句構造文法に対する効率的文解析手法†

林 達也††

本論文では，一般の句構造文法（チョムスキーの0型，1型文法をベースとする論理文法）に対する効率的な文解析手法について述べる．この方法は，限定文脈依存文法 (RCSG) を受け入れる YAPXR 文解析システムを拡張することによって得られる．したがって本方式は，YAPXR と同様に横型トップダウン手法に基づいており，かつまた，YAPXR の特性を継承して効率の良い文解析を行うことができる．日本語のように語順が自由で省略の多い言語の場合，文節レベル以上の文構造に対して，文脈自由文法ベースの記述形式は必ずしもしっくりしない面がある．すなわち，文節間の係り受けの特徴を統語的事象として捉える立場をとると，むしろ上位レベルの文法の方が文脈自由文法ベースよりも素直に日本語の規則を記述することができる．一方ギャップ文法も，語順の自由な言語を扱えるが，性能に問題がある．本方式はこれに対して，文法が与えられた時点であらかじめすべての位置集合が求められれば，YAPXR と同様に核位置方式により，そうでない場合も擬似核位置方式によって，効率的な文解析を行うことができるので，この種の言語に有効な手法であると考えられる．

1. まえがき

筆者は先に，文脈自由文法を主体にしそれに解析効率に影響を及ぼさない範囲で文脈依存規則を導入した論理文法 RCSG (Restricted Context Sensitive Grammar) に対する文解析システム YAPXR について述べた[1)~5)]．

RCSG は基本的には，文脈自由文法の枠組みからそれほどかけ離れていない．

そこで，日本語のように語順が自由で省略の多い言語の場合は，文節レベル以上の文構造に対して，必ずしも RCSG が適切な文法であるとは言えない．

ギャップ文法は語順の自由な言語を簡潔に記述できる特色を持っているが[6)~8)]，gap が0を含む任意の長さの部分導出列とマッチングし，かつ，generate & test 方式で実現されるので，性能上問題が生じることが予想される．

一方日本語の場合，語順が自由だといっても無条件に自由なわけではない．文節を単位として係り受け関係が右向きでかつ非交叉でなければならないという条件が存在する．

そして，この係り受け関係に対応する非終端記号を導入すると，日本語は一般的な句構造文法（チョムスキーの0型，1型をベースとする論理文法をこう呼ぶことにする）を用いて素直に記述できることが示されている[9)]．本論文も上述した係り受けの特徴を統語的事象として捉える立場をとるが，その際，句構造文法に対する効率的な解析手法が課題となる．

これに対しては YAPXR システムを多少拡張することで，横型トップダウン方式の句構造文法向けの実用的な解析システムを得ることができる．

本論文ではこれについて述べることにする．以下では，2章で主な用語を定義し，3章で文法記述形式について説明する．次いで，4章で実現方法，5章で動作例について述べることにする．

2. 用語の定義

本章では，形式言語理論で用いられる一般的用語・記法をベースとして，本稿で新たに用いる用語について定義する．

まず，文法規則は

$\alpha \rightarrow \beta$, $(\alpha \in V^+, \beta \in V^*)$

として，右辺の要素の直前にそれぞれ異なる自然数を付与しておくものとする．これを位置 id と呼ぶ．そして，ある要素の直前・直後の位置 id をそれぞれその要素の左位置・右位置と呼ぶ．ε 規則にも位置 id を付与するものとする．また，第0規則のみは特別に「b」，「e」を位置 id として用いて，$S_0 \rightarrow {}^{b}S^{e}\dashv$ とする．

2.1 最左導出位置

ある規則の右辺の要素を $A(\in V)$ とし，その左位置を n とする．その時，

$S_0 \overset{*}{\Rightarrow} \alpha A \delta \overset{*}{\Rightarrow} \alpha\beta\gamma \Rightarrow \alpha\eta\gamma$, $(\alpha, \gamma, \delta, \eta \in V^*, \beta \in V^+)$

で，η の先頭位置が m ならば，m を n の最左導出位置と呼び，$m \in \mathrm{LD}(n)$ で表す．

† An Efficient Sentence Analysis Method for General Phrase Structure Grammars by TATSUYA HAYASHI (Fujitsu Laboratories Ltd.).

