知識工学と人工知能 48－10
（１９８６・９・１８）

音声理解システムにおける単語予測方式
── island driven 法について ──

渦原 茂　小林 豊　新美 康永
（京都工芸繊維大学工芸学部）

本稿では、現在我々が開発中の音声理解システムにおける単語予測の方式について報告する。本方式は島駆動(island driven) 法に基づいており、文法規則によって与えられた言語情報を用いて、認識された単語列（島）の右または左に接続可能な単語を予測する。また２つの単語列が結合可能かどうかについても調べることができる。

単語予測のプログラムは、ＢＵＰ(Bottom-Up Parser in Prolog)で用いられた方法を応用して、文脈自由文法あるいはＤＣＧ(Definite Clause Grammar) を用いて記述された文法規則をPrologのプログラムに変換することによって得られる。本稿では、その変換方法と得られたプログラムのアルゴリズムについて説明する。

A METHOD FOR WORD PREDICTION IN A SPEECH UNDERSTANDING SYSTEM BASED ON ISLAND PARSING

Shigeru UZUHARA　Yutaka KOBAYASHI　Yasuhisa NIIMI
Kyoto Institute of Technology
Matsugasaki, Sakyo-ku, Kyoto 606, Japan

In this paper, we describe a method for word prediction in a speech understanding system which we are developing. This method is based on island driven strategy. The word predictor is able to parse any consecutive sequence of word matches (i.e. island) and predict words which should be syntactically located at either end of the island. It is able to examine if two islands which are adjacently located acoustic data stream can be merged.

An implementation of the word predictor is based on BUP-like method. We compile a grammar defined in a context free grammar or a definite clause grammar into a Prolog program as the word predictor. In this paper, We describe a compiling technique and an algorithm of the word predictor.

1．はじめに

音声理解システムでは、認識対象となる語彙・構文などを限定しプラグマティクスを含めた言語情報を利用する。本稿では自然言語処理におけるＢＵＰ[1]を音声理解システムの言語処理に応用する方法について述べる。

我々が現在開発中の音声理解システムは、音響処理部、単語認識部、言語処理部、及び制御部の４つのコンポーネントからなり、各々が協力し合いながら音声を認識する。その認識の過程は、ワードスポッティングにより認識されたキーワードを島(island)として、島の左右に接続しうる単語によって島を拡張しながら発話全体を認識する、いわゆる島駆動(island-driven)方式に基づいている。言語処理部は音声中の任意の部分単語列を構文解析し、その単語列の左右両側に接続しうる単語を予測する。また、２つの島が構文的に結合可能かどうかについても調べる。

我々は、言語処理部の実現方法として、文脈自由文法（ＣＦＧ）あるいは Definite Clause Grammar（ＤＣＧ）[2]を用いて記述された文法規則を言語処理部のプログラムとしてPrologに変換する。変換によって得られるプログラムの構文解析は基本的に上昇型(bottom-up)の戦略を用い、かつ島駆動型の解析(island parsing)を行う。本稿では、文法規則からPrologへの変換方法とそれによって得られるプログラムのアルゴリズムについて説明する。

第２節では音声理解システムの言語処理部が持つべき機能とその制御について説明する。第３節では構文解析・単語予測するための基本となるアルゴリズムについて説明し、第４節でそれを島駆動型に解析するPrologのプログラムとして実現する方法について説明する。第５節では２つの島を構文的に結合する方法について説明する。第６節ではＤＣＧを扱うときの問題点やその解決法について考察を行う。

2．音声理解システムにおける言語処理

一般に、音声理解システムにおける言語処理の解析方法が自然言語処理の解析方法と異なる点は、自然言語処理では一文全体の解析を非可分な手続きとして処理が進むのに対して、音声理解システムでは他の処理（音響処理など）と共同で認識処理を進めるため、解析は部分的な文の断片を対象とし、その処理は他の処理の進行に応じて制御しなければならないことである。

島駆動方式では、音響データ列の中の任意の（比較的明瞭な）部分からいくつかのキーワードを見つけ出し、その単語を新しい島として左側あるいは右側に単語を接続し、島を拡張しながら認識を進める。言語処理部には次の３つの機能が要求される。

