BUP 系解析システム上でのトップダウンな情報の制御について†

奥村 学†† 田中穂積††

純粋にボトムアップなパージングを行うシステムでは，構文解析木を下から上に向けて生長させるにつれて，情報を下から上へと自然に送ることができるが，その解析メカニズムによる制約で，上から下へは情報を送ることができない．そのため，解析途中での構文的，意味的チェックも，情報が下から上がってきて初めて実行されることになり，上から下に情報を流すことが可能な場合に比べると，実行を行うタイミングが遅れ，不要な計算をすることがある．既存のボトムアップ解析システムの中には，本論文で述べるように，文を読み進む過程で，部分的な解析結果を計算し，それらをダイナミックに双方向に流せるものは存在しなかった．本研究ではまず，BUP はボトムアップベースの解析システムであるが，純粋なボトムアップ法ではなく，goal 節を解析に用いていることで，上から下へ情報を流すことが可能であることを明らかにする．また，この特徴を生かし，ボトムアップベースの自然言語解析システムでは困難であると思われていた，上から下へ情報を流す機構 BUP-UTI を開発した．この BUP-UTI を用いることにより，ボトムアップ解析システム上で，解析途中に上下双方向へダイナミックに情報を流すことが可能になり，従来のボトムアップ解析システムに比べ，早期に非文法的な箇所や意味的な異常さを検出できるため，効率の良い自然言語解析が可能になることを示す．

1. はじめに

近年，Prologを用いた自然言語解析に関する研究が活発である．Prolog を自然言語解析に用いることの利点の１つは，DCG (Definite Clause Grammar)[9] とよばれる文法記述形式で記述した文法が，Prolog プログラムに変換され，そのままパーザとして動作させることが可能な点にある．我々は，作成する文法中に左再帰的な文法規則が含まれている場合のことを考慮して，DCG で記述した文法を変換して，ボトムアップにパージングを行う Prolog プログラムを生成するシステム BUP[5] を用いてきた．その後 BUP は，いくつかの改良がなされ[2),3),6)]，現在は自然言語解析システム Lang LAB[13] の中核になっている．

純粋にボトムアップなパージングを行うシステムでは，解析木を下から上に向けて生長させるにつれて，情報を下から上へと自然に送ることができるが，その解析メカニズムによる制約で，上から下へは情報を送ることができない．そのため，解析途中での構文的，意味的チェックも，情報が下から上がってきて初めて実行されることになり，上から下に情報を流すことが可能な場合に比べると，実行を行うタイミングが遅れ，不要な計算をすることがある．

また，３章で述べるように，自然言語解析システムにおいては，下から上，上から下の双方向に情報を流せることが望ましい．文を左から右に読み進みながら理解している人間のモデルを考えるなら，読み終った（理解した）部分の情報を左から右に（解析上は上から下に）流しながら文を解析していき，文全体を読み終った時点で，理解した結果が解析木の最上部に得られるシステムが自然であると思われるからである．

既存のボトムアップ解析システムの中には，下から上への情報を用いて一度パージングを終了し，その後，得られた（完全な）解析木をたどる過程で上から下に情報を流すものは存在する[1),11)]．しかし，本論文で述べるように，文を読み進む過程で，部分的な解析結果を計算し，それらをダイナミックに双方向に流せるものは存在しなかった．

BUP は，ボトムアップベースの解析システムではあるが，詳細に検討すると，純粋なボトムアップ法ではなく，２章で述べる goal 節を介して，トップダウンな予測を行っており，また，上から下へ情報を流すことが可能である．左外置処理のため BUP を拡張した BUP-XG[4] では，この特徴を生かし，goal 節を拡張するなどして，左外置処理に必要なスタックを実現している．

本研究では，BUP-XG におけるスタックがトップダウンな情報として用いられていることに着目し，ボトムアップベースの自然言語解析システムでは困難であると思われていた，上から下へ情報を流す機構を開発した．これを BUP-UTI (BUP Using Top-down Information) とよぶ．本論文では，この BUP-UTI により，ボトムアップ解析システム上で，解析途中に上下双方向へダイナミックに情報を流すことが可能に

† An Implementation of Top-down Information Passing on BUP System by MANABU OKUMURA and HOZUMI TANAKA (Department of Computer Science, Faculty of Engineering, Tokyo Institute of Technology).

