自然言語処理 105－4
（1995．1．20）

一意解析可能文法とその自然言語解析への応用

森田 憲一　麻生 博紀　今井 克暢

広島大学 工学部 第二類

一意解析可能文法 (uniquely parsable grammar, UPG) は，バックトラックなしに構文解析を実行できるように書換規則にある種の制約を課した文法である．UPG はこの制約にもかかわらず 0 型文法と等価な (つまり万能の) 生成能力を持つことが知られている．本稿ではまず，UPG に対する構文解析アルゴリズムで，導出のステップ数に比例した時間で解析できるものを示し，これを元に Prolog による簡潔で効率のよいパーザの構成法を与える．また，UPG を自然言語解析に使う利点と問題点を論じる．

Uniquely Parsable Grammars and Their Application to Natural Language Analysis

Kenichi MORITA, Hiroki ASOU, and Katsunobu IMAI

Faculty of Engineering, Hiroshima University, Higashi-Hiroshima-shi, 724 Japan

Uniquely parsable grammar (UPG) is a kind of generative grammar having restricted type of rewriting rules so that parsing can be done without backtracking. It is known that, in spite of such restriction, generating ability of UPG is universal. We first show a parsing algorithm that runs in time proportional to the number of steps of the derivation. Based on it, we give a simple and efficient Prolog implementation of a parser for UPG. Further, we discuss problems when using UPG as a framework for natural language analysis.

1　まえがき

一意解析可能文法 (uniquely parsable grammar, UPG) [3] は，書換規則にある種の制約を課すことにより，構文解析をバックトラックなしに (決定的に) 進められるようにした生成文法である．この制約にもかかわらず，UPG の生成能力は万能である，つまり Turing 機械によって特徴づけられることが示されている [3]．それに加えて，UPG の 3 つの部分族として定義されている 単調 UPG，RC-UPG，正規 UPG は，それらの生成能力がそれぞれ，決定性線形有界オートマトン，決定性プッシュダウン・オートマトン，有限オートマトンによって正確に特徴づけられ [3, 4]，従って古典的な Chomsky 階層に対置されるような「決定性の階層」を形成することが判明している．

UPG はこのように理論的には興味深い性質を持っているが，これを自然言語解析に使用した場合の有用性については未知の部分が多い．本稿ではまず UPG とその基本的性質について解説した後，Prolog による UPG (全体の族) のためのパーザの構成法を考察する．UPG においては構文解析を「最左還元」によって決定的に実行できることが知られている．この性質を用いると導出のステップ数に比例した時間で構文解析を実行するアルゴリズムが得られるが，これをもとに Prolog による非常に簡潔な UPG パーザの構成法を与える．本稿ではまた，UPG による日本語の断片の記述を与えると共に，これを自然言語解析に使う利点と問題点についても考察する．

2 一意解析可能文法 (UPG)

ここでは一意解析可能文法に関する諸定義と，その基本性質を述べる．

Σ を記号の集合とする．Σ 上の (長さ 0 以上の) すべての語 (記号列) の集合を Σ^* と記す．また，$\Sigma^+ = \Sigma^* - \{\varepsilon\}$ とする (ε は空列)．(なお，形式言語理論に関する基本概念に関しては例えば文献 [1] を参照のこと．)

定義 2.1 一意解析可能文法は $G = (N, T, P, S, \$)$ によって定義されるシステムである．但し，N と T はそれぞれ非終端記号と終端記号の集合 ($N \cap T = \emptyset$)，S ($\in N$) は開始記号，$\$$ ($\notin (N \cup T)$) は特別の境界記号である．また P は次のような形の書換規則の集合である (但し $\alpha, \beta \in (N \cup T)^+$, $\alpha \neq \beta$, $A \in N$ であり，α は非終端記号を 1 つ以上含む):

$$\alpha \to \beta,\quad \$\alpha \to \$\beta,\quad \alpha\$ \to \beta\$,$$
$$\$\alpha\$ \to \$\beta\$,\quad \$A\$ \to \$\$.$$

P はさらに次の条件を満たす．

UPG-条件:

