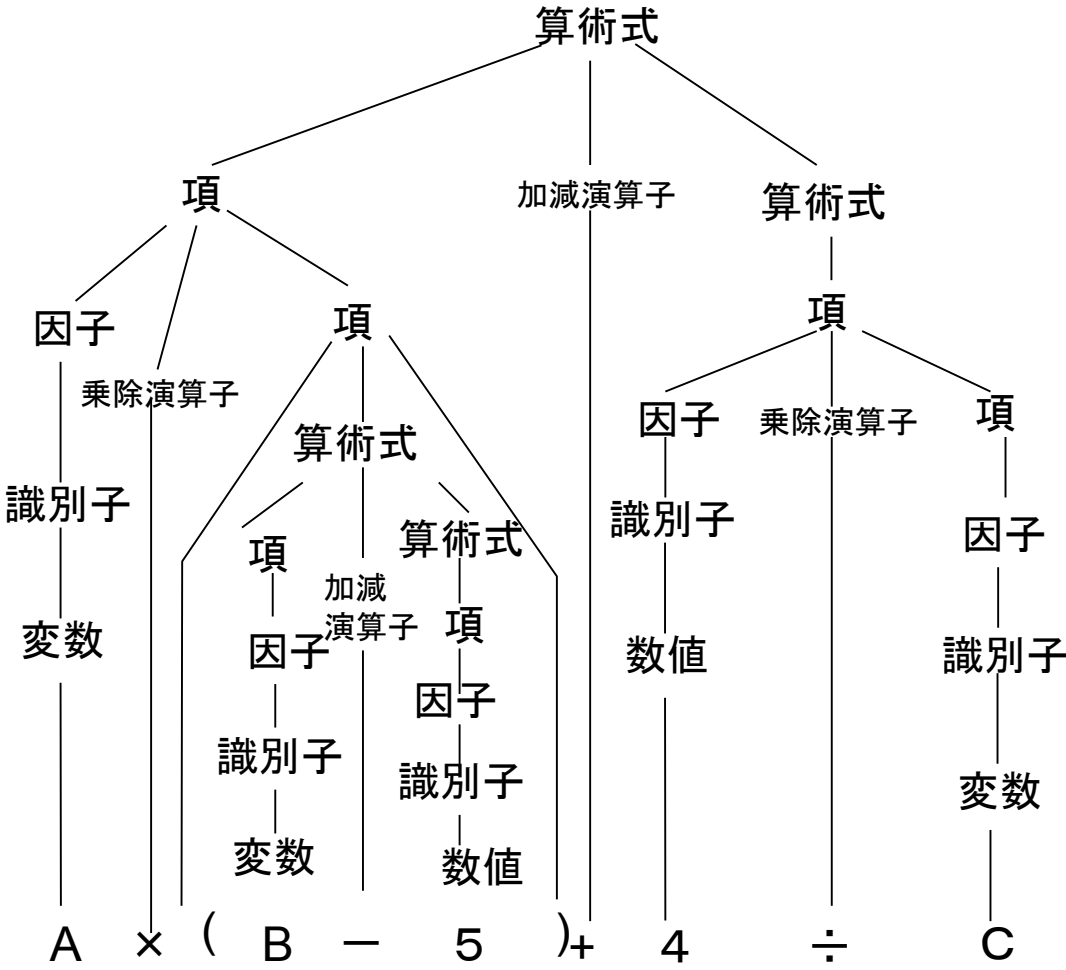
- タイプ2文法（文脈自由文法）

$$A \rightarrow t$$

但し，$t \in (T \cup N)^*$,$A \in N$

文脈自由文法の例

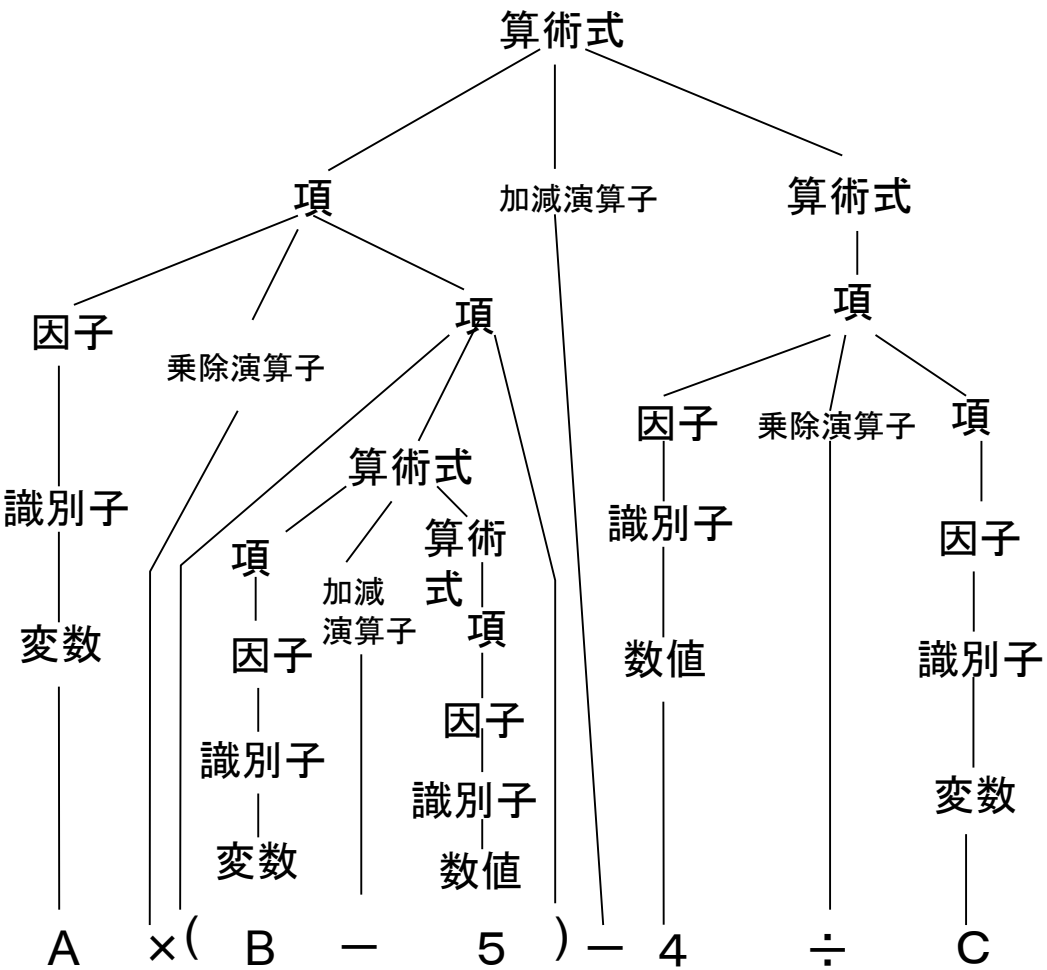
- 生成規則　P={
  算術式 →項 加減演算子 算術式,
  算術式→ 項
  項 → 因子　乗除演算子　項
  項 → 因子
  因子→ 識別子
  因子→ “(“ 算術式 “)”
  識別子→変数
  識別子→数値
  数値→”1”, 数値→”2”, … ,数値→”9”
  変数→”A”, 変数→”B”, 変数→”C”
  加減演算子→“+”,　加減演算子→“－”
  乗除演算子→“×”,　乗除演算子→“÷”}
- 出発記号　S＝算術式
- 非終端記号　N＝ {算術式, 項, 因子, 識別子, 数値, 変数, 英数字, 加減演算子, 乗除演算子}
- 終端記号　T＝{“+”, “－”, “×”, “÷”, “0”,”1”,…,”9”, “A”,”B”,…,”Z”, “a”, “b”,…,”z”, “)”, “(“}