(1) 島の生成(seed)：　スポットされた単語から新しい島を作る。
(2) 島の拡張(extend)：　島の左右に構文的に接続可能な単語を予測し、単語認識部で認識された単語を接続した新しい島を作る。
(3) 島の結合(collision)：　２つの島が構文的に接続可能かどうかを調べ、可能であれば２つを接続した新しい島を作る。

図１は制御部と言語処理部との関係を示したものである。制御部は単語の識別スコア（音響信号と単語の音素列との一致度）などをもとにしたヒューリスティックな評価関数に基づいて、次に拡張すべき島あるいは結合の可能性のある２つの島を決定する。各島には番号が付けられており、島の番号を言語処理部に送ることにより、島の拡張、あるいは結合を要求する。そしてどの島を拡張してどの島が作られたかといった対応を番号の表 (control_table)にして管理する。

言語処理部では、各島の構文情報を表(island_table)で管理しており、受け取った番号から対応する島の構文情報を取り出し、それをもとにして制御部の要求に答える。そして要求に応じて作られた新しい島の番号を制御部に知らせ (hand_to)、新しい島の構文情報を番号と対応づけて表に格納する(make_island)。

3．単語予測のアルゴリズム

本節では、任意の部分単語列に接続可能な単語を予測するための構文解析のアルゴリズムについて考える。

以下の節ではＣＦＧを対象とし、ＤＣＧの場合については６節で触れる。また文法規則を記述する制約として次の２つの条件を置く。

(1) 空系列を生成する規則を含んでいてはならない。
(2) 周期的(cyclic)な規則の集合を含んでいてはならない。

周期的な規則の集合とは、ある非終端記号からの記号系列の生成過程において、他の記号を生成することなしに自分自身の生成に至るような一連の規則のことである。これらの制約は上昇型の戦略を採用し

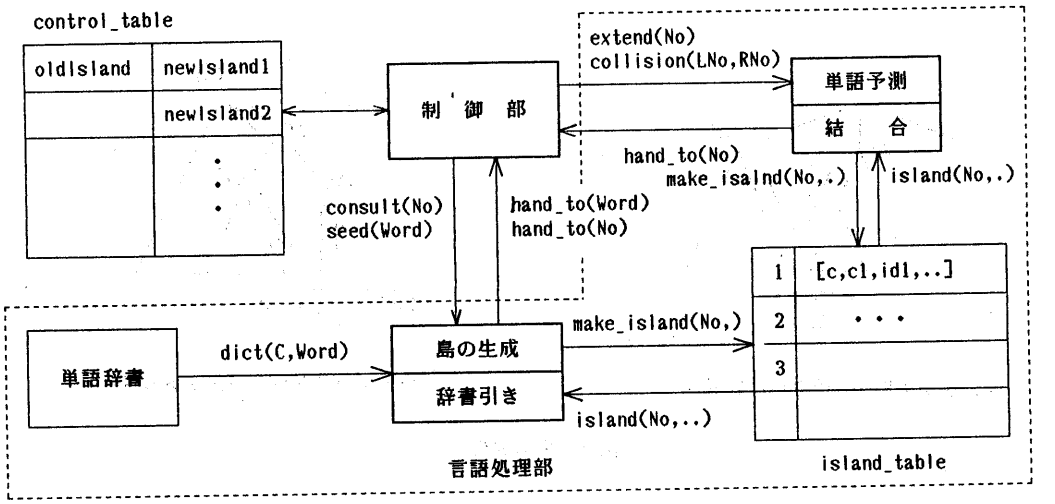


図1　言語処理部の制御

ていることによるが、一般に下降型(top-down)の戦略では許されない左再帰的な規則はそれが周期的でない限り含まれていても構わない。

また文法規則は次のような規則の集合で表す。

```
c --> c1 c2 ••• cn (n>=1)        (1)
c --> a                          (2)
```