1. P 中の各規則の右辺は S, $\$S$, $S\$$, $\$S\$$ のいずれでもない．
2. P 中の任意の 2 つの規則 $r_1 = \alpha_1 \to \beta_1$ と $r_2 = \alpha_2 \to \beta_2$ (r_1, r_2 は同一であってもよい) に対して次の (a),(b) が成り立つ．
   (a) ある $\delta, \beta_1', \beta_2' \in (N \cup T \cup \{\$\})^+$ が存在して $\beta_1 = \beta_1'\delta$ かつ $\beta_2 = \delta\beta_2'$ が成り立つならば，ある $\alpha_1', \alpha_2' \in (N \cup T \cup \{\$\})^*$ が存在して $\alpha_1 = \alpha_1'\delta$ かつ $\alpha_2 = \delta\alpha_2'$ となる．
   (b) ある $\gamma, \gamma' \in (N \cup T \cup \{\$\})^*$ が存在して $\beta_1 = \gamma\beta_2\gamma'$ が成り立つならば，$r_1 = r_2$ である (従って $\gamma = \gamma' = \varepsilon$ (空列))． □

UPG-条件 2(a) は，規則 r_1 の右辺のある接尾語が r_2 の右辺のある接頭語に一致するならば r_1 と r_2 の左辺もそれらをそれぞれ接尾語および接頭語として持たねばならないことを述べている．例えば規則対

$$\underline{A} \to b\underline{A} \quad と \quad \underline{A}C \to \underline{A}d$$

は UPG-条件 2(a) を満たすが，規則対

$$\underline{A} \to b\underline{A} \quad と \quad EC \to \underline{A}d$$

は満たさない．一方，UPG-条件 2(b) は，異なる規則 r_1 と r_2 で，r_2 の右辺が r_2 の右辺の部分列であるようなものは存在しないことを述べている．

UPG における導出の概念は，次のように通常の生成文法と同様に定義できる．但し UPG では，文の生成過程は開始記号 S を境界記号 $\$$ ではさんだ記号列 $\$S\$$ から始まる．

定義 2.2 $G = (N, T, P, S, \$)$ を UPG，η を $(N \cup T \cup \{\$\})$ 上の語とする．P 中の規則 $\alpha \to \beta$ が η に**適用可能**であるとは，ある $\gamma, \delta \in (N \cup T \cup \{\$\})^*$ が存在して $\eta = \gamma\alpha\delta$ と書けることをいう．η に $\alpha \to \beta$ を適用することによって $\zeta = \gamma\beta\delta$ が得られるとき，G において η から ζ が**直接的に導出**されるといい，$\eta \underset{G}{\Rightarrow} \zeta$ と書く．関係 $\underset{G}{\Rightarrow}$ の反射的推移的閉包 $\overset{*}{\underset{G}{\Rightarrow}}$ は**導出** の関係を与える．任意の $\eta, \zeta \in (N \cup T \cup \{\$\})^+$ に対し，ある $\xi_1, \xi_2, \cdots, \xi_{n-1}$ が存在して $\eta \underset{G}{\Rightarrow} \xi_1 \underset{G}{\Rightarrow} \xi_2 \underset{G}{\Rightarrow} \cdots \underset{G}{\Rightarrow} \xi_{n-1} \underset{G}{\Rightarrow} \zeta$ となるとき，$\eta \overset{n}{\underset{G}{\Rightarrow}} \zeta$ と書く．なお，文法 G が明確であるときは $\underset{G}{\Rightarrow}, \overset{*}{\underset{G}{\Rightarrow}}, \overset{n}{\underset{G}{\Rightarrow}}$ の代わりに $\Rightarrow, \overset{*}{\Rightarrow}, \overset{n}{\Rightarrow}$ を使う．語の集合 $L(G) = \{w \in T^* \mid \$S\$ \overset{*}{\underset{G}{\Rightarrow}} \$w\$\}$ を，G によって生成される**言語**という． □

例 2.1 文法 $G_{J_0} = (N_{J_0}, T_{J_0}, P_{J_0}, \langle 文 \rangle, \$)$ は日本語のある (ごくわずかの) 断片を生成する UPG である．但し，

$$N_{J_0} = \{\langle 文 \rangle, \langle 述句 \rangle, \langle 補語 \rangle, \langle 名詞句 \rangle, \langle 名詞 \rangle, \langle 動詞 \rangle, \langle 格助詞 \rangle\}$$
$$T_{J_0} = \{\text{花子}, \text{愛する}, \text{走る}, \text{が}\}$$

