第1章 演算子順位による構文解析 (教科書 **p.40**)

1.1 構文解析とは（復習）

構文解析とはプログラム（式）の構造を木の形に表すことである。

正規言語の反復補題（ポンプの補題）からわかるように、正規表現 (DFA) では構文解析はできない。

例:

```
gokei = soryo + kosu * tanka
              ↑       ↑       ↑
              Ⓐ       Ⓑ       Ⓒ
```

1.2 演算子順位法のアイデア

左から右に読む

Ⓐの地点 … 「=」の後ろに「+」があるのでオペランドの決定を保留する

Ⓑの地点 … 「+」の後ろに「*」があるのでオペランドの決定を保留する

Ⓒの地点 … 「*」の後ろにもう演算子がないのでオペランドを確定する

ここから、以下のアイデアがでてくる

- 演算子の＿＿＿＿・＿＿＿＿の情報を使う
- データを保留しておく必要がある　→　スタックを用いる
  左の演算子　<　右の演算子　　→　＿＿＿＿＿＿＿＿
  左の演算子　>　右の演算子　　→　＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

1.3 先人の知恵

- 演算子の関係は左右非対称で良い
  （x < y でも y > x とは限らない）←　「<」,「>」,「=」の代わりに「＿＿」,「＿＿」,「＿＿」 と書く
- 始・終・識別子・かっこなども演算子と同様に ⋖, ⋗, ≒ の関係をつける
  (教科書 p.45 表4.2)

左＼右	+	−	*	/	(	)	識別子	終	
⋮					⋮				
(	⋖	⋖	⋖	⋖	⋖	≒	⋖		
)	⋗	⋗	⋗	⋗		⋗		⋗	この行を追加
識別子	⋗	⋗	⋗	⋗		⋗		⋗	

始・終をともに「　」と書く

1.4 演算子順位法の例

教科書 p.45 表4.2 の表による演算子順位法で構文解析する。元の BNF は次のようになる。（曖昧な文法である。）

$E \to E + E \mid E * E \mid \mathbf{id} \mid (E)$

入力例として　$a+b*(c+d)$ を考える。（ただし、a ～ d は **id**（識別子）に属するトークンである。）

各動作では、スタックの一番上と入力の残りの先頭を比べている。このとき、スタックに非終端記号（この例の場合は *E*）が入っていても無視する。

スタック	入力の残り	動作	補足
		だから	
		だから	
		だから	
		だから	
		だから	
		だから	
		だから	
		だから	
		だから	
		だから	
		だから	
		だから	
		だから	
		だから	
		だから (※)	
		だから	
		だから	
		だから	

シフト (shift)　　…
還元 (reduce)　　…

補足 「≒」は何のためにある？
「⋖」「≒」ともに動作はシフトである。ただし、あとで「⋗」になって還元するときに「≒」を通り越して「⋖」のところまでポップする（２つ以上の終端記号をポップする）。上の※のところ参照。

還元が起こった箇所を下から読むと、

という導出列になっている。

1.5 演算子順位法の特徴

- 最も右側の非終端記号を書き換える（　　）
- 解析木の葉から幹へ向かって節を作っていく （　　）

- （ほとんど）どんなプログラミング言語でもコードを書ける
- 実行時にも構文解析表を更新できる(他の構文解析法の後処理に使える)

1.6 演算子順位法の制限

右辺に

- ε がでてくる、終端記号が出現しない
- あるいは、非終端記号が隣り合う

ような生成規則があると対応できないことが知られている

1.7 演算子順位行列 (**p.45** 表 **4.2** など) の作り方

1. ⊗ が ⊕ より優先順位が高いとき
   → ＿＿＿＿かつ ＿＿＿＿とする

2. ⊕ と ⊖ が同じ優先順位のとき
   左結合 → ＿＿＿＿かつ ＿＿＿＿、
   右結合 → ＿＿＿＿かつ ＿＿＿＿、
   非結合 → 空欄のまま（エラーを表す）とする

3. すべての演算子 ⊕ について、「**id**」、「(」、「)」、「$」 との関係は、表 4.2 と同じである。つまり、 ＿＿＿＿かつ ＿＿＿＿かつ ＿＿＿＿かつ ＿＿＿＿かつ ＿＿＿＿かつ ＿＿＿＿かつ ＿＿＿＿かつ ＿＿＿＿とする

4. さらに「**id**」、「(」、「)」、「$」 同士の関係は、表 4.2 と同じである。つまり、 ＿＿＿＿かつ ＿＿＿＿かつ ＿＿＿＿かつ ＿＿＿＿などなど、とする

注： 単項演算子の「−」については、字句解析のときに二項演算子の「−」と区別しておく必要がある

問 **1.7.1** 教科書 p.45 表 4.2 に累乗演算子「^」と比較演算子「<」を追加せよ。ただし「^」は右結合で 「*」や「/」よりも、優先順位が高いものとする。 また「<」は非結合（例えば `2 < 2 < 3` は、構文エラー）で、「+」や 「−」よりも、優先順位が低いものとする。

左＼右	<	+	−	*	/	^	(	)	識別子	終
始		⋖	⋖	⋖	⋖		⋖	✗	⋖	≒
<										
+		⋗	⋗	⋖	⋖		⋖	⋗	⋖	⋗
−		⋗	⋗	⋖	⋖		⋖	⋗	⋖	⋗
*		⋗	⋗	⋗	⋗		⋖	⋗	⋖	⋗
/		⋗	⋗	⋗	⋗		⋖	⋗	⋖	⋗
^										
(		⋖	⋖	⋖	⋖		⋖	≒	⋖	✗
)		⋗	⋗	⋗	⋗		✗	⋗	✗	⋗
識別子		⋗	⋗	⋗	⋗		✗	⋗	✗	⋗

注**:** 通常はエラーは空欄のままとするが、この問では無回答と区別するために明示的に「X」を書け。

1.8 演算子順位法の応用

＿＿＿＿＿＿＿＿を生成する。 ＿＿＿＿＿＿＿＿＿＿(RPN)、＿＿＿＿ (postfix notation) とも呼ばれる。演算子をオペランドの後に書く。そのため（二項演算子しかない場合は）かっこが必要ない。

通常（中置記法）	**RPN**
12 + 34 * 56	＿＿＿＿＿＿
(12 + 34) * 56	＿＿＿＿＿＿

機械語と順序が同じであるため、式に対する（簡易）＿＿＿＿と考えることができる

演算子順位法では、還元が起こったときに、含まれている（かっこ以外の）終端記号を出力すれば良い

例 $a+b*(c+d)$ （さっきの入力例）

問 **1.8.1** 下の 2 つの入力例

1. $a+b*c$
2. $(a+b)*c$

について、教科書 p.45 表4.2 の表による演算子順位法が、さっきの入力例と同様に逆ポーランド記法を出力することを確認せよ
