拡張 CFG とその構文解析法 YAPX について†

林 達也††

筆者は先に，横型下降方式に基づく構文解析を論理型言語を用いて実現する方法 YAP について述べた．本論文では，YAP をベースにした上で，extraposition grammar (XG) や BUP-XG と同様に自然言語の外置現象を考慮して，トレース（痕跡）を直接的に指定できるように文法記述形式を拡張し，それに基づいて横型下降方式で構文解析を行う方法 YAPX について提案する．XG や BUP-XG と比較した場合，YAPX では文法記述形式がより一層拡張されている．YAPX は次のような特徴を持っている．（1） BUP-XG と同様に，文法記述者にとってより自然な方法と考えられるスラッシュカテゴリ記法を用いている．（2） スラッシュカテゴリには任意の構文記号列が許される．（3） スラッシュカテゴリのギャップ指定，非ギャップ指定が行える．（4） 規則の右辺の任意の位置にスラッシュカテゴリを指定できる．（5） 等位構造を考慮して，外置された単語に複数のトレースを対応づけることができる．このために，スコープやスラッシュに加えてドメインなる概念を導入している．上述した機能を実現するため，YAPX では解析スタック 1 対に加えて省略スタックと呼ばれる 1 対のスタックを用いる．ドメイン等の制御は省略スタックにより行われる．

1. まえがき

筆者は先に，横型下降方式に基づく構文解析を論理型言語を用いて実現する方法 (YAP) について述べた[1]．そこで許される文法記述形式は，基本的には文脈自由文法 (CFG) であるが，ただ，論理型言語へ展開されることを考慮して，利用者が自由にユーザ定義の引数や述語を導入することができ，構文処理と意味処理を融合させることができた．

本論文では，Pereira の extraposition grammar (XG)[2] や BUP-XG[3] と同様に，自然言語の外置現象を考慮してトレース（痕跡）を直接的に指定できるように CFG を拡張し，それに基づいて横型下降方式で構文解析を行う方法 YAPX について提案する．なお，ユーザ定義の引数や述語を自由に導入できる点に関しては YAP や XG, BUP-XG の場合と同様である．XG をさらに一般化したものに gapping grammar[4)~6)] があるが，性能上問題があるのでここでは考慮しないことにする．

CFG の拡張の仕方について，まず XG や BUP-XG が採っている方法を最初に見てみよう．

まず，XG では，

$A\alpha_1 \cdots \alpha_2 \cdots \alpha_n \rightarrow \beta$

なる規則が許される．ここで，A は非終端記号，α_i $(1 \leq i \leq n)$ および β は終端記号と非終端記号からなる記号列で $\alpha_1 \equiv \varepsilon$ でもよい．また，「…」は「ギャップ」と呼び，任意の構文記号列を表す．「$\alpha_i \cdots \alpha_{i+1}$」の意味は，$\alpha_i$ と α_{i+1} の間にギャップがあっても（無くても）よいことを表す．同時に，α_i (α_{i+1}) を構成する記号間には，ギャップがあってはならないことを表す．これにより，例えば，

rel→relm s

relm…np→relpn

なる規則を用意すれば，「The mouse that the cat chased…」のような関係代名詞節を直接的に記述することができる．しかし，XG には次のような問題がある．

（1） 言語学における複合名詞句制約 (complex noun phrase constraint) を満足させるために，非終端記号 open および close を導入して，トレースが出現する範囲を限定しなければならない．

（2） 規則中でギャップ指定あるいは非ギャップ指定が行えるにもかかわらず，非終端記号の場合は非ギャップ指定の際もギャップ指定と何ら区別なく処理されている．

（3） 外置された単語に対して，1 個のトレースしか対応付けられない．

次に BUP-XG では，規則の右辺に，第 1 要素を除いて

$B../C$

を記述することができる．ここで，B, C は非終端記号である．C はスラッシュカテゴリと呼ばれる．$B../C$ の意味は，B を根とする部分解析木において，C から導出される入力部分列がトレースになることを示している．BUP-XG ではこのように，GPSG[7] のスラッシュカテゴリ記法を用いている．BUP-XG の問題

† YAPX: An Extended Context Free Grammar and Its Parsing Method Based on the Logic Programming Language by TATSUYA HAYASHI (Fujitsu Laboratories Ltd.).

†† (株)富士通研究所

点は次のとおりである.

(1) スラッシュカテゴリは1個の非終端記号のみから成る.

(2) スラッシュカテゴリはすべてギャップ指定と見なされる.

(3) 規則の右辺第1要素には,スラッシュカテゴリを指定することができない.

(4) 外置された単語に対して,任意個のトレースを対応づけることができない.

これに対して,YAPX は次のような特徴を持っている.

(1) BUP-XG の場合と同様に,文法記述者にとってより自然な方法と考えられるスラッシュカテゴリ記法を用いている.

(2) スラッシュカテゴリには任意の記号列が許される.

(3) ギャップ指定,非ギャップ指定が行える.

(4) 規則の右辺の任意の要素にスラッシュカテゴリを指定できる.

(5) 外置された単語に複数のトレースを対応づけることができる(等位構造の場合).