であり，P_{J_0} は以下の書換規則からなる．

$\$ \langle 文 \rangle \$$	→	$\$ \langle 述句 \rangle \$$
$\langle 述句 \rangle$	→	$\langle 動詞 \rangle$
$\langle 述句 \rangle$	→	$\langle 補語 \rangle \langle 述句 \rangle$
$\$ \langle 補語 \rangle$	→	$\$ \langle 名詞句 \rangle \langle 格助詞 \rangle$
$\langle 補語 \rangle \langle 補語 \rangle$	→	$\langle 補語 \rangle \langle 名詞句 \rangle \langle 格助詞 \rangle$
$\langle 名詞句 \rangle$	→	$\langle 名詞 \rangle$
$\$ \langle 名詞句 \rangle$	→	$\$ \langle 述句 \rangle \langle 名詞句 \rangle$
$\langle 名詞 \rangle$	→	花子
$\langle 動詞 \rangle$	→	愛する
$\langle 動詞 \rangle$	→	走る
$\langle 格助詞 \rangle$	→	が

このとき，文 '愛する花子が走る' の導出過程は次のようになる．

$ ⟨文⟩ $
⇒ $ ⟨述句⟩ $
⇒ $ ⟨補語⟩ ⟨述句⟩ $
⇒ $ ⟨補語⟩ ⟨動詞⟩ $
⇒ $ ⟨補語⟩ 走る $
⇒ $ ⟨名詞句⟩ ⟨格助詞⟩ 走る $
⇒ $ ⟨名詞句⟩ が 走る $
⇒ $ ⟨述句⟩ ⟨名詞句⟩ が 走る $
⇒ $ ⟨述句⟩ ⟨名詞⟩ が 走る $
⇒ $ ⟨述句⟩ 花子 が 走る $
⇒ $ ⟨動詞⟩ 花子 が 走る $
⇒ $ 愛する 花子 が 走る $

□

導出の逆過程である還元を次のように定義する.

定義 2.3 $G = (N,T,P,S,\$)$ を UPG, $\eta = x_1x_2\cdots x_m$ を $(N \cup T \cup \{\$\})$ 上の長さ m の語とする. 規則 $\alpha \to \beta \in P$ が存在し, ある i に対して $\beta = x_ix_{i+1}\cdots x_{i+|\beta|-1}$ となるならば, $\alpha \to \beta$ は η に位置 i で**逆適用可能**であるという. このような対 $[\alpha \to \beta, i]$ を η に対する**逆適用可能項**という. もし ξ がそのような逆適用によって得られる語である (つまり $\xi = x_1\cdots x_{i-1}\ \alpha\ x_{i+|\beta|}\cdots x_m$) ならば η は ξ に**直接的に還元**されるといい, $\eta \Leftarrow \xi$ あるいは $\eta \overset{[\alpha\to\beta,i]}{\Longleftarrow} \xi$ と書く. 明らかに $\eta \Leftarrow \xi$ と $\xi \Rightarrow \eta$ は同値である. 関係 $\Leftarrow$ の反射的推移的閉包 $\overset{*}{\Leftarrow}$ は**還元**の関係を与える. 関係 $\overset{n}{\Leftarrow}$ は $\overset{n}{\Rightarrow}$ と同様に定義できる. また, 語 η は $\eta \Leftarrow \xi$ となる ξ が存在するとき**還元可能**, そうでないとき**還元不能**という. □

次に還元の特別な場合である最左還元を定義する. これは最左の位置にある逆適用可能項によって行う還元のことである.

定義 2.4 $G = (N,T,P,S,\$)$ を UPG とする. G における直接的な還元 $\eta \overset{[\alpha\to\beta,i]}{\Longleftarrow} \zeta$ が**直接的な最左還元**であるといわれるのは, η に対する任意の逆適用可能項 $[\alpha' \to \beta', i']$ に対して $i \le i'$ が成り立つことをいう. 直接的な最左還元は $\eta \overset{[\alpha\to\beta,i]}{\underset{\text{lmr}}{\Longleftarrow}} \zeta$ または $\eta \underset{\text{lmr}}{\Leftarrow} \zeta$ と書かれる. また, 還元 $\eta_0 \Leftarrow \eta_1 \Leftarrow \cdots \Leftarrow \eta_n$ が**最左還元**であるといわれるのは $\eta_0 \underset{\text{lmr}}{\Leftarrow} \eta_1 \underset{\text{lmr}}{\Leftarrow} \cdots \underset{\text{lmr}}{\Leftarrow} \eta_n$ が成り立つことをいう. さらに η_0 から η_n への (長さ n の) 最左還元が存在するとき, $\eta_0 \overset{*}{\underset{\text{lmr}}{\Leftarrow}} \eta_n$ ($\eta_0 \overset{n}{\underset{\text{lmr}}{\Leftarrow}} \eta_n$) と書く. □

例 2.1 における '愛する花子が走る' の導出過程を逆にたどったものは最左還元過程になっている.