†† (株)富士通研究所

ただし，$\alpha A\delta \Rightarrow \alpha\beta\gamma$ においてε規則の適用は禁止するものとする．

2.2 位置集合

$A(\in V)$ をある規則の右辺の左端でない要素とし，左位置を n とする．この時，集合 $\{n\} \cup \mathrm{LD}(n)$ を A の位置集合と呼ぶ．ただし例外として，第0規則の左端要素 S に対してのみ位置集合を認める．

2.3 核位置

位置集合においてある元を n とし他の任意の元を x とした時，$x \in \mathrm{LD}(n)$ ならば n をその集合の核位置と呼ぶ．また，n を x に対応する核位置と呼ぶ．

この定義により，x は規則右辺の左端要素の左位置であり，前述した「b」を例外として，核位置にはなれないことが容易にわかる．なお，核位置は位置集合の id とみなすこともできる．

2.4 解析パス

入力文を $a_1a_2\cdots a_n$ とする．入力部分列を $a_1\cdots a_i (1 \leqq i \leqq n)$ とし，それを含む最左導出列の一つを

$$S_0 \Rightarrow a_1\cdots a_i\alpha,\ (\alpha \in V^*)$$

とする．

解析パスは b を先頭とする核位置の列 $bn_1n_2\cdots n_i (1 \leqq i)$ から成り，第0規則から始まりどの規則をどの順に適用して入力部分列 $a_1\cdots a_i$ が得られたか，また，各規則のどの位置まで対応する（と思われる）部分列が検出されたかを示している．すなわち，解析パスは言わば最左導出の縮小表現であり，かつ，異なる最左導出に対しては，得られる解析パスも異なる．そして逆に，解析パスから対応する最左導出を求めることができる．厳密には，複数の最左導出が一時的に同一の解析パスに対応する場合があるが，それらをその時点で区別する必要はない．

2.5 （内部）引数

規則の左辺および右辺の要素 A に対して，A (IX) のようにかっこで括って引数を付与することができるが，これを（内部）引数と呼ぶ．

引数は，A に対応する部分入力列を何らかの意味で特徴づける属性として用いられる．

引数の値は単語辞書から直接間接に得られる場合と，規則中で直接値を設定される場合の2通りある．後者の場合は引数名の後にかっこを用いて，値を指定する．

2.6 （基本）制約

基本制約は，内部引数に関する論理式または prolog 述語呼出しである．

制約は規則右辺の任意の場所に記述することができる．

3. 文法記述形式

ここで許される文法は，従来の YAPXR で用いられている機能をすべて包含した句構造文法である．

基本形式は次に示すとおりである．

左辺→右辺，

ここで，左辺：左辺項…

左辺項：構文要素[(内部引数，…)]
[(/外部引数，…/)]

右辺：ε または右辺項…

右辺項：構文要素[(内部引数，…)]
[(/外部引数，…/)]
[スラッシュカテゴリ]
[{文脈依存制約}]
[{基本制約}]

ただし，外部引数や文脈依存制約，スラッシュカテゴリは本論と直接関係しないので以下では省略する（興味のある読者は文献 2), 4), 5) を参照されたい）．

4. 実現方法

記述形式が RCSG に合致する場合は，従来の実現方法と全く同じなので，ここでは，一般の句構造文法規則を対象にして処理方法を述べることにする．

ここで問題は，RCSG の場合は文法が与えられた時点で，規則右辺の左端要素に対応する核位置をすべて容易に求めることができた．しかし句構造文法の場合は，それが常に可能であるとは限らない．そこで以下ではまず，手動，半自動，自動の如何を問わず何らかの方法で，核位置がすべて求められる場合に関して，話を進めよう．まず，文法規則を

$$P_1P_2\cdots P_m \rightarrow {}^{n1}Q_1\ {}^{n2}Q_2\cdots {}^{nn}Q_n,$$
$$(P_i, Q_j \in V,\ \mathrm{nk}\sim\text{位置 id})$$

とする．簡単のため引数や制約は除外して考えることにする．また，P_i, Q_j に対応する述語をそれぞれ p_i, q_j とする．

述語の引数は YAPXR では以下のとおりであった．

n：解析スタックの先頭要素，
A_i：解析スタックの残り（入力），
A_o, A_{o1}：解析スタック（出力，差分リスト），
O_i：省略スタック（入力），
O_o, O_{o1}：省略スタック（出力，差分リスト），
L_i：引数スタック（入力），

L_o, L_{o1}: 引数スタック（出力，差分リスト），
T_i: 解析木スタック（入力），
T_o, T_{o1}: 解析木スタック（出力，差分リスト），
しかしここでは，本論に関係ない省略スタックは除外して考える．また，解析木は句構造文法の場合複雑になるので，解析木スタックには替わりに係り受け関係(W1, W2)を格納することにする．

(W1, W2)は文節W1が文節W2に係ることを示す．

4.1 左端型

構文要素 Q_1 に対応するホーン節は，

$q_1(l1, A_i, [[n2, l1|A_i]|A_o], A_o, L_i, [[L_i]|L_o], L_o, T_i, [[T_i]|T_o], T_o)$.
$q_1(l2, A_i, [[n2, l2|A_i]|A_o], A_o, L_i, [[L_i]|L_o], L_o, T_i, [[T_i]|T_o], T_o)$.
…
…
$q_1(lk, A_i, [[n2, lk|A_i]|A_o], A_o, L_i, [[L_i]|L_o], L_o, T_i, [[T_i]|T_o], T_o)$.