†† 東京工業大学工学部情報工学科

なることを示す．また，それを応用して，効率よく自然言語解析を行うことができることを示す．

本論文では，まず2章で，BUP-UTI の基礎をなす BUP-XG の概要を述べるとともに，BUP-UTI の基本原理を説明する．3章では，簡単な日本語解析システムを例にあげ，トップダウンな情報が具体的にどのように有効に用いられるか，また，解析中それらがどのように伝わっていくかを述べる．また，4章では，トップダウンな情報の流れを生かした解析の例を示し，BUP-UTI を用いることにより，従来のボトムアップ解析システムに比べ，早期に非文法的な箇所を検出でき，効率よく文の解析を行うことが可能なことを示す．

2. BUP-UTI の基本原理

BUP-XG は，左外置処理を BUP 上で実現したものである．このため，文法の記述形式にも拡張を施したもの (XGS) を用いている．本章では，トップダウンな情報の流れを実現する基礎となる，BUP-XG におけるスタックの実現法および，文法記述形式 XGS について簡単に述べるとともに，BUP-UTI の基本原理を説明する．なお，BUP-XG の詳細は文献 4) を参照していただきたい．

2.1 BUP-XG におけるスタックの実現法

Prolog 上で左外置処理をトップダウンに実行する方法として，XG (extraposition grammar)[10]がある．この方法では，左外置処理実現のため，スタックを用いている．ボトムアップ法をベースにする BUP-XG では，このスタックを実現するため，後述するように，goal 節を用いた BUP の動作特性を利用している．本節では，このスタックの実現法について簡単に説明する．

BUP-XG では，スタックの状態を構造体で表し，それを後続する文法カテゴリに次々受け渡していくことで，スタックを実現している．例えば，下の文法規則(1)に対して，(2)のように2変数を付加したものを考える．(2)の各文法カテゴリに付加された2変数(イタリック体)のうち，左の変数は，その文法カテゴリの解析を始める時点でのスタック（入力スタック）を表し，右の変数は，解析が終了した時点でのスタック（出力スタック）を表す．

```
s→np, vp.                                   (1)
s(X0, X2)→np(X0, X1), vp(X1, X2).           (2)
```

(2)はトランスレータにより次の(3)に変換され，実際の解析は(3)の BUP 節を用いて行う．(3)を用いた解析では，

```
np(G, [ ], I, X0, X1, XR)→{s(G)},
goal_x(vp, [ ], X1, X2),
s(G, [ ], I, X0, X2, XR).                   (3)
```

np の解析が成功すると，$X0, X1$ にスタックの内容が返される．そして，np の出力スタック $X1$ が，次の文法カテゴリ vp の入力スタックに渡され，vp の解析が成功すると，$X2$ に新しいスタックが返される．その後，np の入力スタック $X0$，vp の出力スタック $X2$ をそれぞれ s の入力スタック，出力スタックとして解析を続ける．

ここで注意したいのは，np の出力スタック $X1$ を vp の入力スタックに受け渡す点である．この vp に対する入力スタック $X1$ は，np を解析した結果得られた情報であり，goal 節を用いて vp を解析する時に，goal 節のボディで辞書引きされる文法カテゴリまで伝わる（**図1**参照）．これは，np から得られた情報（スタック）が，文法カテゴリ vp を経て，次に辞書引きされる文法カテゴリまで，トップダウンに流れることを意味している．このスタック（トップダウンな情報）の流れを実現するメカニズムが，BUP-UTI の基本原理となっている．次節では，この基本原理について詳しく述べる．

2.2 BUP-UTI の基本原理

前節では，BUP-XG におけるスタックの実現法について述べた．BUP-XG では，変数を介して次々とスタックの状態を受け渡している．その際，各文法カテゴリに対して入力，出力スタックが存在し，ある文法カテゴリの出力スタックは，文法規則中のその文法カテゴリの直後にある文法カテゴリの入力スタックになっている．そして，この入力スタックは，goal 節を用いて解析することにより，トップダウンな情報として，次に辞書引きされる文法カテゴリまで伝わる．

以上のような情報（スタック）の受け渡しに関する

```
goal(G, A, X, Z):-
    dict(C, A1, X, Y),
    Link=..[C, G], call(Link),
    P=..[C, G, A1, AA, [ ], [ ], [ ], Y, Z], call(P),
    A=AA.                                   (4)
goal_x(G, A, XI, XO, X, Z):-
    dict(C, A1, X, Y),
    Link=..[C, G], call(Link),
    P=..[C, G, A1, AA, XI, XI, XOO, Y, Z], call(P),
    A=AA, XO=XOO.                           (5)
```

図1 BUP-UTI における goal 節
Fig. 1 Goal clause in BUP-UTI.