以下では,2章で拡張された文法記述形式について述べ,3章で実現方法,4章で動作例について述べる.

2. 文法記述形式の拡張

YAPX では,スラッシュ,スコープおよびドメインなる概念を用いて,規則を以下のように拡張する.

2.1 スコープ/スラッシュの導入

まず,YAPX では規則右辺の任意の位置に,

$B/(\alpha)$

を記述することが許される.ここで,Bは非終端記号である.また,αはスラッシュと呼ばれる.スラッシュは,終端記号,非終端記号,ギャップ指示子から成る記号列で,ギャップ指示子としては「〜」を用いることにする.例えば,

rel→relpn s/([the]〜[of]np)

relpn→whose

なる規則を用意すれば,「The man whose name is well known…」のような関係節を自然な形で扱うことができる.この場合のスラッシュ「[the]〜[of]np」は,非ギャップ指定の [the],ギャップ指定の [of] および非ギャップ指定の np をまとめてトレースとすべきことを意味している.

ところで,BUP-XG では XG の open, close に対応して,終端記号 〈, 〉 を導入して $\langle B../C\rangle$ のように表して,トレースの出現範囲を限定するようにしているが,本来これは不要と思われる.すなわち,この場合,Bをトレースの出現範囲と考えるのが自然で,YAPX ではしたがってこのような指定を不要にしている.そして,この時のBをスコープと呼んでいる.なお,YAPX では,下降型解析を行っているので,規則の右辺第1要素にもスラッシュを指定することができる.これにより例えば,

conds→condc/(〜np bep) s

なる規則を用意すれば,「If not specified, A is used」のような省略のある条件文(右外置)を自然な形で扱うことができる.

2.2 ドメインの導入

例えば,「She is the girl that I love but you dislike」なる文のように,関係節が等位接続子 but を持つ複文の場合には,外置された「the girl」に対して2個のトレース (love の直後と dislike の直後) を対応づけなければならない.一般には,対応づけするトレースの数が固定しているわけではないので,このために YAPX では規則右辺の任意の位置に,

$D \,//\, B/(\alpha)$

を記述することを許している.ここで,Bはスコープ,αはスラッシュである.また,Dは非終端記号で,ドメインと呼ばれる.$D \,//\, B/(\alpha)$ の意味は,Dを根とする部分解析木において,スコープBが出現するたびに α をスラッシュとすべきことを表している.これにより例えば,「He is the boy that I love and you love but she dislikes」なる文は,

rel→relpn cs // s/(〜np)

cs→cs conj s

cs→s

なる規則を用いて簡潔に記述しかつ解析することができるのである.

3. 実現方法

前章で述べた文法記述形式の拡張に対処するために,YAP で用いた解析用スタック1対に加えて,さらに1対のスタックを使用することにする(これらを省略スタックと呼ぶ).ドメイン,スコープおよびスラッシュは,解析の過程で動的に省略スタックに格納される.そして,省略スタック上のスラッシュは,一時的にε規則と似たような役割を果たすのである.な

お見やすさのため，以下のホーン節ではカット記号や差分リスト等効率上の配慮はしないことにする．

3.1 省略スタックの導入

YAPX では，ドメインやスコープの制御を行ったりスラッシュを処理するために，省略スタック1対を導入する．省略スタックも解析スタックと同様に一般にマルチスタックで，具体的にはリスト形式で表され，要素スタックの要素は次のようなタップルである．

(id, v), id: d ドメイン
s スコープ
g ギャップ指定
ng 非ギャップ指定
v: 構文記号

id が d, s, g, ng に対応して，タップルをそれぞれドメイン要素，スコープ要素，ギャップ要素，非ギャップ要素と呼ぶ．省略スタックに対する操作に関して，ホーン節のタイプ別アクションは次のようになる．

（1） 左端／中間タイプ：ドメイン，スコープ，スラッシュのすべてに関して，登録・削除の可能性がある．

（2） 右端タイプ：ドメイン，スコープに関して削除の可能性がある．

なお YAP では，解析パスの数（具体的には解析用引数（リスト）の要素の数）が，SAX[8]におけるよりも一般に下回るように最適化されていた．しかし YAPX においては，ドメイン，スコープ，スラッシュを自由に記述できるので，一般にはそのようなことができるとは限らない．そこで，以下では簡単のため，解析パス数の最適化は行わないことにする．また，同様に，解析／省略スタックとも入力用に関してはシングルスタック扱いとする．

3.2 スコープ制御

3.2.1 スコープの登録

スコープの登録には，左端／中間タイプのみが関係する．構文要素 A に対応する述語を $aL(aC)(C, [C|X], Y, T, V)$ とする．

ただし，C：入力解析スタックの先頭（位置番号の集合），
$[C|X]$：解析スタック（入力），
Y：解析スタック（出力），
T：省略スタック（入力），
V：省略スタック（出力）．

ホーン節は次のようになる．

（1） 左端タイプの場合

```
aL([ ], -, [ ], -, [ ]).
aL([n|C1], [C|X], [[[n1, ···, np], C|X]|Y],
    T, [[(s, b)|T]|V]) :-aL(C1, [C|X], Y,
    T, V).
    ···
aL([-|C1], X, Y, T, V) :-aL(C1, X, Y, T, V).
```