となる．$l1$〜lk は $n1$ に対応する核位置である．lj を左位置として持つ要素は一般に，非終端記号に限らないのが句構造文法の場合の特色である．

前述したように一般には，$n1$ に対応する核位置を求めることが容易でない場合がある．ただしそれだからといって，解析ができないわけではないということに注意を要する．

その際は以下に述べる二つの方法のいずれかを用いることによって，解析を進めることができる．

第1の方法は，解析パスの先頭要素が何であってもあたかも核位置であるかのようにみなして，解析パスを $n2$ に進めるものである．

これは時間，空間的に非能率ではあるが，解析を正しく行うことができる．ただ，その時の解析は完全なボトムアップになるので，ε 規則は禁止されることになる．これに対して第2の方法は，ε 規則を許しかつ現実的な方法であると言える．

まず最初に，句構造規則の左辺は第1項のみを考える．すると，左端要素に対応する核位置は限定文脈自由文法の場合と同様に，すべて求めることができる．

これを擬似核位置と呼ぶことにしよう．容易にわかるように，本来の核位置は同時にまた擬似核位置でもある．

こうしておいて，左端型のアクションを擬似核位置に基づいて作成すれば良い．

4.2 中間型

Q_2 に対応するホーン節は，

$q_2(n2, A_i, [[n3|A_i]|A_o], A_o, L_i, [[L_i]|L_o], L_o, T_i, [[T_i]|T_o], T_o)$.

で，これは従来と全く同じである．

4.3 右端型

Q_n に対応するホーン節を次のように定めることがポイントである．

$q_n(nn, [n|A_i], A_o, A_{o1}, L_i, L_o, L_{o1}, T_i, T_o, T_{o1})$:-
$p_1(n, A_i, A1, A_{o1}, L_i, L1, L_{o1}, T_i, T1, T_{o1})$,
$p_2 00(A1, A2, L1, L2, T1, T2)$,
…

```
0.  s0 ---> ' s ' .

1.  s  ---> ' prseq ' [p ] ( w,c1,c2) {c2=end} ' [e ]

2.  prseq---> ' [p ] (w,c1,c2) 'rel

3.  prseq---> ' [p ] (w,c1,c2) 'rel 'prseq

4.  [p ] (w1,c11,c12) rel [p ] (w2,c21,c22)---> ' [p ] (w1,c11,c12) '0prseq
        '1 [p ] (w2,c21,c22) {check(c11,c12,c21),put(w1,w2) }

5.  [p ] (w1,c11,c12) rel [p ] (w2,c21,c22)---> '2 [p ] (w1,c11,c22) '3rel
        '4prseq '5 [p ] (w2,c21,c22) {check(c11,c12,c21),put(w1,w2) }

6.  [p ] (w1,c11,c12) rel [p ] (w2,c21,c22)---> '6 [p ] (w1,c11,c12) '7 [p ]
        (w2,c21,c22) {check(c11,c12,c21),put(w1,w2) }


    check(n,ga,vt).
    check(n,ni,vt).
    check(n,wo,vt).
    check(vt,cn,n).
```

図1 句構造文法例
Fig. 1 An example of phrase structure grammar.

…

$p_m00(A_{m-1}, A_0, L_{m-1}, L_0, T_{m-1}, T_0)$.

つまり，左辺の要素全体に対応する（と思われる）部分入力列が検出されたとみなして，記述された順序で連続して述語呼出を行う点が句構造文法の場合の特色である．

これを S_0 からの導出と関連づけて見ると，右端型のアクションは，

$S_0 \overset{*}{\Rightarrow} \alpha P_1 P_2 \cdots P_m \gamma \Rightarrow \alpha Q_1 Q_2 \cdots Q_n \gamma$

において，α の最後の要素の右位置から γ の最初の要素の左位置に解析パスを進めることにほかならない．$p_200 \sim p_m00$ は入力文に対応する述語と同様に，解析スタック（入力）は一般に複数の解析パスから成る（ただし，リストの最後が変数である点が異なる）．したがって，以下のホーン節を設けて，インタフェースを合わせるものとする．

```
p_j00([[N|A_h]|A_t], A_j,
  [L_h|L_t], L_j,
  [T_h|T_t], T_j):-
p_j(N, A_h, A_j, A_j1,
  L_h, L_j, L_j1,
  T_h, T_j, T_j1),
p_j00(A_t, A_j1,
  L_t, L_j1,
  T_t, T_j1).
p_j00(A_01, A_01,
  L_01, L_01,
  T_01, T_01,):-!.
```

擬似核位置方式の場合も，右端型のアクションは同様に，左辺の要素に対応する述語を逐次的に呼び出せばよい．

ただしこの場合の相違点は，核位置方式では最終的な解析パスが常に存在するが，擬似核位置方式では逐次的な述語呼出しの過程で，解析パスが消滅してしまう場合があるということである．

