7D-5

On some properties of an algorithm for constructing LL(1) parsing-tables using production indices

Keiichi YOSHIDA (College of Engineering Shizuoka Univ.) and Yoshiko TAKEUCHI (Shizuoka Technical School)

1.Introduction

We have proposed an algorithm for constructing LL(1) parsing-table using prduction indices. For convenience, this algorithm is named "algorithm H". This paper describes the properties of algorithm H. Further, based on its properties, it is understandable that H may be revised into such an algorithm as to point out non-LL(1)ness for a CFG.

2.Definitions

Let's N, Σ the sets of nonterminals and terminals of a CFG, G respectively, and S the start symbol of G. Refer [1] about notations used in this paper without the definitions.

We use five sets in algorithm H; FIRST, FOLLOW, END-FOLLOW, G, and P. They are defined as follows:

[definition 1]
FIRST of X is defined as follows:

$FIRST(X)=\{a \mid a\in\Sigma,\ X\overset{*}{\Rightarrow}a\alpha,\ \alpha\in(N\cup\Sigma)^*,\ X\in(N\cup\Sigma)\}$

[definition 2]
FOLLOW of A is defined as follows:

$FOLLOW(A)=\{a \mid a\in\Sigma,\ S\overset{*}{\Rightarrow}A\beta,\ \beta\overset{*}{\Rightarrow}a\nu,\ \nu\in(N\cup\Sigma)^*,\ A\in N\}$

[definition 3]
END-FOLLOW of B is defined as follows:

$END\text{-}FOLLOW(B)=\{A \mid A\in N,\ A\rightarrow\alpha B\beta,\ B\in N,\ \beta\overset{*}{\Rightarrow}\varepsilon\}$

where ε is a null string.

[definition 4]
Q is defined as follows:

$Q=\{A \mid A\overset{+}{\Rightarrow}\varepsilon\}$

[definition 5]
P is defined as follows:
$P=\{p_i \mid p_i$ is a production index$\}$

3.Algorithm H

It is assumed that a grammar inputted to the H does not contain an useless production at all. Set Q is obtained by using the algorithm of the references [3], thus the following does not occurs for n more than 1;

$A\underset{p_1}{\Rightarrow}\alpha_1\overset{*}{\Rightarrow}\varepsilon$

$A\underset{p_2}{\Rightarrow}\alpha_2\overset{*}{\Rightarrow}\varepsilon$

⋮

$A\underset{p_n}{\Rightarrow}\alpha_n\overset{*}{\Rightarrow}\varepsilon$

Construction of FIRST-table
[STEP 1]
The following operations are applied to the FIRST-table whose entries are zeros.

```
begin
  for each p_i such that A_i →(p_i) Y_i1 Y_i2 … Y_in and
                         Y_i1 Y_i2 … Y_in ≠ ε do
    begin j←1;
      repeat FIRST(A_i, Y_ij)←p_i;
             j←j+1;
      until Y_i,j-1 ∉ Q or j>n;
    end;
end.
```

[STEP 2]

```
begin
  repeat
    for each A∈N do
      for each B∈N then
        if FIRST(A,B)∈P then
          for each C∈N do
            if FIRST(B,C)∈P then
              FIRST(A,C)←FIRST(A,B);
  until no change occurs in the FIRST-table;
  for each A∈N do
    for each B∈N do
      if FIRST(A,B)∈P then
        for each a∈Σ do
          if FIRST(B,a)∈P then
            FIRST(A,a)←FIRST(A,B);
  if Q is empty then skip step 3 through step 8
end.
```

Construction of local-FOLLOW-table
[STEP 3]
The following operations are applied to the FOLLOW-table whose entries have nils.

```
begin
  for each p_i such that A_i →(p_i) Y_i1 Y_i2 … Y_in do
    begin j←1; k←j+1;
      while j<n do
```

```
        begin if Y_ij ∈N then
          repeat FOLLOW(Y_ij,Y_ik)←'*'
                k←k+1;
          until Y_ik-1 ∉Q or k>n;
          j←j+1; k←j+1
        end;
      end;
end.
```

[STEP 4]

```
begin
  for each A∈N do
    for each B∈N do
     for each a∈Σ do
      if FOLLOW(A,B)='*' and FIRST(B,a)∈P then
           FOLLOW(A,a)←'*';
end.
```

Construction of END-FOLLOW-table

[STEP 5]

The following operations are applied to the END-FOLLOW-table whose entries have nils.

```
begin
 for each production such that A_i→Y_i1 Y_i2···Y_in do
   j←n;
   repeat  if Y_ij ∈N then
                END-FOLLOW(Y_ij,A_i)←'*';
             j←j-1;
          until Y_ij+1 ∉Q or j=0;
end.
```

[STEP 6]

```
begin
  repeat
    for each A∈N do
      for each B∈N do
        for each C∈N do
          if END-FOLLOW(A,B)='*' and
             END-FOLLOW(B,C)='*' then
            END-FOLLOW(A,C)←'*'
  until  no change occurs in the END-FOLLOW table
end.
```

Construction of FOLLOW-table

[STEP 7]

```
 begin
   FOLLOW(S,$)←'*' { S:start symbol, $:end mark}
   for each A∈N do
     for each B∈N do
       for each a∈Σ do
         if END-FOLLOW(A,B)='*' and
              FOLLOW(B,a)='*' then
            FOLLOW(A,a)←'*'
 end.
```

Construction of Parsing-table

[STEP 8]

```
 begin
   for each A∈N do
     if A∈Q then
       for each a∈Σ do
          if FOLLOW(A,a)='*' then
            FIRST(A,a)←P_i such that A ⇒(P_i) α ⇒* ε
 end.
```

4.Properties of algorithm H

Before discussing the properties of algorithm H, symbols used in it are explained.

T_{11} :FIRST-table constructed by STEP 1
T_{12} :Additional part to T_{11} by STEP 2
T_1 :$T_{11}+T_{12}$
T_2 :Additional part to T_1 by STEP 8

[property 1]
When STEP 1 is applied to G, if $T_{11}(A,X)=\{p_1, p_2, \cdots, p_n\}(n\geqq 2)$, then G is not LL(1), where $A\in N$, $X\in(N\cup\Sigma)$, and p_i is a production index.

[property 2]
When STEP 2 is applied to G, if $T_{12}(A,X)=\{p_1, p_2, \cdots, p_n\}(n\geqq 2)$, then G is not LL(1), where $A\in N$, $X\in(N\cup\Sigma)$, and p_i is a production index.

[property 3]
When STEP 3 through 7 are applied to G, even if two or more entries into a table-element occur, it can not be concluded that G is not LL(1).

[property 4]
When STEP 8 is applied to G, if $T_2(x,a)=\{p_1, p_2, \cdots, p_n\}(n\geqq 2)$, then G is not LL(1).

[property 5]
At least one of [property 1], [property 2] or [property 4] is true if and only if G is not LL(1).

5.Conclusion

The some properties of algorithm H have been cleared in this paper. Further, based on its properties, it is understandable that H may be revised into such an algorithm as to point out the non-LL(1)ness of input grammars.

We would particularly like to thank Prof. K.INOUE at Science Univ. of Tokyo who gave much valuable advices to us.

References

[1] YOSHIDA, TAKEUCHI and MATSUDA : A constructing method of LL(1) parsing table. 27th Nat. Conf. IPS Japan, 7E-2, (1983)
[2] YOSHIDA and TAKEUCHI: A constructing method of LL(1) parsing table using production indices. Under contribution.
[3] Robin Hunter : The Design and Construction of Compiler — , John Wiley & Sons. (1981)