UPG では, 1 つの語に対して逆適用可能項が 2 つ以上あった場合, その中のどれを先に逆適用しても他のものの逆適用可能性に影響を与えることはなく, しかも逆適用の順序に無関係に最終的には同一の語に還元される. これは次の事実に基づく (正確な証明は [3] 参照). もし 2 つの書換規則

$$\alpha_1 \to \beta_1'\delta, \quad \alpha_2 \to \delta\beta_2',$$

があったなら, UPG-条件によりそれらは

$$\alpha_1'\delta \to \beta_1'\delta, \quad \delta\alpha_2' \to \delta\beta_2'.$$

の形でなければならない. 従って, それらが共にある語 η に逆適用可能ならば, 図 1のように, 逆適用の順序によらず同一の語 σ に還元される.

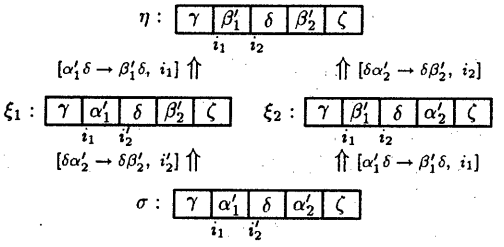


図 1: UPG における還元の合流性

このような性質から UPG の還元過程に関する次の定理が成り立つことが示されている.

定理 2.1 [3] $G = (N,T,P,S,\$)$ を UPG, η を $(N \cup T \cup \{\$\})$ 上の語とする. もし $\eta \overset{n}{\Leftarrow} \$S\$$ ならば, η に対する任意の逆適用可能項 $[\alpha \to \beta, i]$ および $\eta \overset{[\alpha\to\beta,i]}{\Longleftarrow} \xi$ となる語 ξ に対して関係 $\eta \overset{[\alpha\to\beta,i]}{\Longleftarrow} \xi \overset{\eta-1}{\Longleftarrow} \$S\$$ が成り立つ.

この定理は, η から $\$S\$$ への n ステップの還元があるならば, 最初のステップで η をどの逆適用可能項によって還元しても, 全体として必ず n ステップで $\$S\$$ に還元できることを示している. この定理の系として特に, 任意の語を最左還元によって一意に (バックトラックなしに) 解析できるという結果が得られる.

系 2.1 [3] $G = (N,T,P,S,\$)$ を UPG, η を $(N \cup T \cup \{\$\})$ 上の語とする. このときもし $\eta \overset{n}{\Leftarrow} \$S\$$ ならば, $\eta \overset{n}{\underset{\text{lmr}}{\Leftarrow}} \$S\$$ となる.

3 Prolog による UPG パーザ

前節の系 2.1 によって述べられるように UPG の構文解析は最左還元に基づいて実行できるため，パーザの構成が容易である．また，後で示すように導出のステップ数に比例した時間で解析が可能である (定理 3.1)．ここでは，このようなパーザが Prolog によって大変簡潔にインプリメントできることを示す．

以下では UPG の書換規則を

$$B_1 \neq C_1 \ \vee \ j = 0 \ \vee \ k = 0$$

を満たすような $A_1, \cdots, A_i, B_1, \cdots, B_j, C_1, \cdots, C_k \in (N \cup T \cup \{\$\})$ $(i, j, k \in \{0, 1, \cdots\})$ を用いて

$$A_1 \cdots A_i B_1 \cdots B_j \rightarrow A_1 \cdots A_i C_1 \cdots C_k$$

の形に書く．但し，$k = 0$ であるような形の規則

$$A_1 \cdots A_i B_1 \cdots B_j \rightarrow A_1 \cdots A_i$$

は $\$S\$$ からの導出には決して使用されないので予め除去しておく．なぜなら，もし $\$S\$ \stackrel{*}{\Rightarrow} \eta$ である語 η にこの規則が逆適用可能だとすると，η から始まる無限長の還元過程が存在し，定理 2.1 に矛盾するからである．

Prolog によるパーザを示す前に次のアルゴリズムを考える．

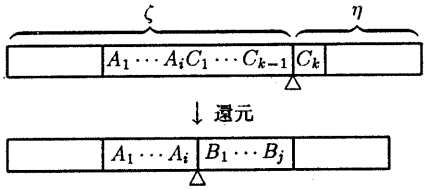
アルゴリズム A

入力： UPG $G = (N, T, P, S, \$)$ および
　　記号列 $d_1 \cdots d_n \in T^*$.
出力： $d_1 \cdots d_n \in L(G)$ であるとき，かつそのときにのみ "yes".