ここで、c,c1,c2,..,cn は非終端記号、aは終端記号（単語）を表す。こうすることによって規則の部分と単語辞書の部分を分けて考えることができる。特に(2)の左辺の非終端記号 cは単語クラスを意味し、以下では、このような非終端記号 cに対して次の述語が成り立つ。

```
part_of_speech(c).               (3)
```

3.1 リンク関係

ここで説明するリンク関係は、ＢＵＰでいうリンク関係を拡張したものである。ある非終端記号 Xがどの非終端記号 Yの構文木の左端（右端）の子孫と成り得るかを表したものを左（右）リンク関係と呼び、

```
left_link(X,Y).
right_link(X,Y).                 (4)
```

と表す。これは文法規則の集合が与えられた時点で事前に計算しておく。任意の文法規則において、右辺の左端（右端）の非終端記号と左辺の非終端記号との間にはリンク関係がある。これらの基本的なリンク関係の反射的かつ推移的な閉包が求めるべきリンク関係である。

ＢＵＰでは上昇型の解析の過程において無駄な規則の適用を避けるためにリンク関係を用いている。我々の単語予測においても、リンク関係は同様な役割を果たすが、それ以外にも重要な役割を果たす。

3.2 単語予測

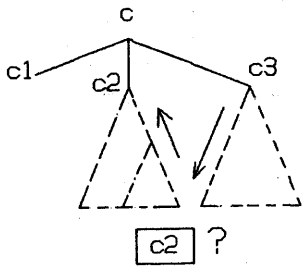
上昇型の解析においては、文法規則の右辺の解析が完了するとその左辺の非終端記号が生成される。そしてこの新しく生成された非終端記号を右辺に含むような文法規則の適用がさらに行われる。もしその規則の右辺に他の非終端記号を持っていれば、それらの記号は共起する構文として予測することができる。これらをトップダウン予測と呼ぶ。

まずスポットされた単語の右側にくる単語を予測する例を考えてみよう。スポットされた単語が単語クラスc2で、c2を右辺に含む文法規則

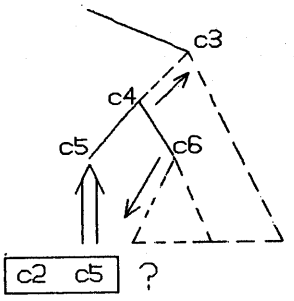
```
c --> c1 c2 c3                   (5)
```

が与えられていたとすれば、(5) よりc2の右側にはc3という構文が予測され、そのc3の左端に位置しうる単語が求める予測単語である（図２の (a)）。これはトップダウン予測として、c3のリンク関係より、

```
left_link(X,c3),part_of_speech(X).  (6)
```



(a) トップダウン予測がない場合



(b) トップダウン予測がある場合

図2　島の拡張

となるX を求めることに相当する。このような規則の適用をあらゆる可能性が尽くされるまで繰り返すことによってc2に続きうるすべての単語を求めることができる。

上で予測された単語のうち単語認識部で認識された単語から単語クラスc5が得られたとして、次に'c2 c5' からなる単語列の右側にくる単語を予測する２つの例を考えてみよう。

まず右辺にc5を含む文法規則

```
c3 --> c5                    (7)
```

が与えられていたとする。(7) の適用はc3の生成を意味し、トップダウン予測されている構文c3が完成したことになる。これをトップダウン予測c3に到達したという。トップダウン予測に到達した場合、その予測を与えた規則の右辺のまだ使っていない部分の非終端記号が新しいトップダウン予測の対象となる。もしその右辺にトップダウン予測の対象となる非終端記号が残っていなければ、左辺の非終端記号（例ではc ）を生成する。

次に右辺にc5を含む文法規則として

```
c4 --> c6 c5 c7              (8)
c4 --> c5 c6                 (9)
```

が与えられていたとする。c5はc3の左端として予測されたのだから、(8) の適用はc5が右辺の左端にないことから矛盾する。また(9) の適用が矛盾しないためには、c4がc3の左端となりうること(left_link(c4,c3) の成立)が必要である。このリンク関係の使用はLINGOL[3] でいえばオラクル(oracle)と呼ばれているものに相当する。もし(9) が適用できるならば、トップダウン予測としてc6から単語を予測する（図２の (b)）。

