コンパイラ補助資料　　　　　　佐々木晃

<u>LL(1)文法</u>　(中田育男著「コンパイラ」，オーム社 (p83～を一部改変))

これから定義するLL(1)文法は、たとえば

$A \to \alpha \mid \beta$

という生成規則で、αか「または」βのどちらかを選択するとき、そのときの<u>入力の先頭記号を１つ</u>見ることによって（後戻りの起こり得ない）<u>選択をすることができる</u>ような文法である。記号を１つ見るだけでなく、一般にｋ個の記号を見ることによってうまく選択できるような文法をLL(k)文法という。ここではLL(1)文法だけの説明をすることにする。

基本的な考え方は

$A \to \alpha \mid \beta$

で、αかβを選択するとき、そのときの入力の<u>先頭記号ａがαの先頭記号になり得るものであればαを選択</u>し、それが<u>βの先頭記号になり得るものであればβを選択</u>すればよいということであり、<u>後戻りがないためには</u>、αの先頭記号にもβの先頭記号にもなり得る共通のものが<u>なければ</u>よい。すなわち、αの先頭記号になり得る終端記号の集合をFirst(α)としたとき

$\mathrm{First}(\alpha) \cap \mathrm{First}(\beta) = \phi$

であればよい。ただし、これだけの条件では不十分である。$\beta = \varepsilon$あるいは$\beta \Rightarrow^* \varepsilon$のときはＡの後ろにくる記号(Follow(A)と書く）とαの先頭記号とにも共通のものがあってはならないことを示唆している。すなわち

$\mathrm{Follow}(A) \cap \mathrm{First}(\alpha) = \phi$

であればよい。

以上の考察から、以下のような定義が得られる。

文法$G=\{V_N, V_T, P, S\}$に関して次のものを定義する。

[定義]　記号列$\alpha \in (V_N \cup V_T)^*$と非終端記号$A \in V_N$について、First($\alpha$)とFollow(A)を次のように定義する。
$\mathrm{First}(\alpha) = \{ a \mid a \in V_T,\ \alpha \Rightarrow^* a \cdots \}$
ただし、$\alpha \Rightarrow^* \varepsilon$なら$\varepsilon \in \mathrm{First}(\alpha)$とする
$\mathrm{Follow}(A) = \{ a \mid a \in V_T,\ S \Rightarrow^* \cdots Aa \cdots \}$

ここで、「...」は任意の記号列を意味する。<u>First(α)はαの先頭の終端記号になり得るものの集合</u>であり、<u>Follow(A)は文形式の中でＡの直後の終端記号になり得るものの集合</u>である。これらの記号を使えば、生成規則$A \to \alpha$によってＡをαに展開できる場合には、そのときの入力の先頭記号ａが$a \in \mathrm{First}(\alpha)$であるか、$\varepsilon \in \mathrm{First}(\alpha)$であるときは$a \in \mathrm{Follow}(A)$でなければならない、ということができる。後者は$A \Rightarrow \alpha \Rightarrow^* \varepsilon$となる場合である。このようなａの集合はDirectorと呼ばれる。

[定義]　記号列$\alpha \in (V_N \cup V_T)^*$と非終端記号$A \in V_N$に対して、生成規則$A \to \alpha$があるとき、Director(A, α)を次のように定義する。
$\mathrm{Director}(A, \alpha) = \{ a \mid a \in V_T,\ a \in \mathrm{First}(\alpha)$　または
$(\alpha \Rightarrow^* \varepsilon$かつ$a \in \mathrm{Follow}(A)) \}$

<u>Director(A, α)はＡをαに展開すべきか判定するための終端記号（入力記号）の集合</u>である。

ところで、同じ入力記号ａがDirector(A, α)にもDirector(A, β)にも入っていたら、αとβのどちらをとるべきか決まらない。そのようなものが<u>ない</u>文法がLL(1)文法である。