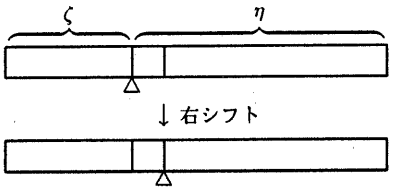
1. **begin**
2. 　$\zeta := \$$, $\eta := d_1 \cdots d_n\$$ とする ;
3. 　**while** $\eta \neq \varepsilon$ **do**
4. 　　**if** 規則 $A_1 \cdots A_i B_1 \cdots B_j \rightarrow A_1 \cdots A_i C_1 \cdots C_k \in P$ が存在し，$\zeta = \zeta' A_1 \cdots A_i C_1 \cdots C_{k-1}$, $\eta = C_k \eta'$ と書ける
5. 　　**then** $\zeta := \zeta' A_1 \cdots A_i$, $\eta := B_1 \cdots B_j \eta'$ とする
6. 　　**else** η の左端の記号を除去して，それを ζ の右端につける ;
7. 　**if** $\zeta = \$S\$$ かつ $\eta = \varepsilon$ **then** "yes" と出力
8. **end**

このアルゴリズムは入力語を左から右に走査し，逆適用可能項が見つかればそれによって還元を行うもので，次の 2 つの基本操作からなる (下図では走査位置のポインタを △ で示す; η の左端が各時点の走査位置に対応)．

(1) 還元操作 (アルゴリズム A の 5 行目)



(2) 右シフト操作 (アルゴリズム A の 6 行目)



定理 3.1　アルゴリズム A は $d_1 \cdots d_n \in L(G)$ であるとき，かつそのときにのみ "yes" を出力する．また，$\$S\$ \stackrel{m}{\Rightarrow} \$d_1 \cdots d_n\$$ であるような入力 $d_1 \cdots d_n$ に対する A の計算時間は $O(m)$ である．

[証明]　まず定理の前半，つまりアルゴリズムの正当性を示す．3 行目の **while** 文の h 回目の実行後の ζ と η の値を ζ_h, η_h で表す．但し $\zeta_0 = \$$, $\eta_0 = d_1 \cdots d_n\$$ である．まず ζ_h, η_h が定義されるような任意の h に対して命題 (a) が成り立つことを h に関する帰納法で示す．

(a) $\$d_1 \cdots d_n\$ \stackrel{*}{\underset{\text{lmr}}{\Leftarrow}} \zeta_h \eta_h$ かつ ζ_h は還元不能．

$h = 0$ のときは明らか．$h < H$ のときに成り立つと仮定し，**while** 文の H 回目の実行を考える．4 行目の条件に合致した場合，ζ_{H-1} が還元不能であること，G が UPG であること，及び 5 行目より，$\zeta_{H-1}\eta_{H-1} \underset{\text{lmr}}{\Leftarrow} \zeta_H \eta_H$ となり，従って $\$d_1 \cdots d_n\$ \stackrel{*}{\underset{\text{lmr}}{\Leftarrow}} \zeta_H \eta_H$．また，$\zeta_H$ は ζ_{H-1} の接頭語なので還元不能．4 行目の条件に合致しなかった場合，6 行目により得られる ζ_H と η_H は $\zeta_{H-1}\eta_{H-1} = \zeta_H \eta_H$ を満たす．従って，$\$d_1 \cdots d_n\$ \stackrel{*}{\underset{\text{lmr}}{\Leftarrow}} \zeta_H \eta_H$．また，4 行目の条件が成立しなかったのだから ζ_H は還元不能となる．

次に，任意の h に対して命題 (b) が成り立つことを示す．

(b) $\zeta_h \eta_h$ が還元可能なら $\exists H > h$ $(\zeta_h \eta_h \underset{\text{lmr}}{\Leftarrow} \zeta_H \eta_H)$．

$\zeta_h \eta_h \underset{\text{lmr}}{\Leftarrow} \zeta_H \eta_H$ となる $H (> h)$ が存在しないと仮定する．そうすると，h 回目より後の **while** 文の実行中に 5 行目の操作により還元されることは有り得ない ((a) よりこれが最左還元となるから)．従って 6 行目の操作だけが何回も反復され，最終的に

はある H' に対して $\zeta_{H'} = \zeta_h\eta_h$, $\eta_{H'} = \varepsilon$ となる. しかし $\zeta_h\eta_h$ が還元可能という仮定から, $\zeta_{H'}$ も還元可能となり (a) に矛盾する.