（2） 中間タイプの場合

```
aC([ ], -, [ ], -, [ ]).
aC([n|C1], [C|X], [[[n1, ···, np]|X]],
    T, [[(s, b)|T]]).
aC([-|C1], X, Y, T, V) :-aC(C1, X, Y, T, V).
```

ここで，YAP の場合と同様に，n は A の左位置，n_i $(i=1\sim p)$ は右位置または最左導出された位置である．また，(s, b) はスコープ要素で，b は規則の右辺で A の右側に隣接するスコープ B を示すものとする．

左端タイプの場合，A を左端に持つ規則が複数個存在し，かつそれらの A の左位置が共に入力解析スタックの先頭（集合である）に含まれている時には，出力の解析／省略スタックはマルチスタックになる．中間タイプの場合には，このようなことは原則としてない．ただし，一般に，n_i $(i=1\sim p)$ を左位置とするスコープが1個以上存在する場合には，左端タイプとか中間タイプの区別なく，出力の解析／省略スタックをマルチスタックにしなければならないことがあり得る．しかし，これは本質的な点ではないので，ここではその可能性を指摘するに留めて，これ以上立ち入らないことにしよう．

3.2.2 スコープの削除

スコープの削除には，すべてのタイプのホーン節が関係し得る．スラッシュを持つスコープ A に対応するホーン節は次のようになる（A がスコープでない場合は省略する）．

（1） 左端タイプの場合

```
aL([ ], -, [ ], -, [ ]).
aL([n|C1], [C|X], [[[n1, ···, np], C|X]],
    [(s, a)|T], [T]).
aL([-|C1], X, Y, T, V) :-aL(C1, X, Y, T, V).
```

（2） 中間タイプの場合

解析スタックの先頭をリプレースする以外は，左端タイプの場合と同じである．

（3） 右端タイプの場合

```
aR([ ], -, [ ], -, [ ]).
aR([n|C1], [C|X], Y, [(s, a)|T], V) :-b(X, Y,
```

T, V).

$aR([\_|C1], X, Y, T, V)$:-$aR(C1, X, Y, T, V)$.

ここで，b は当該規則の左辺の構文記号に対応する述語である．

3.3 スラッシュ制御

3.3.1 スラッシュの登録

明らかなように，スラッシュの登録は実はスコープの登録と同時に行われる．したがって，スコープの場合と同様に，左端／中間タイプのアクションで指定されるのである．アクションの核心部は次のようになる．なお，中間タイプは左端タイプに準じるので，以降省略することにする．

$aL([n|C1], [C|X], [[[n_1, \cdots, n_p], C|X]|Y], T, [[(g, v), (s, b)|T]|V]$:-$aL(C1, [C|X], Y, T, V)$.

ここで，(g, v) はギャップ指定のスラッシュ要素で，非ギャップ指定の場合は，(ng, v) とすればよい．また，スラッシュが「$\sim v1v2\sim v3$」のような場合には，$(g, v1), (ng, v2), (g, v3)$ をまとめて省略スタックに格納すればよい．

3.3.2 スラッシュの削除

前述したように，スラッシュの削除は左端／中間タイプのみに関係する．そして，スラッシュが非終端記号であるか終端記号であるかの区別なく同一に取り扱われる．ホーン節は次のようになる．

（1） ギャップ指定の場合

$aL([n|C1], [C|X], [[[n_1, \cdots, n_p], C|X]|Y], [(g, v)|T], [[(g, v)|T]|V])$:-$v([[n_1, \cdots, n_p], C|X], Y1, T, V1)$, $aL(C1, [C|X], Y2, [(g, v)|T], V2)$, append $(Y1, Y2, Y)$, append $(V1, V2, V)$.

$aL([n|C1], [C|X], [[[n_1, \cdots, n_p], C|X]|Y], [(g, -)|T], [[(g, -)|T]|V])$:-$aL(C1, [C|X], Y, [(g, -)|T], V)$.

（2） 非ギャップ指定の場合

$aL([n|C1], [C|X], Y, [(ng, v)|T], V)$:-$v([[n_1, \cdots, n_p], C|X], Y1, T, V1)$, $aL(C1, [C|X], Y2, [(ng, v|T], V2)$, append $(Y1, Y2, Y)$, append $(V1, V2, V)$.

$aL([n|C1], [C|X], [\ ], [(ng, -)|T], [\ ])$.