このようにトップダウン予測（例ではc3）がある場合は、ＢＵＰの用いるアルゴリズムと同じ左隅構文解析法(left-corner parsing) [4] に基づいた文法規則の適用が行われる。これに対し先のc2が得られたときは、このようなトップダウン予測がないのでc2を右辺に含む規則であれば適用できた。しかし、例えば次のような規則

```
c2 --> c1 c2                 (10)
```

がある場合、右辺のc2より左辺のc2を生成し、この適用を繰り返して停止しない。そこでトップダウン予測がない場合は、規則を適用して推移する代わりに、c2が得られたときにright_link(c2,X)となるX を求め、X を右辺の右端以外の位置に持つ文法規則のみ適用する。先のc2の例では、X=c2が得られて(5) を適用するというのが正確である。

本節では右方向への単語予測について考えたが、左方向もまったく同様に考えることができる。

4．島駆動方式の実現

4.1 島の表現方法

３節で述べたように島は文法規則に従って拡張されるが、島を拡張する際に島の部分単語列を構文解析しなおすのではなく、各島に対して拡張の過程で行われた構文解析の履歴を記録しておいて次の拡張

の際に解析を再開できるようにする。そのためには解析に使用された文法規則と各々の規則がどこまで使われているかを記録する必要がある。それには、次のようにすべての文法規則の右辺の各記号の間に他の位置と区別するための識別子を与える。(1) の規則に対して、

c --> id0 c1 id1 c2 id2 ...idn-1 cn idn (1')
(ただし id0 = idn = c)

のようにidという識別子を与える。構文解析の履歴は、部分単語列の端の単語のクラス(c) と、トップダウン予測とこれらの識別子を組にしたものの系列 (<ci,idi>,<cj,idj>,...) を合わせたリスト

[c, ci,idi, cj,idj, ...]

で表す。そして１つの島は左右方向の２つの構文解析の履歴と左右端の単語のクラスによって表す。例えば３節の(5) の規則は、

c --> c1 id1 c2 id2 c3 (5')

となるが、'c2 c5' からなる島は、

左側：[c2,c2,id1] 右側：[c5,c3,c]

と表される。左側はc2をトップダウン予測に左方向へ単語を予測しながら島を拡張し、c2に到達すれば規則の右辺の id1より左側の部分の解析 (c1の予測) を進めなければならないことを表している。右側はc3をトップダウン予測に島を拡張し、c3に到達すれば cを生成しなければならないことを表している。

4.2 島駆動型への拡張

３節で述べた方法を島駆動型の構文解析で使うには、島の両方向の解析の履歴に対して文法規則を適用するようにする。しかし、常に両方向の履歴に注意する必要はない。島を拡張している方向と反対方向の履歴に注意するのは、トップダウン予測がない状態で非終端記号が生成される場合で、その非終端記号を右辺に含む規則を適用したことを反対側の履歴にも知らせる必要がある。

例えば、３節でスポットされた単語c2を右に拡張する例では、(4')の適用は左方向の履歴([c2])に対してトップダウン予測と識別子の組<c2,id1>を追加して、左の履歴を[c2,c2,id1] とする必要がある。

4.3 文法規則からPrologへの変換

ここでは、３節で文法規則に与えたような手続き的意味を、文法規則からPrologのプログラムに変換することによって実現する方法を述べる。

変換によって得られるPrologのクローズには、非終端記号を述語名に持つもの（タイプ１）と、識別子を述語名に持つもの（タイプ２）の２種類がある。どちらのタイプのクローズにも、入力引数として、第１引数に解析の方向を、第 2,3引数に島の左右の履歴を渡す。さらにタイプ１のクローズは、第 4引数に識別子を、第 5,6引数には拡張によって変化した島の左右の履歴を返す。