つまり，擬似核位置方式は核位置方式より効率が良くないが，ボトムアップ方式に比べると効率的で，両者の中間に位置する方法であるといえる．

```
(1)
(tarouga,yakusokusuru) (jirouni,yakusokusuru) (jirouni,renrakusuru)
(renrakusuru,kotowo)   ( kotowo,yakusokusuru)
(2)
(tarouga,yakusokusuru) (jirouni,yakusokusuru)
(renrakusuru,kotowo)   ( kotowo,yakusokusuru)
(3)
(tarouga,yakusokusuru)                        (jirouni,renrakusuru)
(renrakusuru,kotowo)   ( kotowo,yakusokusuru)
(4)
                       ( tarouga,renrakusuru) (jirouni,renrakusuru)
(renrakusuru,kotowo)   ( kotowo,yakusokusuru)
(5)
(tarouga,yakusokusuru) (tarouga,renrakusuru)  (jirouni,renrakusuru)
(renrakusuru,kotowo)   ( kotowo,yakusokusuru)
```

図 2 例文の解析結果
Fig. 2 Analysis result of input sentence.

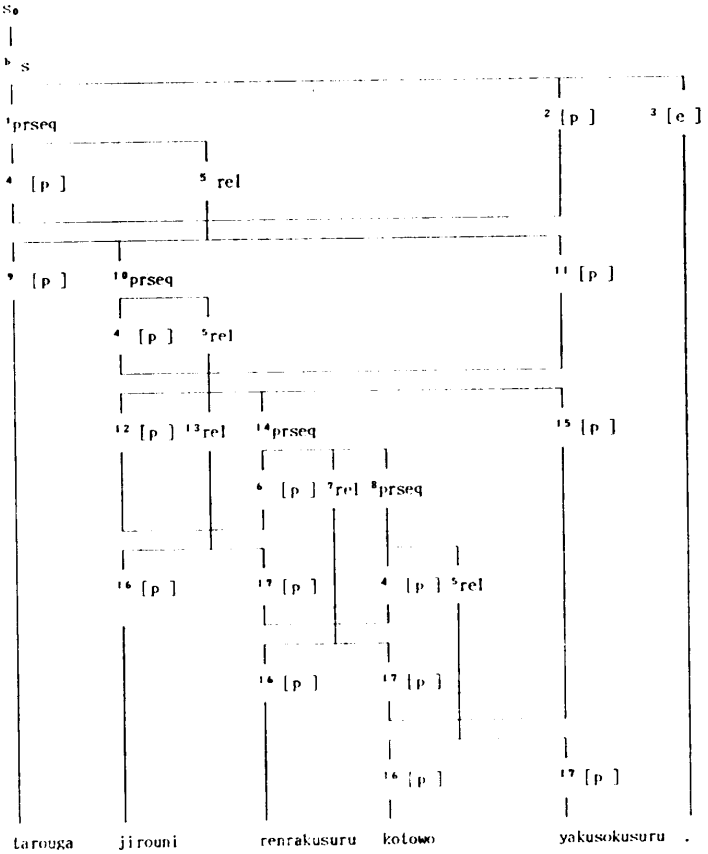

図 3 例文の解析チャート
Fig. 3 A parse chart of input sentence.

5. 動 作 例

本章では簡単な日本語の文法を例にとり上げて，パーサの動きを説明しよう．

使用する文法例を**図1**に示す．これは文献 9) を参考にして少し手を加えたものである．

[e] はピリオドを表す．[p] は文節を示す終端記号で以下の引数を持つものとする．

W：自立語＋付属語，

C1：自立語の品詞区分，

C2：付属語の品詞区分または活用形

rel ら係り受け関係を示す非終端記号である．ただし，rel のみを左辺に持つ規則はなく，したがってそれから導出される終端記号は存在しない点を指摘しておく．

簡単のため，s や prseq には引数を持たせないことにする．規則 1, 4, 5, 6 の check は構文的側面から係り受けの可否を調べる簡単な述語である．また，put は係り受け関係 (W1, W2) を解析木スタックに格納する命令である．規則1の基本制約は最後の文節が終止形でなければならないことを指定している．

この例では，核位置をすべて求めることができる．左端要素 ^{9}[p], ^{12}[p], ^{16}[p] に対応する核位置には，b, 8, 10, 14, 11, 15, 17 の7個が存在する．この文法に基づく解析プログラム (Quintus prolog 版) を付録に示す．