ここで，v はある n_i $(1 \leq i \leq p)$ を左位置に持つスラッシュとする．

3.4 ドメイン制御

ドメインの登録は左端／中間タイプにのみ関係し，削除はすべてのタイプに関係する．規則にドメインの指定がある場合には，実はドメイン自身のほかに，スコープやスラッシュの登録・削除にも影響を与える．

今，$D \,/\!/\, B/(\alpha)$ の場合を考えると，B の左位置が1個でも解析スタックの先頭に含まれる時点で，省略スタックの先頭を参照し，ドメイン要素の有無を確認する．もし存在すれば，スコープ要素およびスラッシュ要素を登録し，存在しなければ登録しない．削除の場合には，スコープ要素，ドメイン要素が省略スタックの先頭および2番目にそれぞれ存在していることを確認して，そのスコープ要素を除去するのである．

上述したように，ドメインの下でスコープが出現するレベルは，常にドメイン要素が省略スタックの先頭に存在する場合に限定している．原理的にはこのような制限を設ける必要は少しもないのであるが，実用上はこれで十分と考えられるからである．

（1） ドメインの登録

$aL([n|C1], [C|X], [[[n_1, \cdots, n_p], C|X]|Y], T, [[(d, d)|T]|V])$:-$aL(C1, [C|X], Y, T, V)$.

（2） ドメインの削除

（a） 左端タイプの場合

$aL([n|C1], [C|X], [[[n_1, \cdots, n_p], C|X], [(d, a)|T], [T])$.

（b） 右端タイプの場合

$aR([n|C1], [C|X], Y, [(d, a)|T], V)$:-$b(X, Y, T, V)$.

（3） ドメイン下のスコープ／スラッシュの登録

$aL([n|C1], [C|X], [[[n_1, \cdots, n_p], C|X], Y], [(d, d)|T], [[(g, v), (s, b), (d, d)|T]|V])$:-$aL(C1, [C|X], Y, [(d, d)|T], V)$.

（4） ドメイン下のスコープの削除

（a） 左端タイプの場合

$aL([n|C1], [C|X], [[[n_1, \cdots, n_p], C|X]], [(s, a), (d, d)|T], [[(d, d)|T]])$.

（b） 右端タイプの場合

$aR([n|C1], [C|X], Y, [(s, a), (d, d)|T], V)$:-$b(X, Y, [(d, d)|T], V)$.

3.5 複合アクション

文法の如何によっては，一つのホーン節の中で上述したアクションを複数個まとめて実行しなければならない場合がある．例えば，

$A \rightarrow D \,/\!/\, B/(C)\ E/(\sim F)\ \alpha$

$E \rightarrow G/(H)\ \beta$

$G \rightarrow H\ \gamma$

のような場合，Dに対応するホーン節では，

（1） ドメイン要素 (d, d) の省略スタックからの削除，

（2） スコープ要素 (s, e) の省略スタックへの登録，

（3） スラッシュ要素 (g, f) の省略スタックへの登録，

（4） スコープ要素 (s, g) の省略スタックへの登録，

（5） スラッシュ要素 (ng, h) の省略スタックへの登録，

（6） スラッシュ要素 (ng, h) の省略スタックからの削除とHに対応するホーン節の呼出し，

をこの順に一括して行わなければならない．

このような際には，動作時ではなくホーン節を作成する時に，関係する個別のアクションを順次結合してかつ省略スタックの冗長操作は除去した上で，一つにまとめればよい．

3.6 入力形式

パーサの動作は上述したように，YAPX の場合も YAP と同様に，モジュール化されたホーン節の集合として記述され，見かけ上構文要素対応で互いに独立している．そこで YAPX では，解析スタック1対，省略スタック1対を引数として用いて，入力文 $w_1 \cdots w_n$ から次のようなホーン節を作成して，これを実際の入力としてモジュール化されたアクション間の結合を行っている．

$$:\text{-open}(X_1, T_1), w_1(X_1, X_2, T_1, T_2), \cdots,$$
$$w_n(X_n, X_{n+1}, T_n, T_{n+1}), \ \text{close}(X_{n+1}, T_{n+1}).$$

ここで，open, close はそれぞれ前処理，後処理（後述）である．また，X_i は解析スタック，T_i は省略スタックである．X_i, T_i はそれぞれ，w_1 から w_{i-1} までの入力部分列を走査した時点での解析パスおよびドメイン／スコープ／スラッシュ要素列を示す．X_i, T_i は述語 w_{i-1} によってセットされ，述語 w_i に渡されるのである．

3.7 前処理／後処理

YAPX では，入力文の第1語の走査に入る前に，次のホーン節で示される前処理が行われる．

$$\text{open}([[[\vdash, \cdots]]], [[\ ]]).$$

第1引数は YAP の場合と同じで，第2引数は省略スタックを空にセットするものである．入力節中の $\text{open}(X_1, T_1)$ は，これによって X_1, T_1 にそれぞれ初期値をセットされる．

解析結果の確認や出力のためには，YAP の場合と同様に，close 述語を適宜定義して用いればよい．

3.8 解析木の生成

YAP と基本的には同じなので，それと異なる外置現象を扱う部分についてのみ示すことにする．

生成スタック1対を用意し，省略スタックからスラッシュ要素 (g, v) または (ng, v) を削除する際に，生成スタックに $v(t)$ を格納すればよい（t はトレースを意味する）．したがって，これは左端／中間タイプでのみ生じる．

（a） ギャップ指定の場合

$aL([n|C1], [C|X], [[[n_1, \cdots, n_p], C|X]|Y], [(g, v)|T], [(g, v)|T]|V], L, [[L]|M]) :-$
$v([[n_1, \cdots, n_p], C|X], Y1, T, V1, [v(t)|L], M1), aL(C1, [C|X], Y2, [(g, v)|T], V2, L, M2),$ append$(Y1, Y2, Y)$, append$(V1, V2, V)$, append$(M1, M2, M)$.