[定義]　文法$G = \{ V_N, V_T, P, S \}$において、任意の$A \in V_N$とＡを左辺に持つすべての生成規則
$A \to \alpha_1 \mid \alpha_2 \mid \cdots \mid \alpha_n$
に対して、$\mathrm{Director}(A, \alpha_i),\ i = 1, \cdots, n$が共通部分を持たない、すなわち
$\mathrm{Director}(A, \alpha_i) \cap \mathrm{Director}(A, \alpha_j) = \phi,\ i \neq j$
ならば、文法ＧはLL(1)文法であるという。

ＬＬ（１）文法のチェックとFirst, Follow, Director

与えられた文法がLL(1)文法かどうか調べるためには、First, Follow,を求めて、最終的にDirectorを求めればよい。この計算のより正確なアルゴリズムは後述するが、文法が単純であればアルゴリズムを追うよりも、定義に従って考える方が簡単である。

<u>例：文法G2（開始記号E)</u>
$E \to TE'$　　(1)
$E' \to +TE' \mid \varepsilon$　　(2)
$T \to FT'$　　(3)
$T' \to *FT' \mid \varepsilon$　　(4)
$F \to (E) \mid i$　　(5)

まず、大前提として文法に**左再帰性がない**ことが必要である。また**左括りだしがなされていな**ければ、明らかに First 集合に交わりができてしまうので、LL(1)ではない。

○First

分かりやすいのは、F である。F から直接導出される、記号列(E), および i の先頭は、それぞれ、($\in V_T$, i$\in V_T$ であるので、

First(F) = { (, i}

が言える。次に、T'から導出される記号列、(1)*FT' ,および (2) ε であるが、(1)の先頭（すなわち First(*FT')）は *$\in V_T$ であり また、(2)は First の定義の「ただし」の部分にあるように、 ε も First 集合に含める。したがって、

First(T') = { * ，ε }

ほとんど同様に

First(E') = { + ，ε }

次に First(T)を考えるが、これは First(FT')に等しい。FT'の先頭は、まず F の先頭（すなわち First(F)）を考えれば良く、この場合は、

First(T) = First(FT')=First(F) ={ (, i}

（なお、もしも仮に First(F)に ε が含まれている場合は、T'の先頭も考えなければならないので、First(FT') = First(F − ε) ∪ First(T')である。First(T')には ε が含まれているので、最終的に First(FT')にも ε が含まれることになる。これは大事)

同様に

First(E) = First(TE') = First(T) = { (, i }

○ Follow

開始記号 E についてまず考える。Follow(E) に$を加えておく。これは、E の解析が終わった後、プログラムの終わりを示す記号$を読む、という意味である(詳しくはアルゴリズムの項を参照)。このほかに E の後ろに何が来るかを観察すると、文法の右辺に E が現れるのは(5)の(E)だけであり、この場合、E の後ろは)である。したがって、Follow(E) = {$,) }

Follow(E')について考える。E'が右辺に現れるのは、(1)の TE' および (2) の+TE'である。(1)の場合、E'の後ろにくるのは、左辺の E の後ろにくるものに等しいので、Follow(E)が含まれる。(2)の左辺は E'であるので、これは考えなくて良い。そのほかにE'は右辺に現れないので、結局、

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

である。

次に Follow(T)。T が右辺に現れるのは、(2)で + TE'である。T の後ろは E'であるから、まず、First(E')をみればよい。すると、First(E') = {+，ε}であるから、+は、T の直後にくるといえる。次に ε の場合を考えると、E' $\Rightarrow^*$ ε を表しているので、E'の後ろに何がくるかを考える。すなわち Follow(E')を考える。最終的に、

Follow(T) = ({ + ，ε } − { ε }) ∪{$,)} = { + , $,)}

また、E'と同様の方法で、

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

同様に F'に関しては、T→FT' に注目して Follow(F) = (First(T') − { ε }) ∪ (Follow(T')) = { *, +, $,) }

Director は First と Follow から、Director の定義に従って求めればよい。

注意：First には ε が含まれる可能性があるが、Follow や、Director には現れない。

Follow や Director には$が含まれる可能性があるが、First には現れない。

アルゴリズム

次に First と Follow を求めるアルゴリズムを述べる。Director は First と Follow から、Director の定義に従って求めればよい。