入力文を「tarouga jirouni renrakusuru kotowo yakusokusuru.」としよう．また，本システムへの入力述語は，

```
yopen (A1, L1, T1),
  p0(A1, A2, L1, L2, T1, T2,
                    [tarouga, n, ga]),
  p0(A2, A3, L2, L3, T2, T3,
                    [jirouni, n, ni]),
  p0(A3, A4, L3, L4, T3, T4,
                    [renrakusuru, vt, cn]),
  p0(A4, A5, L4, L5, T4, T5,
                    [kotowo, n, wo]),
  p0(A5, A6, L5, L6, T5, T6,
                    [yakusokusuru, vt, end]),
  e0(A6, A7, L6, L7, T6, T7,
                    ['.', end, end]),
yclose (A7, L7, T7).
```

としよう．ここで，

n：名詞，

vt：他動詞，

cn：連体形，

end：終止形，

yopen：スタックの初期設定を行う述語，

yclose：結果（係り受け構造）を出力する述語，

である．入力述語のホーン節も付録に示しておく．解析チャートの数はこの場合 22 個であるが，係り受け構造を見ると異なり数は**図2**に示すように5通りとなる．

(1)に対応する解析チャートの一つを示すと**図3**のようになる．余談であるが，この例は文脈処理まで行わなければ曖昧性が解消しない．

本方式は LR 手法をベースにしているので，従来の YAPXR と同様に，解析チャートはボトムアップに作成される．

図3の場合，まず規則6により (kotowo, yakusokusuru) に対応する部分チャートが生成され，述語 p, rel, p が呼び出される．先頭の p はやはり規則6により右端型アクションを行って (renrakusuru, kotowo)

```
0. s0 → ᵇ s ᵉ ┤

1. s → ¹ prseq ² [p] (w,c1,c2) {c2=end} ³ [e]

2. prseq → ⁴ pr(w,c1,c2)

3. prseq → ⁵ pr(w,c1,c2) ⁶prseq

4. pr(w1,c11,c12) [p] (w2,c21,c22) → ⁷ [p] (w1,c11,c12) ⁸prseq
        ⁹ [p] (w2,c21,c22) {check(c11,c12,c21),put(w1,w2)}

5. pr(w1,c11,c12) [p] (w2,c21,c22) → ¹⁰ pr(w1,c11,c22) ¹¹prseq
                ¹² [p] (w2,c21,c22) {check(c11,c12,c21),put(w1,w2)}

6. pr(w1,c11,c12) [p] (w2,c21,c22) → ¹³ [p] (w1,c11,c12) ¹⁴ [p]
        (w2,c21,c22) {check(c11,c12,c21),put(w1,w2)}

check(n,ga,vt).
check(n,ni,vt).
check(n,wo,vt).
check(vt,cn,n).
```

図4 句構造文法例（改良）
Fig. 4 Revised version of the phrase structure grammar.

表 1 動作結果
Table 1 Performance result.

	核位置方式	擬似核位置方式
解析時間	700 ms	800 ms
作業メモリ量	224,036 bytes	261,452 bytes
ピリオドに伝わる解析パス	1,225 本	1,460 本

SUN 3/60, Quintus Prolog R 2.4

表 2 動作結果（改良）
Table 2 Revised performance result.

	核位置（擬似核位置）方式
解析時間	133 ms
作業メモリ量	224,036 bytes
ピリオドに伝わる解析パス	205 本

SUN 3/60, Quintus Prolog R 2.4

に対応する部分チャートを生成し，再び p, rel, p を呼び出す．以下同様の手順を経て図のような解析チャートが生成されるわけである．

ここで，終端要素 ^{17}[p] が左端要素 ^{16}[p] の核位置になっている所が従来の RCSG とは異なる点である．例文に対する解析時間や作業メモリ量は**表 1** のようになる．

核位置方式と擬似核位置方式の相違はこの場合，単に左端要素 ^{9}[p], ^{12}[p], ^{16}[p] に対して，^{2}[p] が擬似核位置として新たに追加されるだけである．

次に，例文に対する解析チャートの数が冗長なので，係り受け構造の異なり数と一致するように文法を改良すると**図 4** のようになる．これに基づく動作結果を**表 2** に示す．

この場合には，核位置方式と擬似核位置方式との相違がなくなり，性能は 5～6 倍向上している．

6. あとがき

本論文では，横型トップダウン文解析システム YAPXR を拡張して一般の句構造文法（チョムスキーの 0 型，1 型をベースとする論理文法）に対する効率的な文解析システムが得られることを示した．

日本語のように語順が自由で省略の多い言語の場合，文節レベル以上の文構造に対して，文脈自由文法の枠組みで記述することは必ずしも適切とは言い難い面がある．ギャップ文法では，記述力は一応あるとしても，能率の良い解析が期待できない．