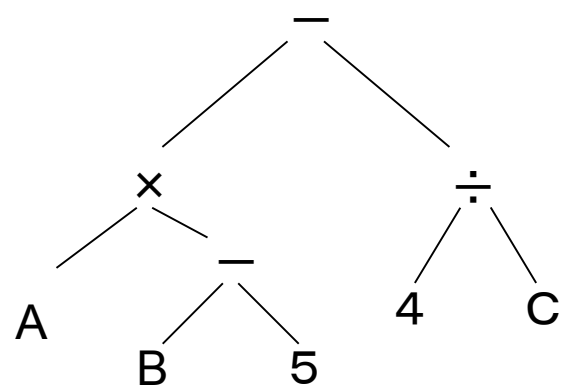
対応する言語の例

A×（B-5）+4÷C

(ちょっと脇道)何のための構文解析
→例:計算のため



導出木



- 単一導出による分岐の無い枝を削除
- 分岐点にある非終端記号を葉の部分にある演算子で置き換える

↓

構文木(演算子木)

↓

どうやって式の計算をすれば良いか？考えてみよう.

より簡単な文脈自由文法

- 整数と実数を定義したい.

出発記号　S＝数値

生成規則　P＝{

数値→ 数字列, 数値→数字列 “.” 数字列,

数字列→数字列　数字,　数字列→数字,

数字→”0”, 数字→”1”, … , 数字→”9”

}

N=数値, 数字列, 数字

T={“0”,”1”, … ,”9”, “.”}

正規文法

- タイプ3文法（正規文法）

$$A \rightarrow a \text{ あるいは } B \rightarrow bB$$

$$\text{但し，} A,B \in N,\ b \in T,\ a \in (T \cup \{\varepsilon\})^*$$

先の例の生成規則を書き換えてみる.

P＝{

数値→ “0” 数値，数値→ “1” 数値，...

数値→”9” 数値，数値→ “.”数値B，数値→ε

数値B→ “0” 数値B，数値B→ “1” 数値B，...

数値B→”9” 数値B，数値B→ε

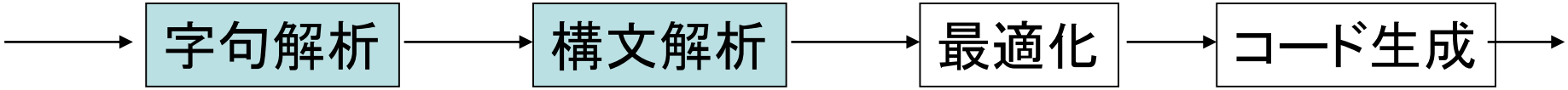
}

この生成規則だと，”.”が2回以上出てしまうことになる.

どうすればよいか？

字句解析と構文解析

- プログラム言語処理系における構文解析では,「整数値」,「実数値」,「変数名」など正規文法で表現可能な部分は，プログラムの中から事前に種類ごとに抽出しておき，後続の構文解析の処理が軽くなるようにしている.



- この字句解析を行うのがオートマトンである.

オートマトンを作るには，まず正規表現で字句を表現する

V 上の正規表現とは次のようなものである.

- 空列 ε は正規表現である.
- $a \in V$ ならば，a は正規表現である.
- r と s が正規表現ならば $r|s$ も正規表現である.
- r と s が正規表現ならば rs も正規表現である.
- r が正規表現ならば r^* も正規表現である.

尚，表現があいまいになる場合は括弧を用いる.

正規表現の意味

- 空列 ε は長さ0の記号列を表す.
- aは記号 a を表す.
- $r \mid s$ は r もしくは s を表す.
- rs はrとsがこの順番で繋がっていることを表す.
- r^* はrの0回以上の繰り返しを表す.

括弧は，一まとまりの正規表現であることを示す.

正規表現による数値の表現

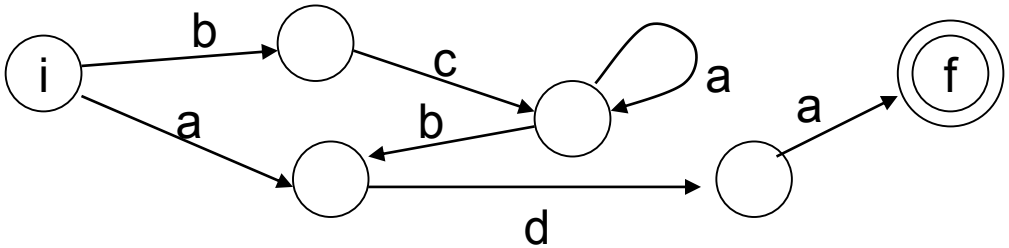
$$(1|2|...|9)(0|1|...|9)^*(\varepsilon|.(0|1|...|9)^*)$$

- 最初の数字は１～９までの数字であり,
- 引き続き０～９までの数字が0回以上繰り返される.
- これで終わりの場合もあるが,
- “. ”が来て０～９までの数字が0回以上繰り返される場合もある.

