左外置を考慮したボトムアップ構文解析システム

今野 聡　田中 穂積

英語における関係代名詞節やWh疑問文などは，埋め込み文の名詞句がその左方に移動して形成されると考えることができるが，痕跡理論では，名詞句が移動しても，その後に痕跡が残るため，句構造は変化しないと考えることが出来る．本稿では，その考え方を文法記述に反映した文法記述形式について検討し，DCGを拡張した新しい文法記述形式について述べる．また，その形式で記述した文法規則を用いて，ボトムアップに構文解析を行う場合の問題点について述べるとともに，その一解決法を提案する．また，提案した解決法を用いて行った実験結果を述べ，本解決法の有効性を示す．

1 はじめに

DEC-10 PrologやC-Prologには，Edinburgh大学のPereiraとWarrenが開発したDCG(Definite Clause Grammar)と呼ばれる文法記述形式がインプリメントされている[8]．このDCGは，文法規則の各非終端記号を述語と見なし，それに任意個の引数を持たせることができる補強文脈自由文法で，各文法規則は1対1に対応するPrologプログラムに変換され，それらがそのままトップダウン・パーザとして動作する．しかし，トップダウン・パーザには，文法規則の中に左再帰規則があると無限ループに陥るという欠点がある．この問題は，電子技術総合研究所の松本らが開発したボトムアップ構文解析システムBUPにより解決された[5][6]．筆者らは，このBUPを用いて英語の構文解析実験を行い，DCGで記述した約400の文法規則からなる文法を作成したが[11]，規則の数が増加するにつれ，文法体系の見通しが悪くなり，その保守も困難になるため，DCG以上の記述能力を持つ文法記述形式の必要性を認識した．

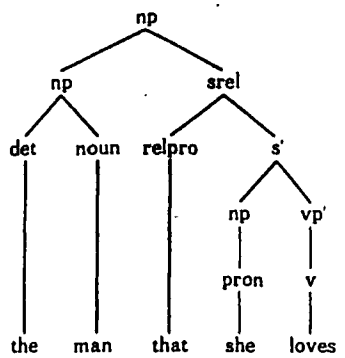
英語における関係代名詞節の埋め込み文は，宣言文中の名詞句が1つ欠落した構文構造をしているが，これは，先行詞が文の左方に移動してできると考えられる．Chomskyの痕跡理論(trace theory)では，語句が移動する場合，移動前の場所に痕跡(trace)を残して移動するとしており，このような語句の移動を左外置(left-extraposition)と呼んでいる．この左外置による痕跡を，システムにサーチさせ発見させることを前提として文法規則を記述すると，そうしない場合にくらべて，文法カテゴリの数や文法規則の数を減らすことができる．その結果，文法の見通しが良くなる．本論文では，前者の立場で文法規則を記述する形式を「トレース・サーチ」形式と呼び，そうしない形式を「非トレース・サーチ」形式と呼ぶ．また，トレース・サーチ形式で記述した文法を用いる構文解析を，「左外置を考慮した構文解析」と呼ぶ．

本研究では，トレース・サーチ形式による文法記述形式として，DCGを拡張した新しい文法記述形式を提案し，それにより英語の文法を記述して，その有効性を確認している．また，左外置を考慮した構文解析をボトムアップ法で行うことは，従来効率の面で問題があると考えられていたが，本研究では，先に述べたBUPを拡張して，左外置を考慮した構文解析を，ボトムアップ法でも効率よく行うことができることを示す．以下では，このBUPの拡張版をBUP-XGと呼ぶ[2][3][12]．

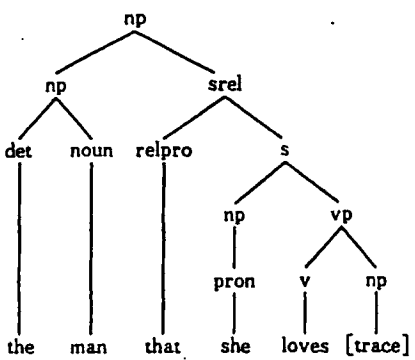
Processing Left-extraposition in Bottom up Parsing System.

Satoshi Konno, Hozumi Tanaka, 東京工業大学情報工学科.

コンピュータソフトウェア, Vol. 3, No. 2(1986), pp. 19-29.
[論文] 1985年7月18日受付.



(a) 左外置による痕跡を考慮しない場合の句構造



(b) 左外置による痕跡を考慮した場合の句構造

図 1 名詞句 "the man that she loves" の句構造

2 左外置と文法記述

2.1 左外置と句構造

英語における関係代名詞節や Wh 疑問文は，先行詞や疑問代名詞が，埋め込み文の左に移動してできると考えることができるが，その結果，宣言文や Yes-no 疑問文と比べ，名詞句が欠落した句構造ができる．たとえば，名詞句 "the man that she loves" の場合，埋め込み文の目的語が欠けているため，その句構造は，図 1(a) のようになる．この図より分かるように，名詞句が 1 つ欠けているため，完全な宣言文や動詞句のカテゴリ s, vp と区別するため，"she loves", "loves" に対して，それぞれ s', vp' を導入している．一方，痕跡理論による痕跡を考慮すると，語句の移動後もその痕跡が存在するため，移動前の句構造が維持されると考えることができるので，先の名詞句の句構造は，図 1(b) に示すものになる．本研究では，以上のように，痕跡を考慮することで語句が移動しても句構造が変化しないことに着目し，その特質を