日本語は，係り受けの特徴を統語的事象として捉える立場をとると，句構造文法の方が素直に表現できるので，本方式の有効性が期待できる．

ここに述べた方法は，従来の YAPXR と同様に，

（1） 左端要素に対応する核位置あるいは擬似核位置をすべて，文法が与えられた時点で求めることができるので，実行時のトップダウン予測が不要，

（2） prolog 処理系の特性を生かして，第 1 引数をアトムにしてダブルハッシュを可能にしている，

（3） 実行時にはバックトラックを生じない，

等により効率の良い解析システムになっているからである．

今後は，句構造文法の広い範囲に対して，核位置を求める手法の開発が課題である．また，本文では基本手法とその有効性の片鱗を簡単な例で示したに過ぎないので，機械翻訳や自然言語インタフェース等の実用規模の日本語への適用が今後の課題と考えている．

謝辞 日頃いろいろと助言していただく田中穂積氏（東工大教授）に感謝する．また，システムの開発に従事していただいた宮俊司，坂巻利哉，吉田健一（富士通 SSL）の諸氏に厚く御礼を申し上げる．

参考文献

1) 林 達也：論理型言語による構文解析法 YAP について，情報処理学会論文誌，Vol. 29, No. 9, pp. 835-842 (1988).
2) 林 達也：拡張 CFG とその構文解析法 YAPX について，情報処理学会論文誌，Vol. 29, No. 5, pp. 480-487 (1988).
3) 林 達也：YAPX の効率的実現法，情報処理学会論文誌，Vol. 30, No. 10, pp. 1354-1356 (1989).
4) 林 達也：横型トップダウン文解析システムの実現と評価，情報処理学会自然言語処理研究会，74-9, pp. 65-72 (1989).
5) 林 達也：文解析システム YAPXR の実現と評価，情報処理学会論文誌，Vol. 31, No. 7, pp. 970-978 (1990).
6) Dahl, V. and Abramson, H.: On Gapping Grammars, *Proc. 2nd International Conference on Logic Programming*, pp. 77-88 (1984).
7) Dahl, V.: More on Gapping Grammars, *Proc. FGCS*, pp. 669-677 (1984).
8) Popowich, F.: Unrestricted Gapping Grammars, *Proc. IJCAI 85*, pp. 765-768 (1985).
9) Sugimura, R.: Logical Dependency Grammar and Its Constraint Analysis, *ICOT Technical Memorandum*, TM-0679, p. 10 (1989).

```
yapxr_2(_Outfile):-
        tell(_Outfile),
        yopen(_Ai1, _Li1, _Ti1), !,
        p0(_Ai1, _Ai2, _Li1, _Li2, _Ti1, _Ti2, [tarouga, n, ga]), !,
        p0(_Ai2, _Ai3, _Li2, _Li3, _Ti2, _Ti3, [jirouni, n, ni]), !,
        p0(_Ai3, _Ai4, _Li3, _Li4, _Ti3, _Ti4, [renrakusuru, vt, cn]), !,
        p0(_Ai4, _Ai5, _Li4, _Li5, _Ti4, _Ti5, [kotowo, n, wo]), !,
        p0(_Ai5, _Ai6, _Li5, _Li6, _Ti5, _Ti6, [yakusokusuru, vt, end]), !,
        e0(_Ai6, _Ai7, _Li6, _Li7, _Ti6, _Ti7, ['.', end, end]), !,
        yclose(_Ai7, _Li7, _Ti7),
        close(_Outfile).


yopen([[b]], [[]], [[]]):-!.

p0([[_N|_Ah]|_At], _Ao, [_Lh|_Lt], _Lo, [_Th|_Tt], _To, [_W, _C1, _C2]):- !,
        p(_N, _Ah, _Ao, _Ao1, [_W, _C1, _C2|_Lh], _Lo, _Lo1, _Th, _To, _To1),
        p0(_At, _Ao1, _Lt, _Lo1, _Tt, _To1, [_W, _C1, _C2]).
p0([], [], _, [], _, [], _):- !.
e0([[_N|_Ah]|_At], _Ao, [_Lh|_Lt], _Lo, [_Th|_Tt], _To, [_W, _C1, _C2]):- !,
        e(_N, _Ah, _Ao, _Ao1, [_W, _C1, _C2|_Lh], _Lo, _Lo1, _Th, _To, _To1),
        e0(_At, _Ao1, _Lt, _Lo1, _Tt, _To1, [_W, _C1, _C2]).
e0([], [], _, [], _, [], _):- !.

s(b, _Ai, [[e|_Ai]|_Ao], _Ao, _Li, [_Li|_Lo], _Lo, _Ti, [_Ti|_To], _To):- !.
s(_, _, _Ai, _Ai, _, _Li, _Li, _, _Ti, _Ti):- !.

prseq(b, _Ai, [[2, b|_Ai]|_Ao], _Ao, _Li, [_Li|_Lo], _Lo, _Ti, [_Ti|_To], _To):- !.
prseq(8, [_N|_Ai], _Ao, _Ao1, _Li, _Lo, _Lo1, _Ti, _To, _To1):- !,