考察から，スタックの実現に用いた2変数をトップダウンに情報を流すための変数（この変数を以後，TI (Top-down Information) 用変数とよぶ）として利用し，この変数で情報を左から右に（上から下に）流すことで，効率の良い解析システムの実現を可能にしたのが BUP-UTI である．

なお，BUP-UTI では，BUP-XG と同様に，文法記述形式として XGS（後述）を利用し，また，トランスレータには BUP-XG トランスレータ（後述）を用いている．

以下では，この BUP-UTI の基本原理について述べる．

BUP-UTI では，前節に示した(1)→(3)のように，ユーザが XGS で記述した文法は，BUP-XG トランスレータにより BUP 節等に変換される．そして，それらが図1の goal 節とともにボトムアップ構文解析システムとして直接動作する．ここで，述語 'goal' は，トップダウンな情報を用いないで解析を行う goal 節であり，一方，述語 'goal_x' は，トップダウンな情報を解析に用いる goal 節である．

解析は，次のような goal 節の呼び出しにより始まる．

?-goal (s, A, [she, smiled], [ ]).

このゴールの実行により，(4)のボディが実行される．(4)のボディでは，入力文の先頭の単語の辞書引きを行い（述語 'dict'），その単語の文法カテゴリをヘッドとする BUP 節を選択する．今の場合，先頭の単語 'she' の辞書引きの結果，その文法カテゴリである np をヘッドとする BUP 節(3)を選択する．そして，そのボディの実行において，文法カテゴリ vp を次のゴールとして解析を行う (goal_x (vp, [ ], *X*1, *X*2))．これは，入力文の残りの部分に対して，文法カテゴリ vp の部分が存在するはずだという予測をし，goal 節を通じて，その予測（文法カテゴリ名 vp）をトップダウンに流していることに相当する．さらに goal 節のボディで次の単語を辞書引きし，その文法カテゴリをヘッドとする BUP 節を選択することで解析が進む．このように BUP は，純粋にボトムアップに解析を行っているわけではなく，部分木の左隅から解析を始め，そこから得た情報を用いて，次に来るべき文法カテゴリについて予測をし，その予測をトップダウンに用いることによって解析を進めている．

したがって，この BUP の動作特性を生かし，従来の goal 節を修正し，TI 用変数を導入して（図1参照），情報をトップダウンに流すことを考える．2.1節で述べたように，ある文法カテゴリを解析して得られた情報が，次に辞書引きする文法カテゴリに（上から下へ）伝わるのは，図1の述語 'goal_x' において，直前の文法カテゴリから伝わった入力 TI 用変数 *XI* が，次の単語を辞書引きして呼び出される述語Cの入力TI用変数に受け渡されることによる（イタリック体の箇所参照）．**図2**に，文法規則 M→H, G. が適用され，Hを頂点とする部分木の解析が成功し，Gをゴールとして解析を行う際の情報の流れを示す．したがって，BUP-UTI におけるトップダウンな情報は，トップダウンという用語から通常想起するように，支配している文法カテゴリ（上）から，支配されている文法カテゴリ（下）へと段階的に流れるわけでなく，goal 節で呼び出される文法カテゴリ(G)から，辞書引きされる単語のレベル（Gを頂点とする部分木の左隅で一番下）の文法カテゴリ(C)にまで一気に伝わる．それから，次の goal 節の呼び出しまでは，部分木がボトムアップに成長するとともに下から上に伝わる．そして，次の goal 節が呼び出されると，再び部分木の左隅で一番下の文法カテゴリへと一気に流れることを繰り返す．

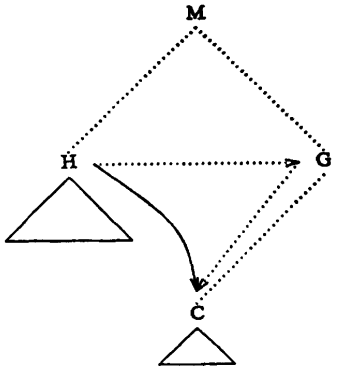

図 2 BUP-UTI における情報の流れ
Fig. 2 Flow of information in BUP-UTI.