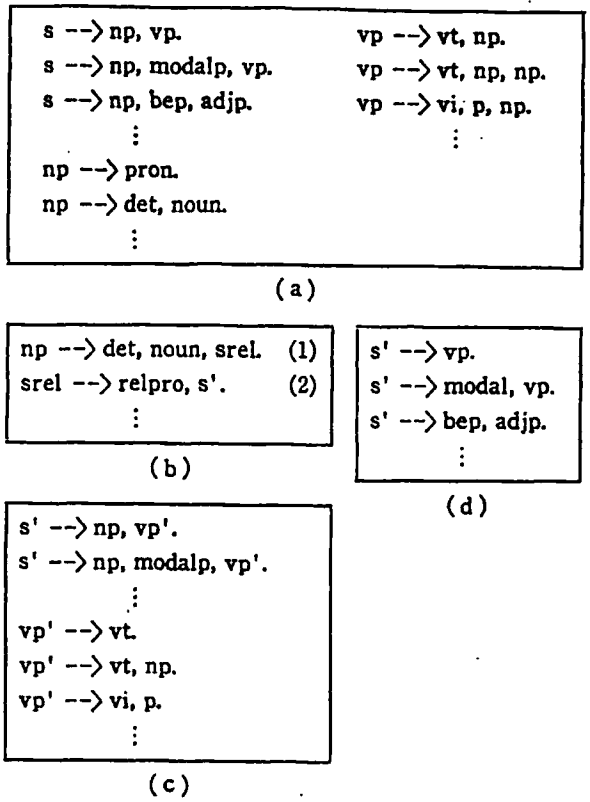


図 2 非トレース・サーチ形式による文法記述例

活かした文法記述形式を提案する (2.3)．それによれば，関係代名詞節に対する規則を記述する際に，名詞句の欠落した埋め込み文に対しても，図 1 に示すように，完全な宣言文と同じカテゴリ s を割り当てることができる．そして，具体的に痕跡が存在すると考えられる個所はシステムにサーチさせ発見させることにより，文法記述者が記述する規則の数を抑え，文法の見通しを良くすることができる．

2.2 非トレース・サーチ形式による文法記述

2.3 で提案する文法記述形式 XGS を示す前に，非トレース・サーチ形式による文法記述の問題点について簡単に述べる．文法規則の記述に，トレース・サーチ形式を用いない場合，文法記述者は，名詞句の欠落した文法カテゴリに対応する文法規則を 1 つ 1 つ記述しなければならない．たとえば，宣言文を解析するための文法規則として，図 2(a) に示す規則があり，ここで，関係代名詞節を解析するための文法規則を追加する場合について考える．まず始めに，図 2(b) に示すように，関係代名詞節を含む名詞句の規則 (1) と関係代名詞節に対する規則 (2) を付加する．ここで，(2) における s' は，図 1 で使用

したカテゴリと同様に，名詞句の欠落した宣言文のカテゴリとして導入してある．さらに，そのs' に対する規則であるが，まず目的語が欠けた文である場合には，図1(a)に示したように，名詞句の欠落した動詞句のカテゴリ vp' を導入する．そして，s' および vp' に対する規則として，図2(a)に示す宣言文と動詞句の規則から，図2(c)に示す規則を作成する．さらに，主語が欠けている埋め込み文を解析するための規則として，宣言文の規則から主語となる名詞句のカテゴリを取り除き，図2(d)の規則を作成する．しかし，s', vp' といった不自然な非終端記号の導入と，そのための文法規則の追加は，文法の見通しを悪くする原因となる．

2.3 トレース・サーチ形式による文法記述

左外置による痕跡を考慮すると，関係節の埋め込み文も，完全な宣言文の構造をしているとみなすことができるため，埋め込み文に対しても，宣言文と同じ非終端記号を割り当てることができる．したがって，非トレース・サーチ形式のように，名詞句が欠落した句構造に対して，s', vp' といった新しいカテゴリを導入したり，そのための規則を記述する必要がなくなる．

この形式による文法記述形式として代表的なものに，Pereira が開発した XG (Extraposition Grammar) [9] [10] がある．この XG は，文法規則の左辺に2つ以上のカテゴリが存在するなど，文脈自由文法規則の形式とは異なる文法記述形式を使用しており，本項で述べる XGS を提案する動機にもなっている．また，近年注目されている文法に，Gazdar らが開発した GPSG (Generalized Phrase Structure Grammar) [1] があるが，この文法もトレース・サーチを前提にしていると考えることができる．周知のとおり，GPSG は，シンタクスの面では，自然言語の文法は文脈自由文法でほぼ完全に記述できると主張している．ただし，文法を文脈自由文法で記述するために，カテゴリと文法規則数の発散を防ぐため，feature や metarule といった概念を導入している．なかでも痕跡を含む文法カテゴリを扱うために，slash という feature をもちいている．

我々が提案する文法記述形式 XGS (Extraposition Grammar with Slash Category) は，DCG の持つ文法規則の読みやすさを保ったままで，痕跡の概念を容易に扱えることを目的として設計された文法記述形式で，

```
np   --> det, noun, srel../np.      (3)
srel --> relpro, s.                 (4)
```