このような２種類のクローズがあるのは、島の拡張を、上昇型に解析する部分とトップダウン予測を得て単語を予測する部分とに分けるためである。

(1')の文法規則を以下の①～⑦のようなPrologのプログラムに変換する。ただし、以下のプログラム中で、identifier(X) は Xが識別子かどうかを、non_terminal(X) は Xが非終端記号かどうかを判定する。new_island(No)は島の新しい番号をNoに返す。hand_to(X)はX の値を制御部に渡す。make_island(No,L,R) は島を番号で登録し、島の構文解析の履歴はその番号からisland(No,L,R)によって取り出すことができる。apply(P,A1,A2,..) はA1,A2,..を引数として述語 Pを呼び出す。

各非終端記号c に対して、

①
```
c(right,L,[],ID,L1,[]) :
    right_link(c,X),
    apply(X,right,L,non,ID,L1,[]).
c(left,[],R,ID,[],R1):-
    left_link(c,X),
    apply(X,left,non,R,ID,[],R1).
```

②
```
c(right,L,[c,CIR],ID,L1,R1) :-
    non_terminal(C),
    apply(C,right,L,R,ID,L1,R1).
c(left,[c,CIL],R,ID,L1,R1) :-
    non_terminal(C),
    apply(C,left,L,R,ID,L1,R1).
```

③
```
c(right,L,[c,IDIR],ID,L,R) :-
    identifier(ID).
c(left,[c,IDIL],R,ID,L,R) :-
    identifier(ID).
```

①はトップダウン予測がない状態で非終端記号が生成された場合に適用され、リンク関係より非終端記号X を求め、X に対応する⑥のクローズを呼び出す。このとき⑥以外のタイプ１のクローズを呼び出さないようにnon という定数を与える。②、③はトップダウン予測に到達するような非終端記号c が生成された場合に適用される。②はこの到達によって文法規則の右辺の解析が完了する場合で、それによって生成される非終端記号C に対応するクローズを呼び出す。③はまだ解析されていない部分が右辺に残っている場合で、その部分を示す識別子を値として返す。

n=1 の文法規則の右辺に対して、

```
④  c1(right,L,[GIR],ID,L1,R1) :-
       left_link(c,G),
       c(right,L,[GIR],ID,L1,R1).
   c1(left,[GIL],R,ID,L1,R1) :-
       right_link(c,G),
       c(left,[GIL],R,ID,L1,R1).
```

n>1 の文法規則の右辺に対して、

```
⑤  c1(right,L,[GIR],idi,L,[GIR]) :-
       left_link(c,G).
   cn(left,[GIL],idi-1,[GIL],R) :-
       right_link(c,G).

⑥  ci(right,L,non,idi,L1,[]) :-
       append(L,[ci,idi-1],L1).  (ただしi<>n)
   ci(left,non,R,idi-1,[],R1) :-
       append(R,[ci,idi],R1).    (ただしi<>1)
```

④、⑤はトップダウン予測G がある場合の文法規則の適用に相当し、規則の右辺の端にある非終端記号に対してのみ用意する。⑥はトップダウン予測がない場合の文法規則の適用に相当し、①から呼び出される。これは、例えば右方向の解析の場合、右辺の右端を除く非終端記号に対して用意する。

n>1 の規則の識別子idi(i=1～n-1)に対して

```
⑦  idi(right,L,R) :-
       right_pred(ci,L,[ci+1,idi+1|R]).
   idi(left,L,R) :-
       left_pred(ci,[ci,idi-1|L],R).
```

ここで、right_pred(G,L,R) (left_pred(G,L,R))は識別子の右（左）の非終端記号 Gからトップダウン予測して右側（左側）にくる単語の予測する。

```
right_pred(G,L,R) :-
    left_link(C,G),part_of_speech(C),
    new_island(No),hand_to(No),
    make_island(No,L,[C|R]).
left_pred(G,L,R) :-
    right_link(C,G),part_of_speech(C),
    new_island(No),hand_to(No),
    make_island(No,[C|L],R).
```

