8 構文解析の自動化

- ♣ [　　] 解析
- ♣ [　　] 解析

構文解析の復習

構文解析とは

与えられた記号列の文法的な構造を調べる

解析法の満たすべき条件

1. 記号列の走査は [　　] のみ

   ただし, k 記号の [　　] を許す (k は通常 1 か 2 程度)

2. [　　] は許さない

解析法の分類

[　　] 型解析 (top-down parsing, descent parsing)

[　　] 記号から始め, 生成規則の適用を繰り返して [　　] 記号列を導くことにより解析木を作る

[　　] 型解析 (bottom-up parsing, ascent parsing)

[　　] 記号列から始め, 生成規則の [　] の適用 ([　　]) を繰返して [　　] 記号を導くことにより解析木を作る

8.1 LL 解析

LL(k) 解析

再帰下降解析において, 再帰呼出しの代わりに [　　] を使う

適用可能な規則が複数ある場合の選択に, [　　] を用いる

この際に [　] 記号の先読みを許す

アルゴリズム

スタックに [　　] 記号をおく

while (スタックが [　　]) {

　スタックトップが

　　非終端記号: 解析表から得られる [　　] でその記号を書き換える

　　(適用できる規則がなければ [　　])

　　終端記号: スタックを [　　] し, 入力記号を [　　]

}

スタックと入力が両方とも [　] になれば成功

解析例

生成規則

1:	$E \to TE'$
2:	$E' \to \varepsilon$
3:	$E' \to +TE'$
4:	$T \to FT'$
5:	$T' \to \varepsilon$
6:	$T' \to *FT'$
7:	$F \to i$
8:	$F \to (E)$

LL(1) 構文解析表

スタックトップの非終端記号が A で, 次の入力記号が a のとき, 適用すべき生成規則 (の番号)

$A\backslash a$	i	$+$	$*$	$($	$)$	\$
E	1			1		
E'		3			2	2
T	4			4		
T'		5	6		5	5
F	7			8		

例えば, 一番左の一番上のマスの 1 は, スタックトップの非終端記号が [　] で, 入力記号列の先頭が [　] のときには, 1 番目の規則 ([　　]) を用いて E を書き換えることを表す.

$i*(i+i)$ の解析例

スタック

左端が [　　], 右端の $ は [　　] を表す

入力記号列

左端が [　　] 記号, 右端の $ は [　　] を表す

1:	$E \to TE'$
2:	$E' \to \varepsilon$
3:	$E' \to +TE'$
4:	$T \to FT'$
5:	$T' \to \varepsilon$
6:	$T' \to *FT'$
7:	$F \to i$
8:	$F \to (E)$

$A\backslash a$	i	$+$	$*$	$($	$)$	$\$$
E	1			1		
E'		3			2	2
T	4			4		
T'		5	6		5	5
F	7			8		

	スタック	入力記号列	規則
1	$E\$$	$i*(i+i)\$$	
2		$i*(i+i)\$$	
3		$i*(i+i)\$$	
4		$i*(i+i)\$$	
5			
6			
7			8
8	$(E)T'E'\$$	$(i+i)\$$	
9	$E)T'E'\$$	$i+i)\$$	1
10	$TE')T'E'\$$	$i+i)\$$	4
11	$FT'E')T'E'\$$	$i+i)\$$	7
12	$iT'E')T'E'\$$	$i+i)\$$	
13	$T'E')T'E'\$$	$+i)\$$	5
14	$E')T'E'\$$	$+i)\$$	3
15	$+TE')T'E'\$$	$+i)\$$	
16	$TE')T'E'\$$	$i)\$$	4
17	$FT'E')T'E'\$$	$i)\$$	7
18	$iT'E')T'E'\$$	$i)\$$	
19	$T'E')T'E'\$$	$)\$$	5
20	$E')T'E'\$$	$)\$$	2
21	$)T'E'\$$	$)\$$	
22	$T'E'\$$	$\$$	5
23	$E'\$$	$\$$	2
24	$\$$	$\$$	
25			(成功)

☆ 構文解析表に適用すべきルールが記入されていなければ, 構文解析は失敗, すなわち [　　]
.

例) スタックトップが E で入力の先頭が $+$ の場合, 構文解析表は [　　]
→ 文法エラー

☆ ここでは, 入力記号列が文法に適合している (文法エラーではない) ことが確かめられただけだが, 適用された生成規則の系列から解析木を作ったり文法構造を知ることができる. あるいは, 次の例のように, 生成規則中に「アクション」を定義することにより, コード生成等を行うこともできる.