図 3 XGS による文法規則例

DCG のシンタクスを拡張した形になっている．以下では，文法規則の記述例をもとに，XGS のシンタクスおよびセマンティクスの概略を説明する．

ここで再び，2.2 で行ったように，図2(a)に示す規則が与えられているとして，関係代名詞節を解析するための規則を付加することについて考える．非トレース・サーチ形式を用いた場合には，図2(a)の規則の他に，図2(b)～(d)の規則を追加する必要があったが，本稿で提案するトレース・サーチ形式の XGS で記述する場合には，図3に示す規則を追加するだけでよい．ここで，規則(3)中の "../" (スラッシュと呼ぶ) が，DCG のシンタクスに対して新たに付加したシンタクス・シュガーである．このスラッシュは二項演算子となっており，スラッシュの後のカテゴリを「スラッシュ・カテゴリ」と呼ぶ．a../b は，カテゴリ a の中に痕跡が1つ存在し，その痕跡はスラッシュ・カテゴリ b の直接構成要素であることを表している．したがって，(3)では，srel../np と記述することで，痕跡を直接支配する np が srel の内部に1つ存在することを表している．なお図4は，スラッシュ・カテゴリ np が痕跡を直接支配している様子を図式化したもので，これを，「スラッシュ・カテゴリと痕跡との対応付け」と呼ぶ．また(4)より分かる通り，痕跡を考慮することで，埋め込み文のカテゴリも宣言文のカテゴリと同じ s となるため，2.2 で述べたように名詞句の欠落したカテゴリを導入したり，そのための規則を記述したりする必要がないことに注意されたい．

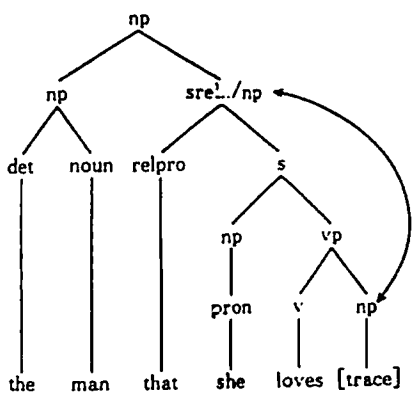


図 4 痕跡との対応

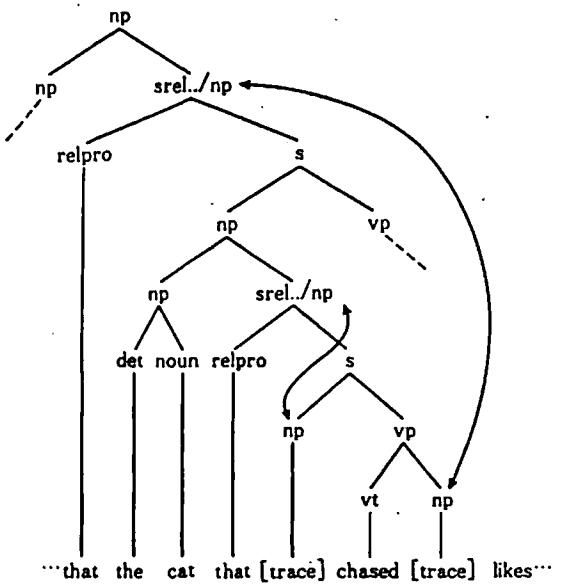


図 5 複合名詞句制約に違反した文

このように，上で示したシンタクスによって，関係節に関する規則を記述することが可能になるが，英語の場合には，さらに次のような問題がある．

The mouse that the cat that chased likes fish squeaks. (5)

上の文は，関係節の埋め込み文が二重になっている例であるが([9]より引用)，一番下の埋め込み文から，名詞句が2つ移動した形になっている非文法文であり，この種の文は，生成文法においては，複合名詞句制約(complex-NP constraint)と呼ばれる制約により生成が禁止されている．ところが，図3の規則を用いて解析すると図5に示したような痕跡と語句との対応付けができてしまうため，解析に成功してしまう．これは，srel../npでは，名詞句の痕跡が，関係節の埋め込み文のどこにあるかということが規定できないため，痕跡をその構成要素の中の埋め込み文の中でみつけしだい，痕跡との対応付けを行ってしまうことによる．そこで，さらに2つのシンタクス・シュガー"〈"(オープン)，"〉"(クローズ)を加え，システムが痕跡をサーチする場合でも，オープンとクローズで囲まれているカテゴリの中では，痕跡のサーチを行わないことを指定出来るようにした．下の(6)は，オープンとクローズをもちいて先の規則(3)を書き換えたものであるが，この規則を用いれば，〈，〉で囲まれたsrel中の痕跡は，detとnounからなる名詞句以外とは対応付けができないため，複合名詞句制約に違反した非文法文の解析に成功することはなくなる．同様な手法は，XGではopen, closeという終端記号を導入して行っている[9]．

np —〉 det, noun, 〈srel../np〉. (6)

3 BUP-XGにおける左外置を考慮した構文解析の基本メカニズム

3.1 トップダウン法と左外置を考慮した構文解析

左外置を考慮した構文解析をトップダウン法で行う場合に，スタックを用いて実行時に痕跡をサーチする方法が考えられている．(なお，この方法は，PereiraがXGで採用している方法である．詳しくは文献[9],[10]を参照されたい．)

関係代名詞節を解析する場合，その埋め込み文の解析中に痕跡を発見しなくてはならない．そこで，痕跡を直接支配する名詞句のカテゴリnpをスタックにプッシュまたはポップすることで，痕跡のサーチに関する処理を行う．以下では，図1の例文の解析過程を例にとり説明する．