（b） 非ギャップ指定の場合

$aL([n|C1], [C|X], Y, [(ng, v)|T], V, L, M) :-$
$v([[n_1, \cdots, n_p], C|X], Y1, T, V1, [v(t)|L], M1), aL(C1, [C|X], Y2, [(ng, v)|T], V2, L, M2),$ append$(Y1, Y2, Y)$, append$(V1, V2, V)$, append$(M1, M2, M)$.

ここで，L, M はそれぞれ入力用，出力用の生成スタックである．

4. 動作例

ここでは，YAPX の動作を例題にもとづいて見てみよう．入力文としては，「The mouse that the cat

#0. $s_0 \longrightarrow {}^{\vdash} s\ {}^{\dashv}$

1 s ⟶ ⁰ np ¹ vp

2 np ⟶ ² [n]

3 np ⟶ ³ [det] ⁴ [n] ⁵ rel

4 vp ⟶ ⁶ [vt] ⁷ np

5 vp ⟶ ⁸ [vi]

6 rel ⟶ ⁹ [relpn] ¹⁰ cs // s / (~np)

7 cs ⟶ ¹¹ cs ¹² [conj] ¹³ s

8 cs ⟶ ¹⁴ s

図1 拡張文脈自由文法 G1
Fig. 1 An extended CFG G1.

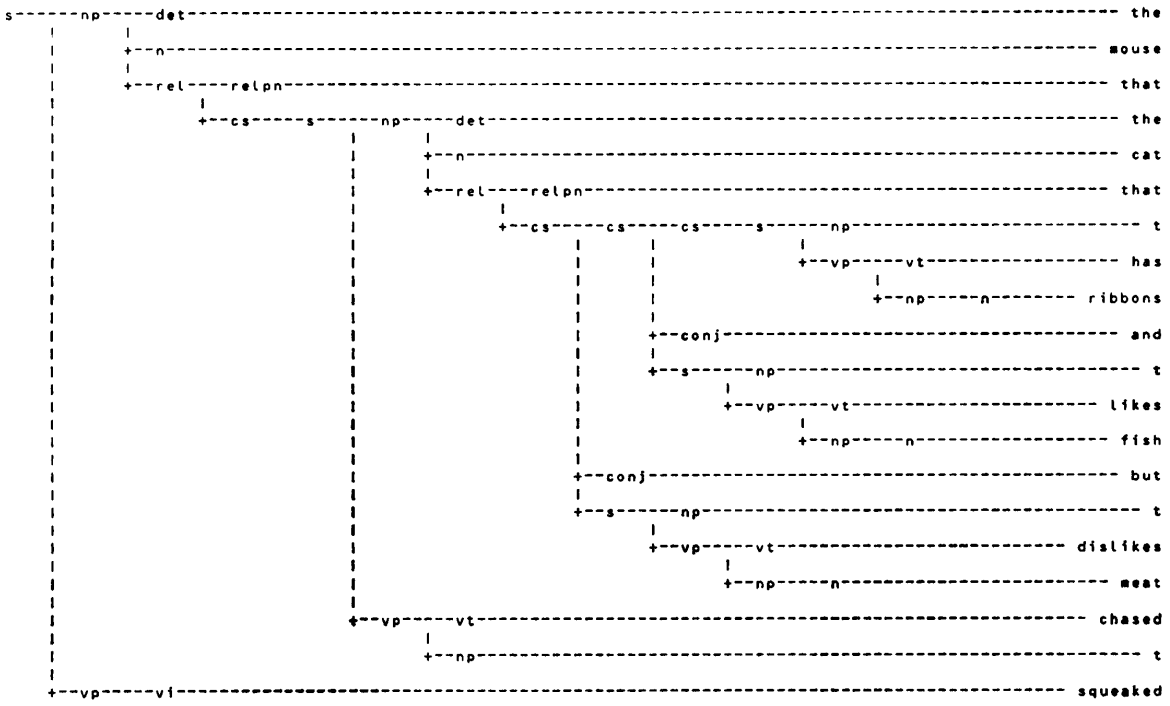

図 2 解 析 木
Fig. 2 A parse tree based on the grammar G 1.

that has ribbons and likes fish but dislikes meat chased squeaked」を用いることにする．また，その際**図 1** の文法を使用することにしよう．ただし，冠詞，名詞等は終端記号扱いとし，[*det*], [*n*] などで表している．この場合の解析木を生成する YAPX プログラムを付録に示す．

入力文の解析結果は**図 2** に示すとおりとなる．図のように，まず「the cat」に対する関係節中の 3 個の主語がトレースで，先行詞「the cat」はこれらのすべてと対応するわけである．動作を順に追ってみると，入力部分列「The mouse that」を走査した時点で，省略スタックの内容は（正しい解析のみに着目すると），

(g, np)	(s, s)	(d, cs)