☆ 解析表さえあれば, 構文解析は機械的に行える. さらに, 与えられた文法から解析表を [ ] することも可能 (付録参照)

LL(1) 解析しながらのコード生成

逆ポーランド記法の生成

文法規則中に [ ] でアクション (記号やコードの出力等) を書き, スタックから下ろすときにこれを実行する

1:	$E \to TE'$
2:	$E' \to \varepsilon$
3:	$E' \to +T\ [+]\ E'$
4:	$T \to FT'$
5:	$T' \to \varepsilon$
6:	$T' \to *F\ [*]\ T'$
7:	$F \to i\ [i]$
8:	$F \to (E)$

	スタック	入力記号列	規則	出力
1	$E\$$	$i*(i+i)\$$	1	
2	$TE'\$$	$i*(i+i)\$$	4	
3	$FT'E'\$$	$i*(i+i)\$$	7	
4	$\underline{i[i]}T'E'\$$	$i*(i+i)\$$		
5	$T'E'\$$	$*(i+i)\$$	6	
6	$*F[*]T'E'\$$	$*(i+i)\$$		
7	$F[*]T'E'\$$	$(i+i)\$$	8	
8	$(E)[*]T'E'\$$	$(i+i)\$$		
9	$E)[*]T'E'\$$	$i+i)\$$	1	
10	$TE')[*]T'E'\$$	$i+i)\$$	4	
11	$FT'E')[*]T'E'\$$	$i+i)\$$	7	
12	$\underline{i[i]}T'E')[*]T'E'\$$	$i+i)\$$		
13	$T'E')[*]T'E'\$$	$+i)\$$	5	
14	$E')[*]T'E'\$$	$+i)\$$	3	
15	$+T[+]E')[*]T'E'\$$	$+i)\$$		
16	$T[+]E')[*]T'E'\$$	$i)\$$	4	
17	$FT'[+]E')[*]T'E'\$$	$i)\$$	7	
18	$\underline{i[i]}T'[+]E')[*]T'E'\$$	$i)\$$		
19	$\underline{T'[+]}E')[*]T'E'\$$	$)\$$	5	
20	$E')[*]T'E'\$$	$)\$$	2	
21	$\underline{)[*]}T'E'\$$	$)\$$		
22	$T'E'\$$	$\$$	5	
23	$E'\$$	$\$$	2	
24	$\$$	$\$$		
25			(成功)	
			出力	$i\ i\ i\ +\ *$

8.2 LR 解析

LR 解析

構文解析ルーチンの [ ] に適している

LL 解析と同様, [ ] と [ ] を用いる

次にどの規則で還元するかがわかるように, スタックには記号と [ ] を交互に積む

基本動作は次の 2 つ

□: 入力記号と状態をスタックに積む

□: スタックの上部の記号列に対して生成規則を逆に適用

解析表

□ 表 $action[s, a]$: スタックトップの状態が □, 次の入力記号が a の時の動作

□ 表 $goto[s, A]$: スタックトップが □, その次の状態が s のときに, A の上に積む状態

アルゴリズム

スタックに初期状態 0 を積む
while (受理でもエラーでもない) {
  action 表を見る
    Si なら: 入力記号を □ し, □ を積む
    Rj なら: スタック上部を j 番目の生成規則で □ する
      goto 表を見て状態を積む
    acc なら: □
    空白なら: □
}

解析例

1: $E \to E + T$
2: $E \to T$
3: $T \to T * F$
4: $T \to F$
5: $F \to (E)$
6: $F \to i$

LR(1) 解析の解析表

状態	action						goto		
	i	$+$	$*$	$($	$)$	\$	E	T	F
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

$i * (i + i)$ の解析例

スタック: 右端が □

入力記号列: 左端が現在読んでいる記号

	スタック	入力記号列	動作	出力
1	0	$i * (i + i)\$$	s5	
2	$0i5$	$*(i + i)\$$	□	
3	□	$*(i + i)\$$	□	□
4	□	$*(i + i)\$$	□	
5	□	$(i + i)\$$	□	
6	□	$i + i)\$$	s5	
7	$0T2 * 7(4i5$	$+i)\$$	r6	
8	$0T2 * 7(4F3$	$+i)\$$	r4	□
9	$0T2 * 7(4T2$	$+i)\$$	r2	
10	$0T2 * 7(4E8$	$+i)\$$	s6	
11	$0T2 * 7(4E8 + 6$	$i)\$$	s5	
12	$0T2 * 7(4E8 + 6i5$	$)\$$	r6	
13	$0T2 * 7(4E8 + 6F3$	$)\$$	r4	□
14	$0T2 * 7(4E8 + 6T9$	$)\$$	r1	
15	$0T2 * 7(4E8$	$)\$$	s11	□
16	$0T2 * 7(4E8)11$	$\$$	r5	
17	$0T2 * 7F10$	$\$$	r3	
18	$0T2$	$\$$	r2	□
19	$0E1$	$\$$	acc	

1:	$E \to E + T$
2:	$E \to T$
3:	$T \to T * F$
4:	$T \to F$
5:	$F \to (E)$
6:	$F \to i$

状態	action						goto		
	i	$+$	$*$	$($	$)$	$\$$	E	T	F
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

