自然言語処理 89－6
（1992. 5. 14）

複数のLRテーブルを用いた一般化LRパーサによる部分列解析

新納 浩幸

松下電器産業株式会社 情報通信東京研究所

自然言語の中には，一般的な文法規則と均一の処理枠組だけでは解析するのが困難な文が多数存在する．それらの文に対しては変換処理を中心とした処理が有効である．そして，そのような変換処理には部分列解析が必要になる．本稿では，構文解析で利用する文法を併用することによって部分列解析を効率的に行なう手法を提案する．具体的には構文解析で利用するCFG規則から数種のLRテーブルを作成しておき，その中から適切なLRテーブルと構文解析で利用する一般化LRパーサを用いて部分列解析を行なう．これによって，部分列解析のための新たな規則やパーサを作成することなく，効率的に部分列解析が行なえる．

A parsing method using multiple LR tables for a part of sentence

Hiroyuki Shinnou
shinnou@trl.mei.co.jp

Matsushita Electric Industrial Co.,LTD.
Tokyo Infomation And Communications Research Laboratory
3-10-1, Higashimita, Tama-ku, Kawasaki 214, Japan

In natural languages, there are sentences which are difficult to parse by general grammar rules and an uniform process. For these sentences, transfer process is effective, in which parsing for a part of sentence palys an important role.

This paper presents a new parsing method for a part of sentence. In the method, a set of LR tables is made from a given CFG. By choosing an apropriate LR table out of that set, an effective parsing can be done for a part of sentence. This method is good in that it does not require developing grammar rules or parsers.

1 はじめに

自然言語文の解析では，一般的な文法規則と均一の処理枠組とを用いて，入力文の構文意味構造を求められることが理想である．しかし言語現象は多様であり，例外が多いことから，用意してある文法規則では対応できない文は多数存在する．それらすべての現象に文法規則の拡張によって対応してゆくと，文法規則の複雑さが増し，以後の文法の拡張性・保守性が損なわれるとともに，不必要な曖昧性の噴出から解析の効率低下や精度の悪化という面が生じ好ましくない．

1つの対処方法として，文法とは別の規則，構文解析とは別の処理を用意し，本来，核となる文法規則，構文解析を併用することによって文法では扱えない言語現象に対応してゆく方法がある．これは一般に変換処理と呼ばれているものである．

変換処理を主とした解析系の場合，共通して必要となる処理に部分列解析がある．

例として，駅員との対話を想定した以下の発話文の解析を考えてみる．

例文: 新宿まで大人1枚お願いします

このような文に対しても，なんらかの文法理論に基づいた文法規則と均一の処理枠組によって解析することは可能ではあろう．しかし，「〜まで大人1枚」や「〜お願いします」のような頻出パターンに対する意味表現を予め用意しておき，「〜」の部分列だけを解析し，その意味表現の空所を埋めてゆく変換処理の方が，文法規則の拡張性や保守性，また速度や精度の面で優れている [古瀬 90]．上記の例では，空所部分の解析として部分列解析が必要となったが，他にも，熟語や相関語句を構文解析の前処理として，それらの部分を1つの句としてまとめておく処理を行なう場合にも，部分列解析が必要となる．また，挿入句や未知語を含む文，非文などの処理も部分列解析が有効である [大場 89]．

このように変換を主とした解析では部分列解析が必要であるが，従来の部分列解析は，解析する部分列の構造をある程度単純なものに限定しているために，変換処理自体の頑健性が乏しい．しかも，頑健性を得るために，単純に部分列を解析するための規則数を増やすことも好ましくない．なぜなら，部分列解析には本質的に文を解析するのと同程度の能力が必要とされ，それら規則の作成や管理が困難になるからである．