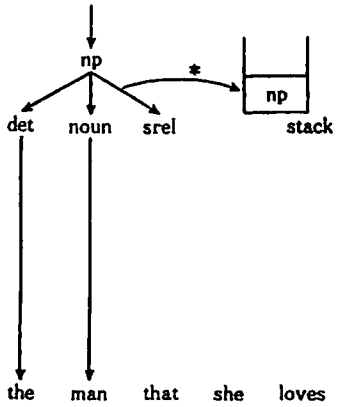
(1) 限定詞det(a, the, everyなど)と名詞nounが見つかり，続いて関係代名詞節の解析を始める前に，痕跡を支配する名詞句のカテゴリnpをスタックにプッシュする(図6(a))．

(2) 名詞句の解析に失敗した時点で，スタックトップがnpであればスタックからnpをポップして，痕跡を発見したと解釈し，名詞句の解析を成功させる(図6(b))．

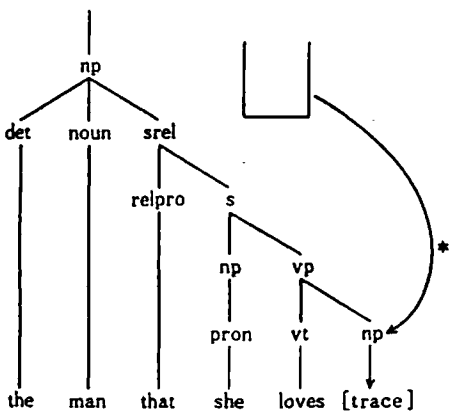
この方法は，ATNで関係代名詞節の解析のために使用するHoldレジスタを用いたHold，Unholdメカニズムに相当する[13]．すなわち，(1)でカテゴリnpをスタックへプッシュすることは，ATNではHoldレジスタにnp(および先行詞のデータ)のコピーをセットすることに対応し，痕跡を持つnpの存在を予測していることになる．また，(2)でスタックをポップしてnpの解析を成功させることは，ATNではHoldレジスタを空にすることに対応している．

左外置を考慮した構文解析を，スタックを用いたトップダウン法で行う場合には次のような特徴がある．

(1) 痕跡のサーチは，痕跡を支配するカテゴリをスタックにプッシュすることで開始される．したがって，痕跡のサーチを行うのは，痕跡を持ちうるカテ



(a) スタックにnpをプッシュし srelの解析を開始



(b) スタックからnpをポップして npの解析を成功させる

図 6　トップダウン法による解析過程

ゴリ(たとえばsrel)の解析時に限定できる.

(2)　スタックから痕跡となるカテゴリをポップするタイミングが明確である. すなわち, スタックにプッシュされた文法カテゴリは, そのカテゴリの解析時以外にポップされることはない.

以上の点から, 痕跡をサーチするという点からみると, トップダウン法では, 痕跡の存在個所を比較的正確に予測できるため, 純粋なボトムアップ法にくらべ効率が良い. しかし, 1章で述べたように, トップダウン法には, 左再帰規則をそのまま扱えないという大きな問題点がある.

3.2　ボトムアップ法と左外置を考慮した構文解析

左外置を考慮した構文解析を, スタックをもちいてボトムアップ法で行う場合には, 従来, 次のような問題があるとされていた.

(1)　痕跡を支配するカテゴリをスタックにプッシュするタイミングが明確でない.

(2)　スタックからカテゴリをポップするタイミングが明確でない.

まず(1)についてであるが, 純粋なボトムアップ法では, 次に解析するカテゴリの予測を行わないため, パーザには, 次に痕跡を持つカテゴリが来るかどうかが予測できない. したがって, ボトムアップ法でカテゴリをスタックにプッシュしたい場合, (名詞は先行詞になりうるので)名詞が見つかった直後に行うことになる.

次に(2)についてであるが, ボトムアップ法の解析とは, 入力文の単語の品詞から上向きに句構造を構築していく解析であるが, 痕跡は, 入力文中に出現しない. したがって, 1つの痕跡の存在が仮定された場合に, 入力文の単語と単語の間のほとんど全てにその痕跡が存在すると仮定して解析を行う必要があるため, 痕跡の存在を仮定する個所がトップダウン法と比べ大幅に増加し, 効率が悪くなると考えられる. この問題に対する解法を3.3で述べる.

3.3　BUP-XG による左外置を考慮した構文解析

ボトムアップ構文解析システムBUPは, 以下で述べるように, ボトムアップ法とトップダウン法を融合した構文解析システムである. 本研究では, 上で述べたボトムアップ法の問題点を, BUPの基本動作を利用することで解決している.

3.3.1　BUP の基本動作

BUPシステムでは, 図7に示すように, ユーザがDCGで記述した文法規則と辞書は, BUPトランスレータ[4]によって, BUP節と呼ばれるものとlink節, 停止条件節とに変換され, それらが, 図8に示すgoal節などとともにボトムアップに構文解析を行うプログラムとして直接動作する.