となっている．これはその時点の解析スタックの先頭が [10, 14, …] なので，まずドメイン要素 (d, cs) が格納され，同時にその環境でスコープ要素 (s, s) がスラッシュ要素 (g, np) と共に格納されるからである．このことはまた，規則 #8 において，右辺の第 1 要素が動的にではあるがスラッシュを持ったことを意味しているわけでもある．さらに「The mouse that the cat that」まで走査した時点では，解析スタックの先頭は再び [10, 14, …] なので，省略スタックは，

(g, np)	(s, s)	(d, cs)	(g, np)	(s, s)	(d, cs)

となる．そして次の「has」の直前に *np* のトレースを認定して (g, np) を削除し，「ribbons」を読んだ時点で (s, s) を削除する．そして次の「and」で（解析スタックの先頭は [13, …]），省略スタックの先頭に (d, cs) が存在することを確認して，再び (s, s) と (g, np) が登録される．これにより，次の「likes fish」が正しく処理され，再び同様な省略スタック操作により，「dislikes meat」が処理され（解析スタック先頭は [10, …]），省略スタックは，

(g, np)	(s, s)	(d, cs)

となる．そして次の「chased」で，その直後に *np* のトレース（先行詞は「The mouse」）を認定し (g, np) を削除する．次いで *s* の解析終了を認定して (s, s) を削除する．さらに続いて，*cs* の解析終了を認定して (d, cs) を削除するのである．そして結局，最後の「squeaked」を走査して，解析は無事終了するわけである．

5. あとがき

筆者は先に，論理型言語による横型下降方式の構文解析法 YAP について述べた．ここでは，特に言語の外置現象を考慮して，文法記述形式を拡張する方法とそれに基づく解析の実現法について考察した．

ここで述べた方式 YAPX により，自然な形で簡潔に自然言語の文法を記述することができると期待している．もちろん YAPX の文法記述形式は，外置現象とは独立に省略が生じる場合（つまり先行詞／後続詞が無い場合）にも適用できよう．

YAPX の実際の有効性については，まとまった自然言語の文法をもとに，別途文法記述力や性能について評価し報告する予定である．

謝辞 本研究の機会を与えていただいた富士通研究所常務山田博氏に感謝する．また，本論文をまとめるに当りお世話になった東工大教授田中穂積氏に深謝する．さらに，本論文の仕上げに種々協力していただいた富士通研究所情報処理研究部門小野越夫，泉田義男，小部正人，杉山健司の諸氏に厚くお礼を申し上げる．

参考文献

1) Hayashi, T.: YAP: Yet Another Efficient Parsing Method for General Context Free Grammars Based on the Logic Programming Language, *Trans. IPS Japan*, (submitted) (Japanese) (1987).
2) Pereira, F.C.N.: Extraposition Grammars, *AJCL*, Vol. 7, No. 4, pp. 243-256 (1981).
3) Konno, S. and Tanaka, H.: Processing Left-extraposition in Bottom up Parsing System, *Computer Software*, Vol. 3, No. 2, pp. 19-29 (Japanese) (1986).
4) Dahl, V. and Abramson, H.: On Gapping Grammars, *Proc. 2nd International Conference on Logic Programming*, pp. 77-88 (1984).
5) Dahl, V.: More on Gapping Grammars, *Proc. International Conference on Fifth Generation Computer Systems*, pp. 669-677 (1984).
6) Popowich, F.: Unristricted Gapping Grammars, *Proc. IJCAI 85*, pp. 765-768 (1985).
7) Gazdar, G., Klein, E., Pullum, G.K. and Sag, I.A.: *Generalized Phrase Structure Grammar*, Basil Blackwell, Oxford (1985).
8) Matsumoto, Y. and Sugimura, R.: SAX: A Parsing System Based on Logic Programming Languages, *Computer Software*, Vol. 3, No. 4, pp. 4-11 (Japanese) (1986).

付録 拡張 CFG G1 に対する YAPX 解析プログラム

(Appendix YAPX implementation of the extended CFG G1)