以上をもとに次の命題を示す.

(c) $\$d_1\cdots d_n\$ \overset{*}{\underset{\mathrm{lmr}}{\Leftarrow}} \xi$ となるための必要十分条件は $\exists h\ (\zeta_h\eta_h = \xi)$.

必要性: 任意の m に対し, $\$d_1\cdots d_n\$ \overset{m}{\underset{\mathrm{lmr}}{\Leftarrow}} \xi$ ならば $\exists h\ (\zeta_h\eta_h = \xi)$ となることを帰納法で証明. $m = 0$ のときは明らか. $m < M$ のときに成り立つと仮定. 次の還元を考える. $\$d_1\cdots d_n\$ \overset{M-1}{\underset{\mathrm{lmr}}{\Leftarrow}} \xi_0 \underset{\mathrm{lmr}}{\Leftarrow} \xi_1$. 帰納法の仮定より, ある H_0 が存在して $\xi_0 = \zeta_{H_0}\eta_{H_0}$ となる. $\zeta_{H_0}\eta_{H_0}$ が還元可能なので, (b) よりある H_1 が存在して $\zeta_{H_0}\eta_{H_0} \underset{\mathrm{lmr}}{\Leftarrow} \zeta_{H_1}\eta_{H_1} = \xi_1$ となる. 従って, $\$d_1\cdots d_n\$ \overset{M}{\underset{\mathrm{lmr}}{\Leftarrow}} \xi_1$ ならば $\exists h\ (\zeta_h\eta_h = \xi_1)$.

十分性: 命題 (a) より明らか.

さて, (c) より $\$d_1\cdots d_n\$ \overset{*}{\underset{\mathrm{lmr}}{\Leftarrow}} \$S\$$ であるための必要十分条件が $\exists h\ (\zeta_h\eta_h = \$S\$)$ となることがわかる. $\$d_1\cdots d_n\$ \in L(G)$ の場合を考える. UPG の定義より $\$S\$$ は還元不能なので, 最終的には $\zeta = \$S\$, \eta = \varepsilon$ となって **while** 文の実行を終了し, 7 行目の条件に合致して "yes" が出力される. 一方, $\$d_1\cdots d_n\$ \notin L(G)$ の場合には $\zeta_h\eta_h = \$S\$$ となる h が存在しないので, "yes" が出力されることはない.

次に計算時間の評価を行う. アルゴリズム A による最左還元過程 $\$d_1\cdots d_n\$ \overset{m}{\underset{\mathrm{lmr}}{\Leftarrow}} \$S\$$ を考える. P 中の規則の左辺と右辺の長さの最大値をそれぞれ L, R とおく. A の 2 行目により η の長さの初期値は $n+1$ となる. η の長さが増加し得るのは 5 行目が実行された場合だけであり, 規則 $\alpha \to \beta$ の逆適用により $|\alpha| - |\beta| \leq L - 1$ だけ増加する. 5 行目は m 回実行されるので, η の長さの増加は合計で高々 $m(L-1)$ である. 従って 6 行目の実行回数が高々 $n + 1 + m(L-1)$ 回になることがわかる (6 行目の実行により η の長さが 1 減少するので). 一方, 導出 $\$S\$ \overset{m}{\Rightarrow} \$d_1\cdots d_n\$$ を考えると $n - 1 \leq m(R-1)$ が成り立つことがわかる (規則 $\alpha \to \beta$ の適用による記号列の長さの増加が $|\beta| - |\alpha| \leq R - 1$ なので). よって 6 行目の実行回数は高々 $(L + R - 2)m + 2$ となる.

A の 2,4,5,6,7 行目はそれぞれ O(1) で実行できる. また, 3-6 行目の **while** 文の実行回数は 5 行目と 6 行目の実行回数の和, すなわち $m + (L + R - 2)m + 2 = (L + R - 1)m + 2$ である. 以上より A の実行時間は $O(m)$ である. □

アルゴリズム A は次に示すように, 非常に単純な Prolog のプログラムとして実現できる. ここでは 2 つの述語 parse と rule を用いる. parse(X,Y) はアルゴリズム A の制御構造を実現するもので, その引数 X は ζ の反転列を, Y は η を保持するのに使うリストである. また述語名 rule を持つ宣言節は各々の書換規則に対応している.