実行に際しては, まず入力文の最初の単語の辞書引きを行い, その単語のカテゴリをヘッドとするBUP節を選択し解析を進める. たとえば, 入力文"john walks"を解析する場合, 辞書引きするとjohnのカテゴリがnpなので, BUP節(g1)が選択され, そのボディでは, 次にカテゴリvpの解析を試みる(goal(vp,[])). このことは, vpが後続語に対するカテゴリのトップダウン予測

```
s --> np, vp.                          (G1)
np --> [john].                         (D1)
vp --> [walks].                        (D2)
```

↓変換

```
np(G, [], I) --> {link(s, G)},
        goal(vp, []),
        s(G, [], I).                   (g1)
dict(np, []) --> [john].               (d1)
dict(vp, []) --> [walks].              (d2)
link(np, s).                           (l1)
link(X, X).                            (l2)
      ⋮
```

図 7 トランスレータによる変換例

になっていることを意味している．ここで，(g1)の link (s, G)は，カテゴリ s からゴール・カテゴリ G への到達可能性があるかどうかチェックするために使われる．到達可能性とは s を根とする部分木を上に成長させていく場合，将来 G を根とする部分木ができる可能性のことである．

このように，BUP は，入力文の単語の品詞やボトムアップ解析の結果出来上がったカテゴリをキーとしてそれをヘッドとする BUP 節を選択して解析を進める過程がボトムアップの動作，また，キーとなっているカテゴリの次に続くカテゴリをサブゴールとして予測して解析を行う過程がトップダウンの動作，となっている．

3.3.2 BUP-XG の解析メカニズム

(a) 痕跡となるカテゴリのスタックへのプッシュ

BUP-XG では，3.3.1 で述べた BUP のトップダウン予測の機能を利用することで，3.1 で述べたトップダウン法と同様のタイミングで，スタックに痕跡となるカテゴリをプッシュする．ただし BUP では，文法規則の右辺の第一カテゴリは，ボトムアップ解析のキーとなるため，トップダウンに予測されることがない．したがって，右辺の第一カテゴリがスラッシュ付きのカテゴリである規則は，スラッシュ・カテゴリに関する情報をスタックにプッシュできないため，BUP-XG では，このような規則を扱うことができない．

(b) 痕跡となるカテゴリのスタックからのポップ

BUP-XG では，スタックから痕跡となるカテゴリをポップするタイミングが 2 つある．第一は，BUP 節においてサブゴールとして予測されたカテゴリがスタックトップにある場合で，この場合には，スタックからカテゴリをポップして，そのカテゴリの解析を成功させる．これはトップダウン法の場合と同じである．

```
goal(G, A, X, Z):-
        dict(C, A1, X, Y),
        link(C, G)
        P=..[C, G, A1, A, Y, Z]
        call(P).
```

図 8 goal 節の定義

第二は，スタックトップにあるカテゴリから，現在のゴールへの到達可能性がある場合で，この場合には，スタックからカテゴリをポップし，そのカテゴリをキーとして BUP 節を選択し解析を続ける．純粋なボトムアップ法では，3.2 で述べたように，スタック上にカテゴリがある場合には，やみくもにそれをポップして，そのカテゴリから部分木を成長させる形で解析を進める．それに対し，BUP-XG の場合，スタック上のカテゴリからゴール・カテゴリへの到達可能性がある場合にのみポップするので，痕跡が存在する個所の予測が精密化し，無駄な解析を行う機会は大幅に減少する．

以上のスタックのチェックおよび操作は，従来の BUP における goal 節を拡張することで行われる．これについては，4.3 で述べる．

(c) BUP-XG による解析例

名詞句(7)の解析を例にとり，BUP-XG による解析の過程を簡単に説明する．文法規則は図 3 に示した規則を使用する．(ただし，スタックは実現されているとし，上で述べた操作を行うために goal 節も拡張されているものとする．)

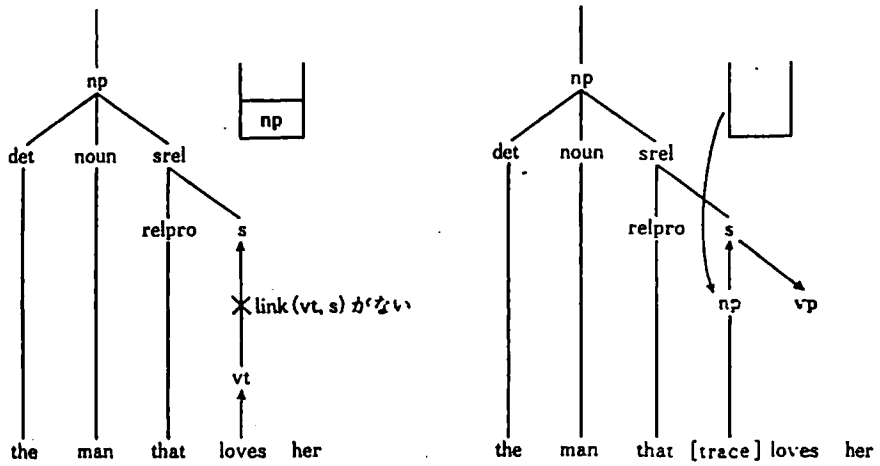
the man that loves her (7)

(1) 解析は goal(s, X, [the, man, that, loves, her], [])をコールすることから始まる．辞書引きによって先頭の単語 the の品詞が det であることから，det をヘッドとする BUP 節が選択される．

(2) noun の解析が成功すると，srel../np の記述からスタックに np がプッシュされ，関係代名詞節の解析が始まる．

(3) 関係代名詞(relpro)の that が見つかると，続いて s の解析が始まる．

(4) 辞書引きにより，先頭のカテゴリが vt であることが分かるが，図 3 の文法規則の場合，vt から s



(a) vtからsへの到達可能性のチェック

(b) スタックからポップしたnpをキーとしてBUP節を選択

図 9 BUP-XG による解析過程

への到達可能性がないため，vt をキーとした解析をあきらめる(図9(a))．しかし，スタックトップのカテゴリ np からゴール s への到達可能性があることから，スタックから np をポップし，np をヘッドとする BUP 節が選択される(図9(b))．