正規表現からオートマトンへ

- オートマトンとは何か？

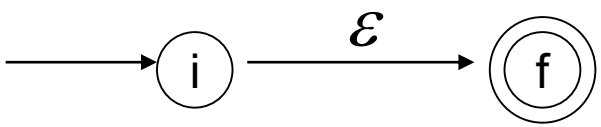
  初期状態からスタートして，記号を受け取りながら状態遷移を起こし終了状態に遷移する.



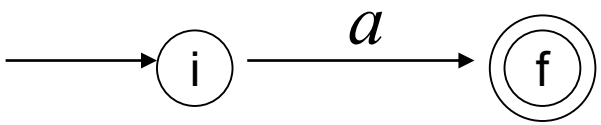
- 非決定性と決定性の2種類がある.
- 正規表現からは非決定性有限オートマトン（NFA）に変換できる.
- NFAから決定性の有限オートマトンに変換することができる

正規表現からオートマトンへ

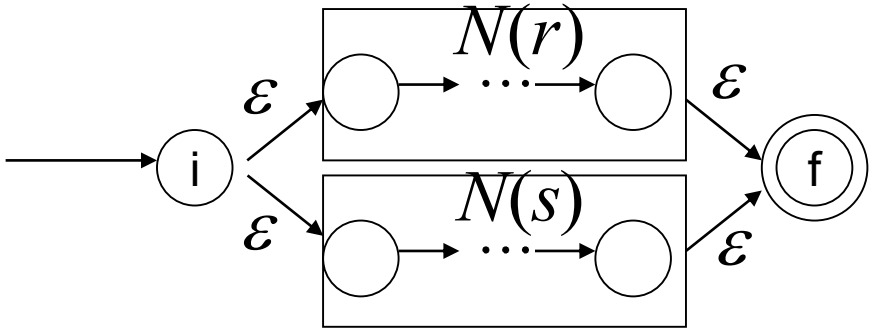
- ε



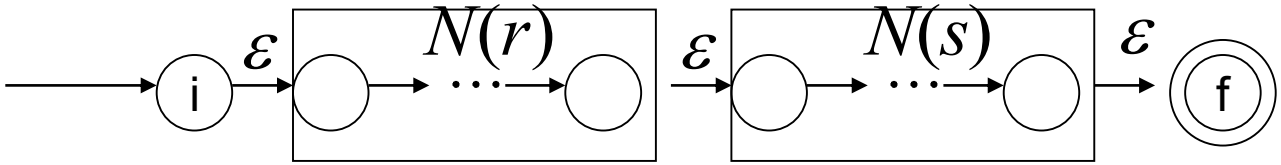
- a



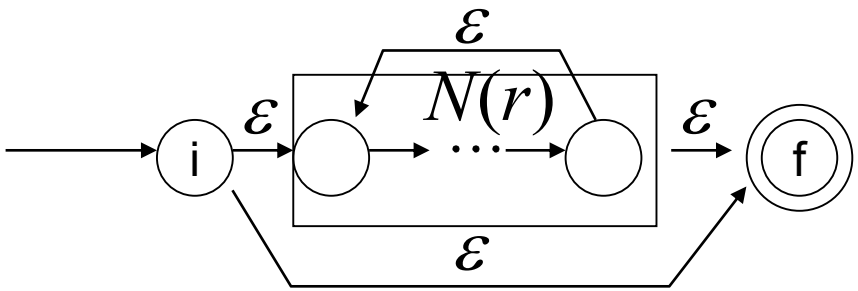
- $r \mid s$



- rs



- r^*



先ほどの数値の正規表現に対応するオートマトンを書きなさい