下記においては，便宜上「⊥」，「⊣」の代わりに「*b*」，「*e*」を用いている．また，入力節は $det_1(X1, X2, T1, T2, L1, L2, \text{the})$, $n_1(X2, X3, T2, T3, L2, L3, \text{mouse})$ などから構成している．なお，close の定義は省略する．

```
det([C|X],Y,T,V,L,M) :- detL(C,[C|X],Y,T,V,L,M).

detL([],_,[],_,[],_,[]).
detL([3|C1],X,[[[4]|X]],T,[T],L,[L]).
detL([_|C1],X,Y,T,V,L,M) :- detL(C1,X,Y,T,V,L,M).

n([C|X],Y,T,V,L,M) :-
        nC(C,[C|X],Y1,T,V1,L,M1),
        nR(C,[C|X],Y2,T,V2,L,M2),
        append(Y1,Y2,Y),append(V1,V2,V),append(M1,M2,M).

nC([],_,[],_,[],_,[]).
nC([4|C1],[C|X],[[[5,9]|X]],T,[T],L,[L]).
nC([_|C1],X,Y,T,V,L,M) :- nC(C1,X,Y,T,V,L,M).

nR([],_,[],_,[],_,[]).
nR([2|C1],X,Y,T,V,[N|L],M) :- np(X,Y,T,V,[np(N)|L],M).
nR([_|C1],X,Y,T,V,L,M) :- nR(C1,X,Y,T,V,L,M).

vi([C|X],Y,T,V,L,M) :- viR(C,[C|X],Y,T,V,L,M).

viR([],_,[],_,[],_,[]).
viR([8|C1],X,Y,T,V,[VI|L],M) :- vp(X,Y,T,V,[vp(VI)|L],M).
viR([_|C1],X,Y,T,V,L,M) :- viR(C1,X,Y,T,V,L,M).

np([C|X],Y,T,V,L,M) :-
        npL(C,[C|X],Y1,T,V1,L,M1),
        npR(C,[C|X],Y2,T,V2,L,M2),
        append(Y1,Y2,Y),append(V1,V2,V),append(M1,M2,M).

npL([],_,[],_,[],_,[]).
npL([0|C1],X,[[[1,6,8]|X]],T,[T],L,[L]).
npL([_|C1],X,Y,T,V,L,M) :- npL(C1,X,Y,T,V,L,M).

npR([],_,[],_,[],_,[]).
npR([7|C1],[C|X],Y,T,V,[NP,VT|L],M) :-
        vp(X,Y,T,V,[vp(VT,NP)|L],M).
npR([_|C1],X,Y,T,V,L,M) :- npR(C1,X,Y,T,V,L,M).

vp([C|X],Y,T,V,L,M) :- vpR(C,[C|X],Y,T,V,L,M).

vpR([],_,[],_,[],_,[]).
vpR([1|C1],[C|X],Y,T,V,[VP,NP|L],M) :- s(X,Y,T,V,[s(NP,VP)|L],M).
vpR([_|C1],X,Y,T,V,L,M) :- vpR(C1,X,Y,T,V,L,M).

rel([C|X],Y,T,V,L,M) :- relR(C,[C|X],Y,T,V,L,M).

relR([],_,[],_,[],_,[]).
relR([5|C1],[C|X],Y,T,V,[REL,N,DET|L],M) :-
        np(X,Y,T,V,[np(DET,N,REL)|L],M).
relR([_|C1],X,Y,T,V,L,M) :- relR(C1,X,Y,T,V,L,M).

s([C|X],Y,T,V,L,M) :-
        sL(C,[C|X],Y1,T,V1,L,M1),
        sR(C,[C|X],Y2,T,V2,L,M2),
        append(Y1,Y2,Y),append(V1,V2,V),append(M1,M2,M).

sL([],_,[],_,[],_,[]).
sL([b|C1],X,[[[e]|X]],T,[T],L,[L]).
sL([_|C1],X,Y,T,V,L,M) :- sL(C1,X,Y,T,V,L,M).

sR([],_,[],_,[],_,[]).
sR([13|C1],[C|X],Y,[[s|s],[d|cs]|T],V,[S,CONJ,CS|L],M) :-
        cs(X,Y,[[d|cs]|T],V,[cs(CS,CONJ,S)|L],M).
sR([14|C1],X,Y,[[s|s],[d|cs]|T],V,[S|L],M) :-
        cs(X,Y,[[d|cs]|T],V,[cs(S)|L],M).
sR([_|C1],X,Y,T,V,L,M) :- sR(C1,X,Y,T,V,L,M).

vt([C|X],Y,T,V,L,M) :- vtL(C,[C|X],Y,T,V,L,M).

vtL([],_,[],_,[],_,[]).
vtL([6|C1],X,[[[7,2,3]|X]|Y],[[g|np]|T],[[[g|np]|T]|V],L,[L|M]) :-
        np([[7,2,3]|X],Y,T,V,[np(t)|L],M).
vtL([6|C1],X,[[[7,2,3]|X]],T,[T],L,[L]).
vtL([_|C1],X,Y,T,V,L,M) :- vtL(C1,X,Y,T,V,L,M).

relpn([C|X],Y,T,V,L,M) :- relpnL(C,[C|X],Y,T,V,L,M).

relpnL([],_,[],_,[],_,[]).
relpnL([9|C1],X,[[[10,11,14,0,2,3]|X]|Y],T,
                [[[g|np],[s|s],[d|cs]|T]|V],L,[L|M]) :-
        np([[10,11,14,0,2,3]|X],Y,[[s|s],[d|cs]|T],V,[np(t)|L],M).
```

```
relpnL([_|C1],X,Y,T,V,L,M) :- relpnL(C1,X,Y,T,V,L,M).

conj([C|X],Y,T,V,L,M) :- conjC(C,[C|X],Y,T,V,L,M).

conjC([],_,[],_,[],_,[]).