Prolog による UPG パーザの構成法

1. 述語 parse を次の 3 つの節によって定義する. 但し s は開始記号 S に対応するアトムとする.

```
parse(['$',s,'$'],[]) :- !.
parse(X,Y) :-
  rule(U,V,X,Y),!,parse(U,V).
parse(X,[H|Y]) :- !,parse([H|X],Y).
```

2. P 中の各規則

$$A_1\cdots A_iB_1\cdots B_j \to A_1\cdots A_iC_1\cdots C_k$$

に対して次の節を加える. 但し, $A_1,\cdots,A_i$, $B_1,\cdots,B_j$, $C_1,\cdots,C_k$ に対応するアトムをそれぞれ $a_1,\cdots,a_i,b_1,\cdots,b_j,c_1,\cdots,c_k$ とする.

```
rule([a_i,...,a_1|X],[b_1,...,b_j|Y],
     [c_{k-1},...,c_1,a_i,...,a_1|X],[c_k|Y]).
```

3. 入力記号列 $d_1d_2\cdots d_n \in T^*$ に対する解析は, 次の問い合わせ節によって開始される.

```
?- parse(['$'],[d_1,...,d_n,'$']).
```

例 3.1 例 2.1に与えた文法 G_{J_0} に対する Prolog パーザを示す. ここでは parse に引数を追加した述語 parse1 を用いて解析木を複合項の形で得られるようにしている.

```
parse1(['$',文(T),'$'], [], T) :- !.
parse1(X,Y,T) :-
  rule(U,V,X,Y),!,parse1(U,V,T).
parse1(X,[H|Y],T) :- !,parse1([H|X],Y,T).
rule(['$'|X],[文(文(T)),'$'|Y],
     [述句(T),'$'|X],['$'|Y]).
rule(X,[述句(述句(T))|Y], X,[動(T)|Y]).
rule(X,[述句(述句(S,T))|Y],
     [補(S)|X],[述句(T)|Y]).
rule(['$'|X],[補(補(S,T))|Y],
     [名句(S),'$'|X],[格(T)|Y]).
rule([補(R)|X],[補(補(S,T))|Y],
     [名句(S),補(R)|X],[格(T)|Y]).
rule(X,[名句(名句(T))|Y], X,[名(T)|Y]).
rule(['$'|X],[名句(名句(S,T))|Y],
     [述句(S),'$'|X],[名句(T)|Y]).
rule(X,[名(名(花子))|Y], X,[花子|Y]).
rule(X,[動(動(愛する))|Y], X,[愛する|Y]).
rule(X,[動(動(走る))|Y], X,[走る|Y]).
rule(X,[格(格(が))|Y], X,[が|Y]).
```

```
?- parse1(['$'],[愛する,花子,が,走る,'$'],T).
T = 文(述句(補(名句(述句(動(愛する))),名句(名(花子))),格(が)),述句(動(走る))))
```

4 自然言語解析における問題点

UPG は導出の過程を(最左還元によって)一意に逆にたどれる文法であるため，適切に設計された UPG に対する構文解析は大変高速である(但し UPG (全体の族) は，その能力が Turing 機械と等価であるため，導出過程自体が非常に長くなるような言語も生成し得る)．

自然言語解析においてももちろんこのような解析の高速性は非常に有用であるがその反面，(1) 文脈自由文法に比べて文法記述が困難，(2) 構文上の曖昧性が扱えない，などの問題がある．

まず (1) に関しては，UPG 条件を満たすように文法を構成するのに技術を要するという問題がある(現時点では UPG 条件のチェックを行うソフトウェアを使用しながら試行錯誤的に文法設計を行っている)．しかし UPG 条件は書換規則の表面的な形に関する条件であるので，UPG と同様に決定性の解析が可能な LL(k) 文法 [5] や LR(k) 文法 [2] における条件に比べて単純で直観的理解が容易である．また UPG はこれらとは異なり，生成能力が文脈自由言語の部分族に限定されてはいない．