        prseq(_N, _Ai, _Ao, _Ao1, _Li, _Lo, _Lo1, _Ti, _To, _To1).
prseq(10, _Ai, [[11|_Ai]|_Ao], _Ao, _Li, [_Li|_Lo], _Lo, _Ti, [_Ti|_To], _To):- !.
prseq(14, _Ai, [[15|_Ai]|_Ao], _Ao, _Li, [_Li|_Lo], _Lo, _Ti, [_Ti|_To], _To):- !.
prseq(_, _, _Ai, _Ai, _, _Li, _Li, _, _Ti, _Ti):- !.

rel(5, [_N|_Ai], _Ao, _Ao1, _Li, _Lo, _Lo1, _Ti, _To, _To1):- !,
        prseq(_N, _Ai, _Ao, _Ao1, _Li, _Lo, _Lo1, _Ti, _To, _To1).
rel(7, _Ai, [[8|_Ai]|_Ao], _Ao, _Li, [_Li|_Lo], _Lo, _Ti, [_Ti|_To], _To):- !.
rel(13, _Ai, [[14|_Ai]|_Ao], _Ao, _Li, [_Li|_Lo], _Lo, _Ti, [_Ti|_To], _To):- !.
rel(_, _, _Ai, _Ai, _, _Li, _Li, _, _Ti, _Ti):- !.

e(3, [_N|_Ai], _Ao, _Ao1, [_W, _C1, _C2|_Li], _Lo, _Lo1, _Ti, _To, _To1):- !,
        s(_N, _Ai, _Ao, _Ao1, _Li, _Lo, _Lo1, _Ti, _To, _To1).
e(_, _, _Ai, _Ai, _, _Li, _Li, _, _Ti, _Ti):- !.

p(2, _Ai, _Ao, _Ao1, [_W, _C1, _C2|_Li], _Lo, _Lo1, _Ti, _To, _To1):-
        ((_C2 = end, !,
          _Ao = [[3|_Ai]|_Ao1],
          _Lo = [_Li|_Lo1],
          _To = [_Ti|_To1]);
         (_Ao = _Ao1, _Lo = _Lo1, _To = _To1)).
p(b, _Ai, [[5, b|_Ai], [7, b|_Ai], [10, b|_Ai], [13, b|_Ai], [17, b|_Ai]|_Ao], _Ao,
    [_W, _C1, _C2|_Li],
        [_Li, _Li, [_W, _C1, _C2|_Li], [_W, _C1, _C2|_Li], [_W, _C1, _C2|_Li]|_Lo], _Lo,
        _Ti, [_Ti, _Ti, _Ti, _Ti, _Ti|_To], _To):- !.
p(8, _Ai, [[5, 8|_Ai], [7, 8|_Ai], [10, 8|_Ai], [13, 8|_Ai], [17, 8|_Ai]|_Ao], _Ao,
    [_W, _C1, _C2|_Li],
        [_Li, _Li, [_W, _C1, _C2|_Li], [_W, _C1, _C2|_Li], [_W, _C1, _C2|_Li]|_Lo], _Lo,
        _Ti, [_Ti, _Ti, _Ti, _Ti, _Ti|_To], _To):- !.
p(10, _Ai, [[5, 10|_Ai], [7, 10|_Ai], [10, 10|_Ai], [13, 10|_Ai], [17, 10|_Ai]|_Ao], _Ao,
    [_W, _C1, _C2|_Li],
        [_Li, _Li, [_W, _C1, _C2|_Li], [_W, _C1, _C2|_Li], [_W, _C1, _C2|_Li]|_Lo], _Lo,
        _Ti, [_Ti, _Ti, _Ti, _Ti, _Ti|_To], _To):- !.
p(14, _Ai, [[5, 14|_Ai], [7, 14|_Ai], [10, 14|_Ai], [13, 14|_Ai], [17, 14|_Ai]|_Ao], _Ao,
    [_W, _C1, _C2|_Li],
        [_Li, _Li, [_W, _C1, _C2|_Li], [_W, _C1, _C2|_Li], [_W, _C1, _C2|_Li]|_Lo], _Lo,
        _Ti, [_Ti, _Ti, _Ti, _Ti, _Ti|_To], _To):- !.
```

```
p(11, [_NI_Ai], [[10, 11, _NI_Ai], [13, 11, _NI_Ai], [17, 11, _NI_Ai]|_Ao3], _Ao1,
        [_W2, _C21, _C22, _W1, _C11, _C12|_Li],
        [[_W2, _C21, _C22, _W1, _C11, _C12|_Li],
        [_W2, _C21, _C22, _W1, _C11, _C12|_Li],
        [_W2, _C21, _C22, _W1, _C11, _C12|_Li]|_Lo3], _Lo1,
        _Ti, [_Ti, _Ti, _Ti|_To3], _To1):-
      check(_C11, _C12, _C21), !,
          p(_N, _Ai, _Ao, _Ao1, [_W1, _C11, _C12|_Li], _Lo, _Lo1,
                [(_W1, _W2)|_Ti], _To, _To1),
          rel00(_Ao, _Ao2, _Lo, _Lo2, _To, _To2, []),