(5) その後，"loves her" を vp として解析し，名詞句の解析が終了する．

4 インプリメント

4.1 スタック

4.1.1 スタックの実現方法

BUP-XG では，スタックの状態を，4.1.2 で述べる Xlist と呼ばれる構造体で表し，それを後続する述語に次々に受け渡して行くことで，スタックを実現している．たとえば，次の(8)の規則の場合，(9)のように変数を2つ付加した規則に変換して，Xlist の受け渡しを行う．(9)の各カテゴリに付けられた変数のうち左の Xlist は，そのカテゴリの解析を始める時点でのスタックを表し，右の Xlist は解析が終了した時点でのスタックを表す．なお，変数の付加は，4.2 で述べるトランスレータによって自動的に行われるため，文法記述者は(9)の形で文法を記述する必要はないことに注意されたい．

```
s --> np, vp, pp.                          (8)
        |
        v
s(X0, X3) --> np(X0, X1), vp(X1, X2),
              pp(X2, X3).                  (9)
```

4.1.2 Xlist の構造

Xlist は(10)に示す構造体である．

x(category, argument, xlist)　　(10)

category はスタックトップのカテゴリ，argument はその category の引数のリストである．最後の xlist は，現在スタックトップにあるカテゴリをプッシュする前のスタックを表す Xlist である．また，空スタックは，空リスト([])で表される．

スタックへのプッシュは，プッシュしたいカテゴリと現在のスタックを表す xlist から，新しい Xlist(10)を作ることである．たとえば，x(np, [ARG], [])は，空スタックに np(ARG) をプッシュした状態のスタックを表している．

4.2 BUP-XG トランスレータ

4.2.1 BUP-XG トランスレータによる Xlist 用変数の付加

図10は，XGS で記述した文法規則を BUP-XG トランスレータで変換した例である．BUP-XG トランスレータは，従来の BUP トランスレータによって変換された BUP 節の全ての述語に，Xlist 用の変数を2つ付け加える．(ただし，{ }で囲まれた Prolog プログラムの部分の述語は除く．また，変数を付加するにあたって，goal の述語名を goal_x とし，従来の述語 goal と区別する．) 図10(11)の DCG 規則を変換した(12)において，X で始まる変数が Xlist 用の変数で，一番左の Xlist は，

```
s(S_A) --> np(NP_A), vp(VP_A), pp(PP_A),
        {agreement(NP_A, VP_A)}.                    (11)
                    ↓
np(G, [NP_A], I, X0, X1, XR) --> {link(s, G)},
        goal_x(vp, [VP_A], X1, X2),
        goal_x(pp, [PP_A], X2, X3),
        {agreement(NP_A, VP_A)},
        s(G, [S_A], I, X0, X3, XR).                 (12)
```

図 10 BUP-XG トランスレータによる変換例(1)

```
np --> det, noun, srel../np.
                ↓
det(G, [ ], I, X0, X1, XR) --> {link(np, G)},
        goal_x(noun, [ ], X1, X2),
        goal_x(srel, [ ], x(np, [ ], X2), X3),
        {depth_check(X2, X3)},
        np(G, [ ], I, X0, X3, XR).                  (13)
```

図 11 BUP-XG トランスレータによる変換例(2)

```
a --> b, c, <d>, e.
          ↓
b(G, [ ], I, X0, X1, XR) --> {link(a, G)},
        goal_x(c, [ ], X1, X2),
        goal_x(d, [ ], [ ], [ ]),
        goal_x(e, [ ], X2, X3),
        a(G, [ ], I, X0, X3, XR).                   (14)
```

図 12 BUP-XG トランスレータによる変換例(3)

解析を始める時点でのスタックを表し，二番目の Xlist は，その解析が終了した時点でのスタックを表している．

4.2.2 痕跡となるカテゴリのスタックへのプッシュ

図 11 は，スラッシュ・カテゴリを含む文法規則の変換例である．変換後の BUP 節において下線部が，スタックにカテゴリ np をプッシュする操作を表している．すなわち，noun の解析が終了した時点のスタック X2 と，プッシュされるカテゴリ np とから，np がプッシュされた状態を表す Xlist (下線部) を作り，これを srel の解析を起動する述語 goal_x に渡すことでプッシュが行われる．

補強項として加えられている述語 depth_check は，srel の解析前後のスタックの深さを比べ，プッシュした np が srel の解析中に使用されたかどうかをチェックするためのものである．なお，スタックは Xlist の受け渡しで実現されているので，srel の解析に完全に失敗した場合には，スタックにプッシュしたものは，バックトラックによって自動的にスタックから取り除かれる．

4.2.3 オープン，クローズのある規則の変換

図 12 はオープン，クローズのある規則の変換例である．2 章で示したように，文法規則の非終端記号がオープンとクローズで囲まれている場合には，その非終端記号で表されるカテゴリの中にある痕跡と，オープン，クローズの外にあるスラッシュ・カテゴリとの対応付けが禁止される．BUP-XG では，痕跡のサーチをスタックを用いて行っているため，上記の制約を満足するためには，オープンとクローズで囲まれたカテゴリを解析する際に，スタックを一時的に空にして解析を行えばよい．そのため，図 12 の (14) に示したように，変換後の BUP 節のボディ中の goal_x(d, ...) を下線部のようにする．そして，カテゴリ d の解析終了後に，スタックをカテゴリ c の解析終了時の状態に戻している (goal_x(c, ...) と goal_x(e, ...) の X2)．