既存のボトムアップ解析システムの中には，トップダウンな情報を解析に利用しているものもある．これらと BUP-UTI は，以下の点で異なる．

1) 拡張 LINGOL[1]や属性文法における属性計算[11]などのように，ボトムアップな情報だけを用いて一度パージングを終了した後，完成した解析木上を上から下に情報を流すものと異なり，BUP-UTI では，解析木を作成する過程でダイナミックに情報を上から下に流すことができる．
2) 属性文法の1つのクラス（LR 属性文法；LR-

attributed grammar) では，LR 構文解析の過程で双方向に情報（属性）を流し，属性計算をすることが可能である．しかし，このクラスでは，相続属性（上から下へ伝えられる属性）の使用には以下の条件が課されている．

構文解析過程のすべての状態において，適用可能なすべての文法規則に対して相続属性は唯1つに決定できること

この条件により，以下のような文法規則を含む文法はこのクラスには入らない．

A→X, B, X. with ↓B=↓A+1　　A→X, B, Y. with ↓B=↓A+2

（ここで，with 以後は属性計算式，↓B は B の相続属性を表す．）

なぜなら，Bの相続属性は，{↓B=↓A+1, ↓B=↓A+2} の2通りの計算式があり，唯1つに決定できないからである[11]．

この例から明らかなように，相続属性に対する条件は強力で，LR 属性文法では，限られた場合にしか上から下への情報（属性）を利用できない．これに対し，BUP-UTI では，上から下へ流す情報として任意のものを記述できる．

2.3 XGS および BUP-XG トランスレータ

BUP-XG における拡張された文法記述形式 XGS で記述した文法は，BUP-XG トランスレータにより BUP 節に変換される．その際，上で述べたトップダウンな情報を受け渡すための変数（TI 用変数）をトランスレータが自動的に付加する．したがって，通常ユーザはその変数について気にする必要はない．しかし，ユーザが直接変数を操作したいこともある．そのために，XGS では，文法規則の中に TI 用変数を陽に記述できるようになっている．図3は，XGS でのその記述例および変換結果である（この例は，文が 'and' で等位接続された場合には，その接続された2文に対するギャップは同じでなければならないことを記述するための BUP-XG の文法規則である[4]）．

図3からわかるように，XGS の記述における文法カテゴリの直後の[　]内の2つの変数が，TI 用変数として BUP 節にそのまま付加される．

```
s1[X0,X1] ==> s2[X0,X1],[and],s3[X0,X1].
                ↓変換
s2(G,[],I,X0,X1,XR) --> {s(G)},
        [and],
        goal_x(s3,[],X0,X1),
        s1(G,[],I,X0,X1,XR).
```

図3 BUP-XG トランスレータによる変換例（図中の文法カテゴリ s に対する添字は，理解しやすさのためにつけたもので，実際の解析には用いられない）

Fig. 3 An example of rule translation by BUP-XG translator.

3. BUP-UTI を用いた簡単な日本語解析システム

図4に示す XGS で記述した簡単な日本語文法を用いて，「花子は学校へ行った．」という文を解析してみよう．解析は文を左から右に読み進みながら行う．

図4の文法を用いると，最終的には図5の解析木を得るが，その解析過程で情報は概略矢印のように流れる．この情報の流れを利用し，この日本語解析システムでは，TI 用変数を後置詞句の意味情報のスタックとして用いて解析を行っている．すなわち，後置詞句

```
文(S) --> 後置詞句(_) , 文(S) .
文(S)[X0,X1] ==> 述語(Vp) ,
                 {interp((v,t),Vp,X0,S,X1)} .
後置詞句(_)[X0,X1] ==> 名詞(N) , 助詞(P) ,
                       {interp((n,v),P:N,X0,_,X1)} .
                ↓変換
後置詞句(G,_,I,X0,X1,XR) --> {文(G)},
        goal_x(文,S,X1,X2),
        文(G,S,I,X0,X2,XR).                    (g1)
述語(G,Vp,I,X0,X0,XR) --> {文(G)},
        {interp((v,t),Vp,X0,S,X1)},
        文(G,S,I,X0,X1,XR).                    (g2)
名詞(G,N,I,X0,X0,XR) --> {後置詞句(G)},
        goal(助詞,P),
        {interp((n,v),P:N,X0,_,X1)},
        後置詞句(G,_,I,X0,X1,XR).              (g3)
```

図4 簡単な日本語解析用文法

Fig. 4 Small Japanese grammar.