          p00(_Ao2, _Ao3, _Lo2, _Lo3, _To2, _To3, [_W2, _C21, _C22]).
p(15, [_NI_Ai], [[10, 15, _NI_Ai], [13, 15, _NI_Ai], [17, 15, _NI_Ai]|_Ao3], _Ao1,
        [_W2, _C21, _C22, _W1, _C11, _C12|_Li],
        [[_W2, _C21, _C22, _W1, _C11, _C12|_Li],
        [_W2, _C21, _C22, _W1, _C11, _C12|_Li],
        [_W2, _C21, _C22, _W1, _C11, _C12|_Li]|_Lo3], _Lo1,
        _Ti, [_Ti, _Ti, _Ti|_To3], _To1):-
      check(_C11, _C12, _C21), !,
          p(_N, _Ai, _Ao, _Ao1, [_W1, _C11, _C12|_Li], _Lo, _Lo1,
                [(_W1, _W2)|_Ti], _To, _To1),
          rel00(_Ao, _Ao2, _Lo, _Lo2, _To, _To2, []),
          p00(_Ao2, _Ao3, _Lo2, _Lo3, _To2, _To3, [_W2, _C21, _C22]).
p(17, [_NI_Ai], [[10, 17, _NI_Ai], [13, 17, _NI_Ai], [17, 17, _NI_Ai]|_Ao3], _Ao1,
        [_W2, _C21, _C22, _W1, _C11, _C12|_Li],
        [[_W2, _C21, _C22, _W1, _C11, _C12|_Li],
        [_W2, _C21, _C22, _W1, _C11, _C12|_Li],
        [_W2, _C21, _C22, _W1, _C11, _C12|_Li]|_Lo3], _Lo1,
        _Ti, [_Ti, _Ti, _Ti|_To3], _To1):-
      check(_C11, _C12, _C21), !,
          p(_N, _Ai, _Ao, _Ao1, [_W1, _C11, _C12|_Li], _Lo, _Lo1,
                [(_W1, _W2)|_Ti], _To, _To1),
          rel00(_Ao, _Ao2, _Lo, _Lo2, _To, _To2, []),
          p00(_Ao2, _Ao3, _Lo2, _Lo3, _To2, _To3, [_W2, _C21, _C22]).
p(_, _, _Ai, _Ai, _, _Li, _Li, _, _Ti, _Ti):- !.

rel00(_Ai, _Ai, _Li, _Li, _Ti, _Ti, _):-
        var(_Ai), !.
rel00([[_NI_Ah]|_At], _Ao, [_Lh|_Lt], _Lo,
                [_Th|_Tt], _To, []):- !,
        rel(_N, _Ah, _Ao, _Ao1, _Lh, _Lo, _Lo1, _Th, _To, _To1),
        !,
        rel00(_At, _Ao1, _Lt, _Lo1, _Tt, _To1, []).
p00(_Ai, _Ai, _Li, _Li, _Ti, _Ti, _):-
        var(_Ai), !.
p00([[_NI_Ah]|_At], _Ao, [_Lh|_Lt], _Lo,
                [_Th|_Tt], _To, [_W1, _C11, _C12]):- !,
        p(_N, _Ah, _Ao, _Ao1, [_W1, _C11, _C12|_Lh], _Lo, _Lo1, _Th, _To, _To1),
        !,
        p00(_At, _Ao1, _Lt, _Lo1, _Tt, _To1, [_W1, _C11, _C12]).


check(n, ga, vt):- !.
check(n, ni, vt):- !.
check(n, wo, vt):- !.
check(vt, cn, n):- !.


yclose([], [], []):- !.
yclose([_Acar|_Acdr], [_Lcar|_Lcdr], [_Tcar|_Tcdr]):- !,
        print('Ai='), print(_Acar), nl,
        print('Li='), print(_Lcar), nl,
        print('Ti='), print(_Tcar), nl,
        yclose(_Acdr, _Lcdr, _Tcdr).
```

付録 例文法に対する解析プログラム

Appendix Parsing program for the example grammar.

(平成元年12月 6 日受付)
(平成2年 9 月11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，オブジェクト指向），自然言語処理（機械翻訳，自然言語インタフェイス，情報検索），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川崎研究所所属．元本学会編集幹事．1973年度本学会論文賞受賞．1984年度日経最優秀製品賞（団体）受賞．著書「電子計算機のシステムプログラム」(産報)他．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．