また、n>2 の規則の識別子に対して、

```
adjacent(idi,idi+1).   (ただし i=1～n-2)
```

を用意する。adjacentは５節の島の結合で使用される。

4.4 制御部とのインターフェース

制御部からの要求は次のように実現できる（図１参照）。

まずスポットされた単語(Word)より新しい島を作る要求は、述語seedで実現される。。

```
seed(Word) :-
    dict(C,Word),
    new_island(No),hand_to(No),
    make_island(No,[C],[C]),fail.
```

dict(C,Word)は単語辞書でWordの単語クラスC を求める。

右または左に島を拡張する要求は、述語right_extend,left_extendで実現される。

```
right_extend(No) :-
    island(No,L,[C|R]),!,
    apply(C,right,L,R,ID,L1,R1),
    apply(ID,right,L1,R1),fail.
left_extend(No) :-
    island(No,[C|L],R),!,
    apply(C,left,L,R,ID,L1,R1),
    apply(ID,left,L1,R1),fail.
```

islandによって拡張すべき島の番号(No)から島の履歴を取り出し、拡張する方向の履歴のリストの先頭の非終端記号からタイプ１のクローズを最初のappl

y で呼び出して上昇型の解析を行い、それによって得られる識別子IDから次のapply でタイプ２のクローズを呼び出して単語予測を行う。

右または左に予測された単語は、述語right_consult,left_consultで取り出すことができる。

```
right_consult(No) :-
    island(No,_,[CI_]),!,
    dict(C,Word),hand_to(Word),fail.
left_consult(No) :-
    island(No,[CI_],_),!,
    dict(C,Word),hand_to(Word),fail.
```

５．島の結合

本節では、２つの島の構文的な結合可能性は２つの島の結合する部分の構文解析の履歴から調べることができることを述べる。

図３の２つの島'c2 c3','c4 c5' の結合は次の２つのステップに分けて考えることができる。

(1) ２つの島が結合可能ならば、結合部分の隣合う単語 c3,c4の接点となる識別子idが少なくとも１つ存在する。
(2) (1) の結合をした時点の左島の右の履歴と右島の左の履歴について、

a) 一方でトップダウン予測されている構文木（左のa2）の部分木となるものが片方にある（右のa2）、
b) 図４のように同一の構文木で結合する。

以上のことを２つの構文解析の履歴に沿って調べる。

上記のアルゴリズムをPrologで表すと次のようになる。

```
collision(LNo,RNo) :-
    island(LNo,LI,[Lc,Lr]),
    island(RNo,[RcIRI],Rr),!,
    apply(Lc,right,LI,Lr,ID,LI1,Lr1),
    apply(Rc,left,RI,Rr,ID,RI1,Rr1),
    connect(LI1,Lr1,RI1,Rr1,L,R),
    new_island(No),make_island(No,L,R),
    fail.
```

２つのapplyは上記のステップ(1) の実行に相当し、接点となる識別子IDを求めている。次のconnect がステップ(2) の実行に相当する。

connect は次のようになる。第1～4引数に左島と右島の両方向の解析の履歴をそれぞれ渡し、第 5,6 引数に結合した島の履歴を返す。

```
①  connect(LI,[],RI,Rr,L,Rr) :-
       append(LI,RI,L).
   connect(LI,Lr,[],Rr,LI,R) :-
       append(Rr,Lr,R).
②  connect(LI,[Lc,LidILr],[Rc,RidIRI],Rr,L,R)
     :-
       adjacent(Rid,Lid),
       connect(LI,Lr,RI,Rr,L,R).
```