4.3 goal 節の拡張

図 13 は，3.3 で述べたスタック操作 (ポップ) を行うため，今回拡張した goal_x 節の定義の一部である．(15) は，ゴール・カテゴリとスタックトップのカテゴリが等しい場合に，スタックからカテゴリをポップし，ゴール・カテゴリの解析を成功させる．また，(16) は，スタックトップのカテゴリ C からゴール・カテゴリ G への到達可能性を調べ (link(C, G))，可能性があれば，C をキーとして BUP 節を選択し，解析を進めるための節である．(図 10〜12 では，goal_x の引数の数が 4 つであるが，図 13 においては 6 つになっている．これは DCG のルールを Prolog に読み込む際に，システムが入力文に対する差分リスト用の変数をさらに 2 つ付加するためである [8].)

```
goal_x(G, GARG, x(G, GARG, X0), X0, S, S):-
    assertz(wf_goal(G, GARG, x(G, GARG, X0),
      X0, S, S)), !.                                (15)

goal_x(G, A, X0, X, S0, S):-
    X0=x(C, CARG, X1),
    C\==G,
    link(C, G),
    P=..[C, G, CARG, A, X0, X1, X, S0, S],
    call(P),
    assertz(wf_goal(G, A, X0, X, S0, S)).           (16)
```

図 13 goal_x 節の定義 (スタックのポップ部分のみ)

5 BUP-XG システムの評価

5.1 英語文法の記述

5.1.1 XGS による英語文法記述

英語において痕跡を持つ句構造には，これまで出てきた関係代名詞節のほかに，Wh 疑問文や受動態の文などがあり，これらの規則を，XGS では(17)，(18)のように記述できる．また，Yes-no 疑問文のうち be 動詞，法助動詞，have(完了)で始まる文は，それらが宣言文における通常の位置から文頭に移動してできるものと考えると，左外置の場合と全く同じメカニズムで処理できるため，(19)，(20)のように記述できる．

swhq —〉whnp, sq. . /obj.　　(17)

sdec —〉subj, bep, vp. . /obj.　　(18)

sq —〉bep, sdec. . /bep.　　(19)

sq —〉modalp, sdec. . /modalp.　　(20)

スタックを用いて左外置を考慮した構文解析を行う場合，痕跡を持つ句構造の等位構造が問題になる．たとえば(21)のように，関係代名詞節の埋め込み文が等位接続詞によって連結されている場合，規則(22)では解析できない．(23)は，(22)の変換結果であるが，これを用いて解析すると，関係代名詞節の解析を始める際にスタックにプッシュされたカテゴリ np は，最初の埋め込み文 "I love" の解析で，動詞 "love" の目的語としてスタックからポップされ，スタックは空となるため，二番目の埋め込み文 "you dislike" の解析で，痕跡が見つからず失敗する．

She is the girl that I love but you dislike.　(21)

srel —〉relpro, sdec, conj, sdec.　　(22)

```
relpro(G, [], I, X0, X1, XR) --> {link(srel, G)},
        goal_x(sdec, [], X1, X2),
        goal_x(conj, [], X2, X3),
        goal_x(sdec, [], X3, X4),
        srel(G, [], I, X0, X4, XR).      (23)
```

この問題を解決するには，二番目の埋め込み文の解析を始める前に，スタックを，最初の埋め込み文を解析する直前の状態に復元する必要がある．これは，(24)の下線部で示すように，二番目の文の解析に使用される Xlist を X1, X2 とすればよい．

```
relpro(G, [], I, X0, X1, XR) —〉{link(srel, G)},
        goal_x(sdec, [], X1, X2),
        goal_x(conj, [], X2, X3),
        goal_x(sdec, [], X1, X2),
        srel(G, [], I, X0, X3, XR).      (24)
```

現在，(24)に示すような Xlist 用の変数を割り当てるために，特別な記法(25)を用意して，変数を文法規則中に直接記述する方法をとっているが，(26)に示す形で記述できるよう，記法を拡張する予定である．

srel[X0, X3]==〉relpro[X0, X1], sdec[X1, X2],
conj[X2, X3], sdec[X1, X2].　　(25)

srel —〉relpro, sdec*, conj, sdec*.　　(26)

5.1.2 DCG で記述した英語文法との比較

表1は，トレース・サーチ形式(XGS)と非トレース・サーチ形式(DCG)で，ほぼ同程度の範囲をカバーする文法を記述した場合の，文法規則の数を比較したものである．表より分かるとおり，トレース・サーチ形式で記述すると，非トレース・サーチ形式による文法規則数と比較して，3割程度減少した．その中でも Yes-no 疑問文に関する規則の減少が著しいことが分かる．

表 1 文法規則数の比較(抜粋)

項　目	非トレース・サーチ	XGS	差
1) 動詞句に関する規則	54	46	8
2) 関係節	15	7	8
3) Yes-no 疑問文	72	7	65
文法規則総数	383	268	115

5.2 構文解析の実験と検討

5.2.1 構文解析結果の例

図14は，左外置を含む文の BUP-XG による構文解析結果の一例である．解析木の中で，丸で囲んだ "t" が痕跡を表している．この解析木からも分かるように，埋め込み文の句構造は，完全な宣言文と同じになっている．