① First を求めるアルゴリズム

以下を、どの First にも新たに追加されるものがなくなるまで繰り返す。

a) First(ε) = { ε }

b) First(a α) = { a }, a$\in V_T$

c) if (ε が First(Y)に含まれない)

First(Yα) = First(Y)

else

First(Y α) = (First(Y)-{ ε })∪First(α)

d) X→ α ならば First(α)を First(X)に加える。

② Follow を求めるアルゴリズム
以下を、どの Follow にも新たに追加されるものがなくなるまで繰り返す。

a) Follow(S)に$を加える。S は開始記号、$は入力の最後の記号である。

b) $A \to \alpha B \beta$ ($B \in V_N$) なる生成規則について

(i) First(β)を Follow(B)に加える。ただし、$\varepsilon \in$First(β)のとき ε は加えない。

(ii) $\varepsilon \in$First(β)または$\beta = \varepsilon$ ならば、Follow(A)を Follow(B)に加える。

ここで、Follow(S)に$を加えているのは次のような理由による。一般に構文解析のアルゴリズムは、いままで読んだものと、その次に読むものとの関係を調べることによって解析を進めるように表現される。LL(1)文法に対する構文解析でも次の 1 つの記号が調べられる。したがって、開始記号 S に対応する記号列をすべて読んだ後でも、その次の記号を調べる必要があり得る。その特別な記号としてここでは$を使っているのである。

例：文法 G2（上述のものと同じ例）

$E \to TE'$ (1)

$E' \to +TE' \mid \varepsilon$ (2)

$T \to FT'$ (3)

$T' \to *FT' \mid \varepsilon$ (4)

$F \to (E) \mid i$ (5)

First については、アルゴリズムが示すように、終端記号が右辺の左端にあるものや開始記号から遠いほうから求めると収束が速い。Follow については、その逆に、開始記号から計算すると収束が速い。

First(F) = { (, i }　(5)に①の d), b) を適用
First(“(E)”)={ (}　を First(F)に加える。
First(i)={ i } を First(F)に加える。

First(T') = { *, ε }　(4)に①の d), a), b) を適用
First(*FT')= {*}, First(ε)={ ε }
を First(T')に加える。

First(T) = First(F) = { (, i }　(3)に①の d), c) を適用
First(FT')=First(F)を First(T)に加える。

First(E') = { +, ε }　(2)に①の d), a), b) を適用
First(+TE)={+}, First(ε)={ ε }
を First(E')に加える。

First(E) = First(T) = { (, i }　(1)に①の d), c) を適用
First(TE')=First(T)
を First(E)に加える。

Follow(E) = { $,) }　E に②の a), (5)の E に②の b) の(i)を適用
Follow(E') = Follow(E) = { $,) }　(1)の E'に②の b) の(ii)を適用
Follow(T) = { +, $,) }　(2)の T に②の b) の(i),(ii)を適用 (First(E')-{ ε }) ∪Follow(E')
Follow(T') = Follow(T) = { +, $,) }　(3)の T'に②の b) の(ii)を適用
Follow(F) = { *, +, $,) }　(4)の F に②の b) の(i),(ii)を適用 (First(T')-{ ε }) ∪Follow(T')

これにより、Director は次のようになる。

E に関する Director
Director(E, TE') = First(TE')=First(T) = { (, i }

E'に関する Director
Director(E', +TE') = First(+TE') = { + }
Director(E', ε) = Follow(E') = { $,) }

T に関する Director
Director(T, FT') = First(FT')=First(F) = { (, i }

T'に関する Director
Director(T', *FT') = First(*FT')={ * }
Director(T', ε) = Follow(T') = { +, $,) }

F に関する Director
Director(F, (E)) = First((E)) = { (}
Director(F, i) = First(i) = { i }

これらのうち、複数個の Director を持つものは E',T',F であるが、それらの Director 間には共通部分がないから、文法 G2 は LL(1)文法である。