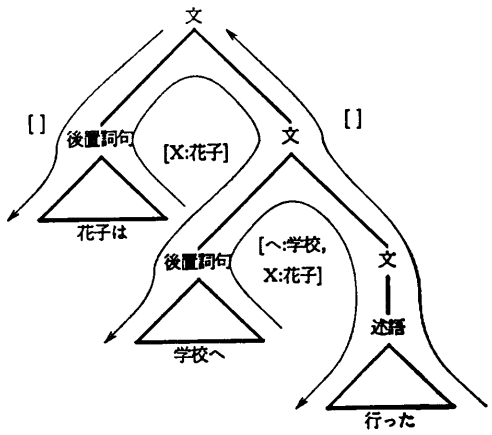

図5 BUP-UTI を用いた解析の流れ

Fig. 5 Analysis process in BUP-UTI.

を解析した意味解析結果を次々にスタックに付加して文末の方向（右方）に流し，述語（動詞）が現れた直後にそのスタックと動詞の意味を用いて全体の文の意味を決定している．また，TI 用変数を後置詞句の意味情報のスタックとして用いることで，日本語に対して一般に成り立つとされる「係り受け非交差の原則」を簡潔に実現することができる．この「係り受け非交差の原則」の実現法に関しては，文献 8) を参照してほしい．

以下，実際の BUP-UTI の実行過程を見ることで，具体的にスタック情報の流れを追い，トップダウンな情報の流れをどのように利用するかを説明する．解析は，

?-goal(文, A, [花子, は, 学校, へ, 行った], [ ]).

の呼び出しで始まる．

（1）goal 節(4)を用いて，最初の単語「花子」の辞書引きを行い，その文法カテゴリ「名詞」をヘッドにもつ文法規則を選択する．

（2）文法規則 (g 3) を適用する．そのボディにおいて，goal(助詞, P) を呼び出す．…これは「は」で成功する．

（3）意味解析プログラム interp を実行する．「花子は」の意味 (X：花子) を入力 TI 用変数 $X0$ (内容は [ ]) の先頭に付加し $X1$ に返す．

（4）後置詞句の解析が終了する．$X1$ には [X：花子] がバインドされている．

（5）文法規則 (g 1) を適用する．そのボディにおいて，goal_x(文, S, [X：花子], $X2$) を呼び出す．

（6）次の単語の辞書引きを行う．以下，(1)～(5)と同様の解析を繰り返し，その結果，goal_x(文, S', [へ：学校, X：花子], $X2'$) を呼び出す．

（7）次の単語「行った」の辞書引きを行う．そして，その文法カテゴリ「述語」をヘッドにもつ文法規則を選択する．この時点で，スタックの内容として，[へ：学校, X：花子] という情報が，「述語」をヘッドにもつ文法規則までトップダウンに伝わってきており，辞書引き直後に直ちに利用可能なことを注意してほしい．

（8）文法規則 (g 2) を適用する．

（9）意味解析プログラム interp を実行する．「行った」の意味 (Vp) と入力 TI 用変数 $X0$ ([へ：学校, X：花子]) を用いて意味解析を行う．そして，解析結果

(行った, (行為者：花子, 目標：学校, _), 1)

を S に返す ($X1$ には [ ] がバインドされる)．

(10)（トランスレータにより付加された）停止条件節を使って S が次々と上のレベルの「文」に上がり，最終的にトップレベルの「文」の呼び出し時の変数 A まで上がっていき，解析が終了する．そして，解析結果として，

A＝(行った, (行為者：花子, 目標：学校, _), 1)