一方，構文上の曖昧性が積極的に扱えないという (2) の問題に関しては UPG の性質上致し方のないことであるが，逆に，局所的な構造を手がかりにしながら曖昧さなしに構文解析を進めたいときには UPG の枠組みが有利になる場合がある．次の例のように被修飾語句の位置を限定してしまう場合は UPG によって比較的容易に記述できる．

例 4.1 以下の書換規則からなる UPG G_{J_1} は例 2.1 の文法 G_{J_0} よりいくらか広い日本語の断片を生成する UPG であり，次の特徴を持つ．

1. 修飾語句 (⟨補語⟩ と ⟨無題述句⟩) はそれがかかり得る最も近い位置の語句 (⟨無題述句⟩ と ⟨名詞句⟩) を修飾する．
2. 上述の修飾語句の後に読点 (‘、’) がある場合は二番目に近い位置の語句を修飾する．

従って例えば文 ‘太郎が家にいる花子に電話をかける’ では ‘太郎が’ は ‘いる’ を修飾するが，文 ‘太郎が、家にいる花子に電話をかける’ では ‘太郎は’ は ‘かける’ を修飾する (つまり後者を表現する場合には読点をつけねばならない)．

$ ⟨文⟩ $ → $ ⟨無題述句⟩ $
$ ⟨文⟩ $ → $ ⟨有題述句⟩ $
⟨無題述句⟩ → ⟨述語⟩
X ⟨無題述句⟩ → X ⟨補語⟩ ⟨無題述句⟩
⟨補語⟩ ⟨読点⟩ ⟨無題述句⟩
→ ⟨補語⟩ ⟨読点⟩ ⟨補語⟩ ⟨無題述句⟩
⟨無題述句⟩ ⟨無題述句⟩
→ ⟨無題述句⟩ ⟨読点⟩ ⟨補語⟩ ⟨無題述句⟩
⟨有題述句⟩ $ → ⟨題目⟩ ⟨無題述句⟩ $
⟨有題述句⟩ $ → ⟨題目⟩ ⟨有題述句⟩ $
⟨有題述句⟩ → ⟨補語⟩ ⟨有題述句⟩
Y ⟨補語⟩ → Y ⟨名詞句⟩ ⟨格助詞⟩
Y ⟨題目⟩ → Y ⟨名詞句⟩ ⟨係助詞⟩
⟨述語⟩ → ⟨動詞⟩
Y ⟨述語⟩ → Y ⟨名詞句⟩ ⟨断定辞⟩
⟨名詞句⟩ → ⟨名詞⟩
Z ⟨名詞句⟩ → Z ⟨無題述句⟩ ⟨名詞句⟩
⟨補語⟩ ⟨名詞句⟩ → ⟨補語⟩ ⟨読点⟩ ⟨無題述句⟩ ⟨名詞句⟩
⟨名詞⟩ → 太郎 | 花子 | 家 | 電話
⟨動詞⟩ → いる | かける | 叫ぶ
⟨格助詞⟩ → が | に | を
⟨係助詞⟩ → は
⟨断定辞⟩ → である
⟨読点⟩ → 、
但し，$X \in \{\$, ⟨補語⟩, ⟨題目⟩\}$,
$Y \in \{\$, ⟨補語⟩, ⟨題目⟩, ⟨読点⟩\}$,
$Z \in \{\$, ⟨題目⟩, ⟨無題述句⟩\}$.

5 むすび

本稿では，UPG に対する Prolog による簡潔なパーザの構成法を与えると共に，その応用における問題点を論じた．自然言語の解析においては，非終端記号に「素性」を付随させ，それらの単一化 (unification) を行いながら解析を進めていく方法がしばしばとられる．それゆえ UPG の枠組みを，definite clause grammar (DCG) のように非終端記号が「引数」を持つような，ある種のユニフィケーション文法に拡張することが今後の課題である．

参考文献

[1] J. E. Hopcroft, and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, Reading, Massachusetts (1979).

[2] D. E. Knuth, On the translation of languages from left to right, *Inf. Control*, **8**, 607–639 (1965).

[3] 森田憲一, 山本泰典, 西原典孝, 張治国, 一意解析可能文法, 信学技報 COMP93-69 (1994).

[4] 西原典孝, 森田憲一, 決定性プッシュダウン・オートマトンおよび有限オートマトンと等価な一意解析可能文法 (UPG) のサブクラス, 信学技報 COMP94-23 (1994).

[5] D.J. Rosenkrantz, and R. E. Stearns, Properties of deterministic top-down grammars, *Inf. Control*, **17**, 226–256 (1970).