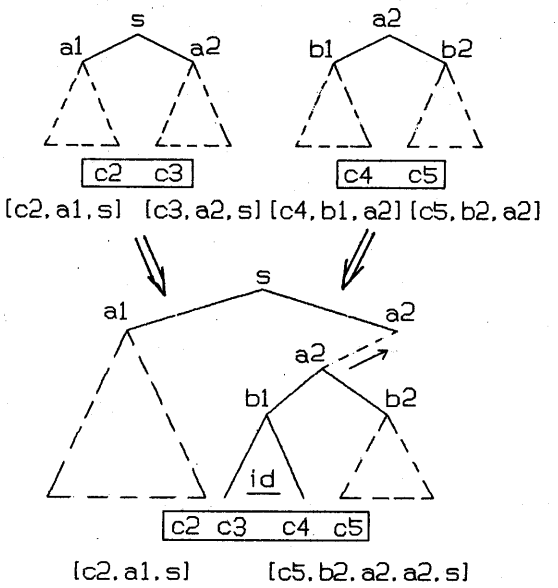


図３　島の結合

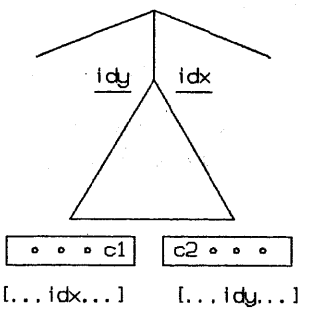


図４　同一の構文木での結合

③ connect(Ll,[Lc|Lr],[Rc,Rid|Rl],Rr,L,R) :-
non_terminal(Rid),
left_link(Rid,Lc),
connect(Ll,[Lc|Lr],Rl,Rr,L,R).
connect(Ll,[Lc,Lid|Lr],[Rc|Rl],Rr,L,R) :-
non_terminal(Lid),
right_link(Lid,Rc),
connect(Ll,Lr,[Rc|Rl],Rr,L,R).

③がステップ(1) のa)に、②がb)にそれぞれ相当する。①は②③を繰り返してどちらかの履歴が尽きた場合で、２つの島が結合可能なことを意味する。結合によって島を作るには、左島の左側の履歴を新しい島の左側の履歴に、右島の右側の履歴を新しい島の右側の履歴にすればよい。ただし、例えば図３のように右の島の構文木が左の島の構文木の下に入る場合（connect でいうと第 3引数が[]になって履歴が尽きた場合）、左の島の右側の残った履歴（[a2, s]）を新しい島の右側の履歴に加える。

６．ＤＣＧへの拡張

本節では、ＣＦＧからＤＣＧへ拡張する場合の問題点とその解決方法について述べる。

ＤＣＧでは、非終端記号に任意個の引数を持つことができ、右辺には任意個のPrologのプログラム（補強項）を書くことができる。しかしここで問題点が２つある。

第一に、非終端記号が引数を持つことからリンク関係も引数の情報を扱えるように拡張することが考えられる。それによって単語予測などに意味的制約を課すことができる。例えば、

c(X)-->c1(Y),c2(Z),{p(X,Y,Z)}.

という文法規則から基本的なリンク関係、

left_link(c1(Y),c(X)) :- p(X,Y,Z).

が考えられる。しかし、推移律を適用してこの拡張したリンク関係を文法規則と補強項のプログラムが与えられた時点で計算できるとは限らない。

第二に、右辺の任意の場所から任意の方向へ解析するため、補強項をいつ実行すれば良いか。例えば、非終端記号c2(X) が生成されたとき、

c(X)-->c1(X),c2(X),c3(Y),{p(X,Y)}.

という文法規則を適用したとする。そのとき、補強項p(X,Y)の実行が考えられるが、引数Y の値が決っていないと停止しないようなp が与えられていた場合、X の値が決っていても実行が停止しないという事態が起こる。

今のところ我々は、引数で受渡しされる情報を簡単なセマンティック・マーカに限定することで補強項のプログラムを単純化して、これらの問題を避けている。

７．おわりに

本稿では、ＣＦＧあるいはＤＣＧで記述された文法規則から島駆動方式に基づいた音声理解システムにおける言語処理部のプログラムを実現する方法について述べた。

今後、言語処理部を音声理解システムの中に組み込んで実験・評価を行う予定である。

参考文献

[1] Matsumoto,Y.,et al.,"BUP: A Bottom-Up Parser Embedded in Prolog," New Generation Computing, vol.1,no.2, pp.145-158, 1983.
[2] Pereira,F.C.N. and Warren,D.H.D., "Definite Clause Grammars--A Survey of the Formalism and a Comparison with Augmented Transition Networks," Artificial Intelligence 13, pp.231-278, 1980.
[3] Pratt,V.R.,"LINGOL--A Progress Report," 4th IJCAI, pp.422-428, 1975.
[4] Aho,A.V. and Ullman,J.D.,:The Theory of Parsing, Translation, and Compiling, vol.1, Parsing, Prentice-Hall, 1972.