を得る．

以上の例から明らかなように，BUP-UTI を用いると，「述語」として「行った」が得られた時点で，意味解析が終了し，全文に対する解析結果が得られることに注意してほしい ((9)参照)．BUP-UTI を用いる場合と，純粋なボトムアップ解析システムを用いる場合とで，解析過程が異なり，どのような利点が得られるかについては 4 章で述べる．

4. BUP-UTI を用いた構文的，意味的チェック

本章では，従来のボトムアップ解析システム BUP と，BUP-UTI における構文的，意味的チェックの実行法を比較することで，その実行のタイミングの違いを明らかにし，BUP-UTI の有効性を示す．なお，構文的チェックの例としては，主語と動詞の呼応のチェックを取り上げる．

4.1 構文的チェック

（i） 従来の BUP の場合

従来の BUP では，下に示すような文法規則を用いてチェックを行う．

```
s(S_A)→np(NP_A), vp(VP_A),
        {concord(NP_A, VP_A)}.        (6)
vp(V_A)→v(V_A),
        that_clause(THAT_A).          (7)
```

v（動詞）の各単語の辞書項目に記述されている人称・数に関する情報 (V_A) は，文法規則(7)を用いて述語 vp の引数まで伝わり，その結果(6)で，np の人称・数に関する情報 (NP_A) との間で呼応のチェックが述語 'concord' を用いて行われる．これは，呼応のチェックが，(6)の vp の解析終了まで遅延されることを意味する（図 6 参照）．

（ii） BUP-UTI の場合

BUP-UTI を用いると，次に示すような文法規則

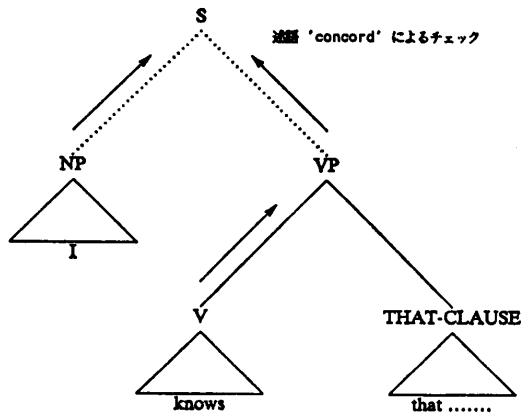

図 6 従来の BUP における情報の流れ
Fig. 6 Analysis process of concord phenomena in conventional bottom-up parser.

でチェックを行うことができる.

```
s(S_A)[X0, X1]-〉
        np(NP_A), vp(VP_A)[[NP_A|X0], X1].
                                        (8)
vp(V_A)[[NP_A|X0], X0]-〉
        v(V_A), {concord(NP_A, V_A)},
        that_clause(THAT_A).            (9)
```

BUP-UTI では, (8)の文法規則で, np の人称・数の情報 (NP_A) を TI 用変数に付加して vp に流し, それを用いて(9)の文法規則中で, v の人称・数の情報 (V_A) との間の呼応のチェックを直ちに行うことができる (図 7 参照).

この例からわかるように, 従来の BUP では, vp の解析が終るまで呼応のチェックが遅延されるのに対し, BUP-UTI の場合には, vp の解析中, v が現れた時点で, すなわち, チェックに必要な情報が得られたらすぐに呼応のチェックが行える. この違いは, vp が単純な構造の場合には, さほど問題にならないが, vp が複雑で大きな構造をもつ場合 (動詞の目的語として不定詞句, that 節など, 文に近いような構造のものがくる場合) には, この非文法的な箇所のチェックが遅延されることによる計算コストは大きくなる.

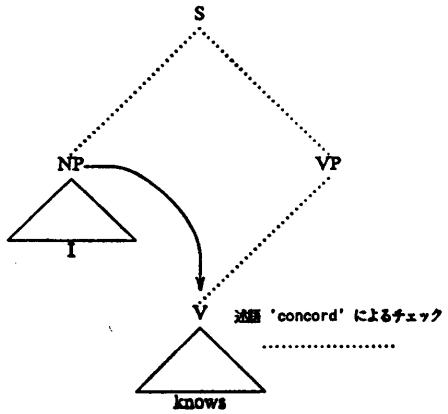

図 7 BUP-UTI における情報の流れ
Fig. 7 Analysis process of concord phenomena in BUP-UTI.