解析表の作り方は省略

LR(1) 解析しながらのコード生成

逆ポーランド記法

還元する際に記号を出力

1: $E \to E + T$ で還元 $\cdots$ □ を出力

3: $T \to T * F$ で還元 $\cdots$ □ を出力

6: $F \to i$ で還元 $\cdots$ □ を出力

4 つ組記法

スタックを用いる

6: $F \to i$ で還元 $\cdots$ □ をスタックに積む

3: $T \to T * F$ で還元 $\cdots$ スタックから r と s をポップして □ を出力し, □ をスタックに積む

1: $E \to E + T$ で還元 $\cdots$ スタックから r と s をポップして □ を出力し, □ をスタックに積む

8.3 構文解析の自動化

「コンパイラコンパイラ」と呼ばれる処理系が開発されている

付録: LL(1) 解析表の作り方

各生成規則 $A \to s$ に対して, s の First 集合, A の Follow 集合を求める

$s \in (N \cup T)^*$ の First 集合は

$First(s) = \{a \in T \mid s \stackrel{*}{\Rightarrow} a\alpha, \alpha \in (N \cup T)^*\}$ で

$First(s) = First(s) \cup \{\varepsilon\}$ if $s \stackrel{*}{\Rightarrow} \varepsilon$

(つまり s から導出される語の先頭に来る終端記号の集合)

$A \in N$ の Follow 集合は

$Follow(A) = \{a \in T \mid S \stackrel{*}{\Rightarrow} uAav, u, v \in (T \cup N)^*\}$ で

$Follow(A) = Follow(A) \cup \{\$\}$ if $S \stackrel{*}{\Rightarrow} uA$

ただし \$ は入力記号列の末尾を表す記号

(つまり開始記号から導出される記号列の中で, A の次に来うる終端記号の集合)

例) 式の文法

$E \to TE'$

$E' \to +TE' \mid \varepsilon$

$T \to FT'$

$T' \to *FT' \mid \varepsilon$

$F \to (E) \mid i$

First 集合

$First(E) = First(T) = First(F) = \{i, (\}$

$First(E') = \{+, \varepsilon\}$

$First(T') = \{*, \varepsilon\}$

$First(\varepsilon) = \{\varepsilon\}$

$First(TE') = \{i, (\}$

$First(+TE') = \{+\}$

$First(FT') = \{i, (\}$

$First(*FT') = \{*\}$

$First((E)) = \{(\}$

$First(i) = \{i\}$

Follow 集合

$Follow(E) = Follow(E') = \{), \$\}$

$Follow(T) = Follow(T') = \{+,), \$\}$

$Follow(F) = Follow = \{+, *,), \$\}$

First 集合構成のアルゴリズム

以下を繰り返す

$s = \varepsilon$ なら $First(s) = \{\varepsilon\}$

$s \to \varepsilon \in P$ なら $First(s) = First(s) \cup \{\varepsilon\}$

$s = a\alpha$ $(a \in T)$ なら $First(s) = \{a\}$

$s \to a\alpha \in P$ $(a \in T)$ なら $First(s) = First(s) \cup \{a\}$

$s = A\alpha$ $(A \in N)$ なら

$\varepsilon \notin First(A)$ のとき $First(s) = First(A)$

$\varepsilon \in First(A)$ のとき $First(s) = (First(A) - \{\varepsilon\}) \cup First(\alpha)$

$s \to A\alpha$ $(A \in N)$ なら

$\varepsilon \notin First(A)$ のとき $First(s) = First(s) \cup First(A)$

$\varepsilon \in First(A)$ のとき $First(s) = First(s) \cup (First(A) - \{\varepsilon\}) \cup First(\alpha)$

Follow 集合構成のアルゴリズム

以下を繰り返す

$Follow(S) = Follow(S) \cup \{\$\}$

$A \to \alpha B\beta \in P$ $(B \in N, \beta \neq \varepsilon)$ なら $Follow(B) = Follow(B) \cup (First(\beta) - \{\varepsilon\})$

$A \to \alpha B \in P$ または $A \to \alpha B\beta \in P$ かつ $\varepsilon \in First(\beta)$ なら $Follow(B) = Follow(B) \cup Follow(A)$

解析表の構成

解析表 $M[A, a]$: スタックトップ (最左の非終端記号が A で次の入力記号が a のとき, 適用すべき生成規則の集合

各規則 $A \to s$ について

1. $a \in First(s)$ なら $A \to s$ を $M[A, a]$ に入れる
2. $\varepsilon \in First(s)$ かつ $b \in Follow(A)$ なら $A \to s$ を $M[A, b]$ に入れる

LL(1) 文法

すべての A, a について $M[A, a]$ の要素が高々 1 つであるような文法を LL(1) 文法という

1 記号先読みの LL 構文解析で, バックトラックなしに解析可能