conjC([12|C1],[C|X],[[[13,0,2,3]|X]|Y],[[d|cs]|T],
                [[[s|np],[s|s],[d|cs]|T]|V],L,[L|M]) :-
      np([[13,0,2,3]|X],Y,[[s|s],[d|cs]|T],V,[np(t)|L],M).
conjC([_|C1],X,Y,T,V,L,M) :- conjC(C1,X,Y,T,V,L,M).

cs([C|X],Y,T,V,L,M) :-
      csL(C,[C|X],Y1,T,V1,L,M1),
      csR(C,[C|X],Y2,T,V2,L,M2),
      append(Y1,Y2,Y),append(V1,V2,V),append(M1,M2,M).

csL([],_,[],_,[],_,[]).
csL([11|C1],X,[[[12]|X]],T,[T],L,[L]).
csL([_|C1],X,Y,T,V,L,M) :- csL(C1,X,Y,T,V,L,M).

csR([],_,[],_,[],_,[]).
csR([10|C1],[C|X],Y,[[d|cs]|T],V,[CS,RELPN|L],M) :-
      rel(X,V,T,V,[rel(RELPN,CS)|L],M).
csR([_|C1],X,Y,T,V,L,M) :- csR(C1,X,Y,T,V,L,M).

det1([],[],[],[],[],[],[],_).
det1([X|XX],Y,[T|TT],V,[L|LL],M,Q) :-
      det(X,Y1,T,V1,[det(Q)|L],M1),
      det1(XX,Y2,TT,V2,LL,M2,Q),
      append(Y1,Y2,Y),append(V1,V2,V),append(M1,M2,M).

n1([],[],[],[],[],[],[],_).
n1([X|XX],Y,[T|TT],V,[L|LL],M,Q) :-
      n(X,Y1,T,V1,[n(Q)|L],M1),
      n1(XX,Y2,TT,V2,LL,M2,Q),
      append(Y1,Y2,Y),append(V1,V2,V),append(M1,M2,M).

vt1([],[],[],[],[],[],[],_).
vt1([X|XX],Y,[T|TT],V,[L|LL],M,Q) :-
      vt(X,Y1,T,V1,[vt(Q)|L],M1),
      vt1(XX,Y2,TT,V2,LL,M2,Q),
      append(Y1,Y2,Y),append(V1,V2,V),append(M1,M2,M).

vi1([],[],[],[],[],[],[],_).
vi1([X|XX],Y,[T|TT],V,[L|LL],M,Q) :-
      vi(X,Y1,T,V1,[vi(Q)|L],M1),
      vi1(XX,Y2,TT,V2,LL,M2,Q),
      append(Y1,Y2,Y),append(V1,V2,V),append(M1,M2,M).

relpn1([],[],[],[],[],[],[],_).
relpn1([X|XX],Y,[T|TT],V,[L|LL],M,Q) :-
      relpn(X,Y1,T,V1,[relpn(Q)|L],M1),
      relpn1(XX,Y2,TT,V2,LL,M2,Q),
      append(Y1,Y2,Y),append(V1,V2,V),append(M1,M2,M).

conj1([],[],[],[],[],[],[],_).
conj1([X|XX],Y,[T|TT],V,[L|LL],M,Q) :-
      conj(X,Y1,T,V1,[conj(Q)|L],M1),
      conj1(XX,Y2,TT,V2,LL,M2,Q),
      append(Y1,Y2,Y),append(V1,V2,V),append(M1,M2,M).

open([[[b,0,2,3]]],[[]],[[]]).

append([],X,X).
append([A|B],X,[A|Y]) :- append(B,X,Y).

start :-
      open(X1,T1,L1),
      det1(X1,X2,T1,T2,L1,L2,the),
      n1(X2,X3,T2,T3,L2,L3,mouse),
      relpn1(X3,X4,T3,T4,L3,L4,that),
      det1(X4,X5,T4,T5,L4,L5,the),
      n1(X5,X6,T5,T6,L5,L6,cat),
      relpn1(X6,X7,T6,T7,L6,L7,that),
      vt1(X7,X8,T7,T8,L7,L8,has),
      n1(X8,X9,T8,T9,L8,L9,ribbons),
      conj1(X9,X10,T9,T10,L9,L10,and),
      vt1(X10,X11,T10,T11,L10,L11,likes),
      n1(X11,X12,T11,T12,L11,L12,fish),
      conj1(X12,X13,T12,T13,L12,L13,but),
      vt1(X13,X14,T13,T14,L13,L14,dislikes),
      n1(X14,X15,T14,T15,L14,L15,meat),
      vt1(X15,X16,T15,T16,L15,L16,chased),
      vi1(X16,X17,T16,T17,L16,L17,squeaked),
      close(X17,T17,L17).
```

(昭和62年7月16日受付)
(昭和63年3月 9 日採録)

林 達也（正会員）

1937年生．1960年早稲田大学第一理工学部応用物理学科卒業．同年富士通(株)入社．以来，ソフトウェア工学(コンパイラ自動作成，設計支援，部品化)，データベース(リレーショナル，分散，マルチメディア)，自然言語処理(機械翻訳，自然言語インタフェイス)，人工知能（エキスパートシェル，知識表現）等の研究開発に従事．富士通研究所情報処理研究部長代，ソフトウェア研究部長を経て現在情報処理研究部門所属．元本学会編集幹事．1973年度本学会論文賞受賞．ACM，日本ソフトウェア科学会，日本モーツァルト協会各会員．AAI (Applied Artificial Intelligence) (Hemisphere) Editorial Board メンバ．