文 1, 2 を, 従来の BUP と BUP-UTI を用いて解析した時の, 解析に要した時間および辞書引きされた単語数を以下に示す.

1. I opens the door with a key.
2. I opens not only the door but also the window.

	BUP	BUP-UTI
1	450(8)	100(2)
2	600(8)	116(2)

(実験は, Quintus-Prolog インタプリタ上で, 文法規則数 13 で行った. 従来の BUP と BUP-UTI では, (6), (7)↔(8), (9)のように, s, vp に関する文法規則だけが異なる. 括弧の中の数字は辞書引きされた単語数, 解析時間の単位は msec である.)

この表から, 呼応のチェックの実行のタイミングの違いによる, 従来の BUP と BUP-UTI の解析に要する時間の差は明らかであろう. また, 従来の BUP では, 動詞句に対応する部分が長くなると, 解析時間が増加するのに対し, BUP-UTI ではほとんど変わらないことに注意してほしい. これは, 従来の BUP の場合には, 動詞句の最後まで解析を進めないと非文法的な箇所を検出できないのに対し, BUP-UTI の場合には, 主語の 'I' と動詞の 'opens' を辞書引きした時点で, 非文法的な箇所を検出し, それ以後の無駄な解析を行わないからである (辞書引きされた単語数参照).

この呼応のチェックと同様に, 日本語における用言 (動詞, 形容詞, 助動詞等) の相互承接関係のチェックや, 係り結び的な現象のチェックにも, BUP-UTI が利用可能であり, 我々はそれにより早期に非文法的な箇所の検出を行っている.

4.2 意味的チェック

(i) 従来の BUP の場合

従来の BUP では, 下のような文法規則を用いて意味解析を行う.

文 (S)→後置詞句 (Pp), 文 (S1),
{interp (Pp, S1, S)}. (10)

文 (Vp)→述語 (Vp). (11)

後置詞句 (P : N)→名詞 (N), 助詞 (P).

具体的には，(10)を再帰的に適用して次々と「後置詞句」の部分木を生成し，最終的に「述語」が現れた時点で(11)を適用する．この点に関しては，BUP-UTI を用いた意味解析（3章参照）と変わらないが，従来の BUP の場合，意味解析を実行するプログラム interp が文法規則(10)の最後にあるため，その実行のタイミングは，「後置詞句」の右側の述語「文」の意味 S1 が得られて初めて実行されることになる．したがって，「花子は学校へ行った」という文で，後置詞句「花子は」に対して意味解析を実行するのは，その右にある「文」の部分木が完成した後，すなわち，後置詞句「学校へ」と文「行った」から文「学校へ行った」の意味を解析した後になってしまう．以上まとめると，従来の BUP の場合には，全体の文に対する意味解析結果が得られるのは，図5における一番上の「文」に対する解析木（全体の文に対する解析木）が形成される時点まで延期される．これは，文献 7) で有用であると論じられている早期意味解析 (early semantic analysis) の考え方に反する．

(ii) BUP-UTI の場合

BUP-UTI を用いる意味解析の詳細については，3章で説明したが，要約すると，後置詞句を解析した(部分的な) 意味解析結果を次々にスタックに付加して文末の方向に流し，述語（動詞）が現れた直後にそのスタックと動詞の意味を用いて全体の文の意味を決定することができる．そして，「述語」として「行った」が得られた時点で，意味解析が終了し，全文に対する解析結果が得られる．この時点では，図5のような全体の文に対する解析木が完成しているわけではなく，完成している部分木は，そのうちの，2つの「後置詞句」の部分木および「述語」の部分木だけであることに注意してほしい．

この例から明らかなように，従来の BUP では，BUP-UTI と比べ，意味解析の実行のタイミングが遅れる．そのため，意味的に異常な箇所の検出が遅くなり，不要な計算を実行する可能性がある．例えば，3章の例文の「花子は」が「花子を」であるような文「花子を学校へ行った.」に対して，BUP-UTI の場合には，文末の「行った」の直後の意味解析で，「花子を」が「行った」に係りえないことから，意味的に異常であると判断することができるのに対し，従来の BUP を用いると，「花子を」が先頭の後置詞句であることから，全体の文に対する解析木が形成される時点で実行される意味解析プログラムにより初めて，「行った」に係りえないことに気付くことになる．