5.2.2 従来の BUP との解析時間の比較

表2は，図15に示す例文についての解析結果をまとめたものである．トレース・サーチ形式で記述した文法規則を用いて構文解析を行う場合，システムが実行時に痕跡の存在個所をサーチするため，非トレース・サーチ形式で記述した文法規則を用いた解析よりも解析時間を要することが予想されていた．ところが，痕跡のない文(1, 2)でも，痕跡のある文(関係代名詞節(3, 4)を含む文)

```
This is the man that she loves.

Used 500  msec  for dictionary

521  msec.
No. 1
 |-sentence
    |-sdec
       |-sdec0
          |-subj
          |   |-np
          |       |-ddet
          |       |   |-det -- this
          |-aux
          |   |-bep
          |       |-be -- is
          |-pred
             |-np
                |-ddet
                |   |-det -- the
                |-nomhd
                |   |-n -- man
                |-ncomp
                   |-srel
                      |-relpro -- that
                      |-sdec0
                         |-subj
                         |   |-np
                         |       |-pron -- she
                         |-vp
                            |-v
                            |   |-v -- love
                            |   |-suffix -- s
                            |-obj
                                |-np -- (t)

Total Time = 792  msec.

number of   wf_goal was :     10.
number of   wf_dict was :     8.
number of fail_goal was :     43.
```

図 14 構文解析例

の場合でも，ほぼ同じ時間で解析結果が得られている．これは，BUP-XG が使用している文法規則の数が少ないことによるものと思われる．また，Wh 疑問文(5, 6)や受動態(7)，Yes-no 疑問文(8)の場合には，予想どおり，BUP-XG の方が時間がかかるが，2 倍以内に抑えられている．

1. I saw many more books than she did.
2. I take my dictionary to her.
3. This is a girl that I love.
4. The book that is on the table is very good.
5. What do you want to eat ?
6. Which ones did he give to her ?
7. She was given some books by her uncle.
8. Can you see that picture ?

図 15 例 文

6 おわりに

名詞句などが左外置され残された痕跡を，システムが自動的にサーチし発見することを前提として文法を記述するトレース・サーチ形式の文法記述法について検討し，DCG を拡張した新しい文法記述形式 XGS を提案した．そして，この形式で英語の文法を作成し，非トレース・サーチ形式(DCG)で記述した文法と比較して，文法規則数を 3 割程度減らすことが可能となり，その有効性を確認した．また，左外置を考慮した構文解析をボトムアップ法で行うには，効率の面で問題があると考えられていたが，トップダウン予測が組み込まれたボトムアップ構文解析システム BUP を拡張することで，それほど効率を落とさず解析できることを示した．今後さらに単語数の多い文についても実験を行いたいと考えている．

参考文献

[1] Gazdar, G., Klein, E., Pullum, G. K. and Sag, I. A.: *Generalized Phrase Structure Grammar*, Oxford, Basil Blackwell, 1985.

[2] 今野聡，田中穂積：ボトムアップ構文解析における左外

表 2 解析結果の比較

	BUP による解析				BUP-XG による解析			
文	木の数	解析時間 (sec)	成功 goal	失敗 goal	木の数	解析時間 (sec)	成功 goal	失敗 goal
1	1	2.64(1.13)	29	71	1	2.62(0.82)	39	82
2	1	0.84(0.31)	10	34	1	0.76(0.30)	11	33
3	1	0.77(0.56)	11	35	1	0.77(0.49)	10	41
4	1	0.82(0.60)	16	37	1	0.97(0.67)	20	37
5	1	1.35(0.32)	16	29	2	2.27(0.70)	37	41
6	1	1.18(1.10)	15	43	1	2.07(1.45)	30	52
7	2	2.40(2.00)	34	53	4	3.98(2.25)	67	63
8	1	0.94(0.71)	9	34	1	1.36(0.91)	20	37

(括弧内の時間は最初の構文解析木が得られるまでの時間である．)

置の処理，日本ソフトウェア科学会第1回大会論文集，1984, pp. 223-226.
[3] 今野聡：左外置を考慮したボトムアップ構文解析に関する研究，東京工業大学大学院修士論文，1985.
[4] 松本裕治，清野正樹，田中穂積：BUPトランスレータ，電総研彙報，Vol. 47, No. 8(1983).
[5] 松本裕治，清野正樹，田中穂積：BUPの高速化，情報処理学会自然言語研究会，39-7，1983.
[6] Matsumoto, Y., Tanaka, H., Hirakawa, H., Miyoshi, H. and Yasukawa, H.: *BUP: A Bottom-Up Parser Embeded in Prolog*, New Generation Computing, 1, 2, 1983.
[7] 大塚高信ほか：新英語学辞典，研究社.
[8] Pereira, F. and Warren, D.: Definite Clause Grammar for Language Analysis——A Survey of the Formalism and a Comparison with Augmented Transition Networks, *Artif. Intell.*, Vol. 13(May 1980), pp. 231-278.
[9] Pereira, F.: Extraposition Grammar, *AJCL*, Vol. 7, No. 4(1981).
[10] Pereira, F.: Logic for Natural Language Analysis, *Technical Note 275*, SRI International, 1983.
[11] 田中穂積，高倉伸，今野聡：ボトムアップ構文解析システムBUP上での英語文法開発とBUPの評価，*Proc. of Logic Programming Conf. '84*, 12-2, 1984.
[12] 田中穂積，今野聡，高倉伸：ボトムアップ構文解析システムBUPでの左外置変形の処理，情報処理学会自然言語処理研究会，44-1，1984.
[13] Winograd, T.: *Language as a Cognitive Process, Vol. 1: Syntax*, Addision-Wesley, 1983.