以上述べてきたように，BUP-UTI を用いれば，構文的，意味的チェックを，従来の BUP に比べ早期に実行できることから，不要な計算を回避することが可能である．

5. おわりに

BUP-UTI の基本原理，その実現法について述べ，実際の解析例を用いてその解析メカニズムを説明した．

BUP-UTI を用いることにより，ボトムアップ解析システム上で，解析途中に上下双方向へダイナミックに情報を流すことが可能になった．それにより，従来のボトムアップ解析システムに比べ，早期に非文法的な箇所や意味的な異常さを検出できるため，効率の良い自然言語解析が可能になることを示した．

3章で述べた日本語解析システムは，文献 7)の早期意味解析法にそったものであり，1章で述べた人間の文理解過程に近いと考えられる．

また，筆者らは，BUP-UTI を用いた日本語の並列句解析システムを開発しており，そこでも BUP-UTI を用いることの利点が述べられている[12]．

今後の課題としては，3，4章の例からわかるとおり，BUP-UTI における文法記述は XGS を用いていることもあり，やや書きにくい，したがって，高水準な文法記述言語を XGS の上に設計する必要がある．

謝辞 電子技術総合研究所の松本裕治氏，東芝総合研究所の木下聡氏，京都大学長尾研究室の佐藤理史氏，横浜国立大学の田村直良氏からは，大変有意義なご助言，コメントをいただきました．ここに，感謝の意を表します．

参考文献

1) 電子技術総合研究所 (編)： 拡張 LINGOL (1978).
2) 上脇 正，田中穂積，沼崎浩明：Lang LAB の構文処理アルゴリズムの高速化，第33回情報処理学会全国大会論文集，1N-8 (1986).
3) 今野 聡，奥村 学，田中穂積：ボトムアップ構文解析システム BUP の高速化，日本ソフトウェア科学会第1回大会論文集，3A-2 (1984).
4) 今野 聡，田中穂積：左外置を考慮したボトム

アップ構文解析システム，コンピュータソフトウエア，Vol. 3, No. 2, pp. 19-29 (1986).
5) Matsumoto, Y., Tanaka, H., Hirakawa, H., Miyoshi, H. and Yasukawa, H.: BUP: A Bottom-Up Parser Embedded in Prolog, *New Generation Computing*, Vol. 1, No. 2, pp. 145-158 (1983).
6) 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処理研究会，39-7 (1983).
7) Mellish, C.: *Computer Interpretation of Natural Language Description*, Ellis Horwood Limited, Chichester (1985).
8) 奥村 学，田村直良，徳永健伸，田中穂積：BUP 系解析システム上でのトップダウンな情報の制御について，情報処理学会自然言語処理研究会，59-5 (1987).
9) 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, pp. 231-278 (1980).
10) Pereira, F.: Extraposition Grammar, *AJCL*, Vol. 7, No. 4, pp. 243-256 (1981).
11) 田村直良，高倉 伸，片山卓也：自然言語処理を目的とした属性文法評価システム，コンピュータソフトウエア，Vol. 3, No. 3, pp. 266-279 (1986).
12) 田村直良，田中穂積：意味解析に基づく並列名詞句の構造解析，情報処理学会自然言語処理研究会，59-2 (1987).
13) 田中穂積，上脇 正，奥村 学，沼崎浩明：自然言語処理のためのソフトウエアシステム Lang LAB, *Proc. of the Logic Programming Conf. '86*, 3.1 (1986).

(昭和62年11月11日受付)
(昭和63年9月5日採録)

奥村 学（正会員）

1962年生．1984年東京工業大学工学部情報工学科卒業．1986年同大学院修士課程修了．現在，同大学院博士後期課程在学中．自然言語理解，知識表現に関する研究に従事．日本ソフトウェア科学会，認知科学会，人工知能学会各会員．

田中 穂積（正会員）

昭和39年東京工業大学理工学部制御工学科卒業．昭和41年同大学修士課程修了．同年電気試験所(現，電子技術総合研究所)入所．昭和58年東京工業大学工学部情報工学科助教授．昭和61年同大学教授となり現在に至る．工学博士．人工知能，自然言語処理の研究に従事．電子情報通信学会，認知科学会，日本ソフトウェア科学会，人工知能学会，計量国語学会各会員．