本稿では，構文解析で利用する文法を併用することによって部分列解析を効率的に行なう手法を提案する．まず，構文解析の文法は拡張(補強)文脈自由文法(拡張 CFG)の形式で記述され，パーサとして一般化 LR パーサ [Tomita 87] を利用するという前提をおく．一般化 LR パーサによる解析では，予め，拡張 CFG の CFG 部分から LR テーブルと呼ばれる表を作成しておく．LR テーブルには解析時の動作が記述されており，一般化 LR パーサは LR テーブルの指示に従いながら解析を行なう．本手法の特徴は構文解析で利用する拡張 CFG から数種の LR テーブルを作成しておき，その中から適切な LR テーブルと構文解析で利用する一般化 LR パーサを用いて部分列解析を行なうことである．これによって，部分列解析のための新たな規則やパーサを作成することなく，部分列解析が行なえる．また，本手法による部分列解析の文法能力は，構文解析で利用する文法と同程度にすることも可能であり，より頑健な部分列解析が行なえる．また，各 LR テーブルは同一の文法から作成されたものなので，解析中に発火した文法規則，その引数となった部分木の番号，および発火結果の3つを保持しておけば，別フェーズの（部分列）解析で重複した処理が避けられ効率的である．

以下2節で部分列解析の問題点を示し，3節で本手法を説明し，4節でまとめを行なう．

2 部分列解析の問題点

本節では一般的な部分列解析を行なう際の2つの問題点を示す．

問題点1 文法能力

句には文を入れ子にした構造が取れるため，部分列の解析には本質的に文を解析する際の困難性がすべて継承されている．このため，部分列を解析するためには少なくとも文を解析できる以上の文法能力が必要になる．

再び，駅員との対話を想定した以下の発話文を考えてみる．

例文1: 新宿までいくらですか
例文2: 次の駅までいくらですか
例文3: 今度の急行が最後に止まる駅までいくらですか

解析系は，「〜までいくらですか」というパターンの意味表現を持っており，「〜」の部分を名詞句として解析する処理を仮定する．例文1のように「新宿」という単純な語句の場合には，文法は必要ないが，例文2の「次の駅」を名詞句として解析するためには何らかの文法が必要になる．更に，例文3の「今度の急行が最後に止まる駅」を名詞句として解

析するためには「今度の急行が最後に止まる」という”文”を解析できる文法が必要である.

問題点 2 重複処理

変換を主とした解析では，部分列を解析した際の処理が，後の処理で重複してしまうおそれがある．特に，部分列を解析した結果，指定した句にまとめられなかった場合，後の処理でその部分列を解析した処理の一部を再び行なってしまい効率が悪い．

再び，駅員との対話を想定した以下の発話文を考えてみる．

例文 4: 登戸で乗り換えて小田急線の向ヶ丘遊園までいくらですか

ここでも「〜までいくらですか」の「〜」の部分を名詞句として解析する処理を仮定する．例文 4 では「登戸で乗り換えて小田急線の向ヶ丘遊園」を名詞句として解析しようとするが名詞句としてはまとめられない．

ここからの解析戦略は幾つか考えられるが，先の部分列解析で「登戸で乗り換えて」を従属句と解析できた結果や，「小田急線の向ヶ丘遊園」を名詞句と解析できた結果を利用しない場合，先の部分列を解析した処理を重複しておこなう可能性が高く，効率上好ましくない．

3 複数の LR テーブルを利用した部分列解析

本節では部分列を解析するために本稿で提案する手法を説明する．

本手法の前提として，構文解析の文法は拡張 CFG の形式で記述され，パーサは一般化 LR パーサであるという前提をおく．一般化 LR パーサによる解析では解析動作が記述される LR テーブルを，与えられた CFG 規則を用いて予め作成しておく．解析時には LR テーブルに指示されている動作に従うことで効率的に解析を行なう．一般化 LR パーサの 1 つの特徴はパーサと文法が独立しているため，入力される形態素列を別の文法規則で解析したい場合に，参照している LR テーブルをその文法に対する LR テーブルに変更するだけでよい点である（図 1）．

本手法は上記の性質を利用し，構文解析で利用する CFG 規則から数種の LR テーブルを作成しておく．部分列の解析の際には，その部分列を解析するのに適した LR テーブルを参照し，構文解析で用いる一般化 LR パーサをそのまま用いて解析する（図 2）．これによって，部分列解析のための新たな規則やパーサを作成することなく，部分列解析が行なえる．

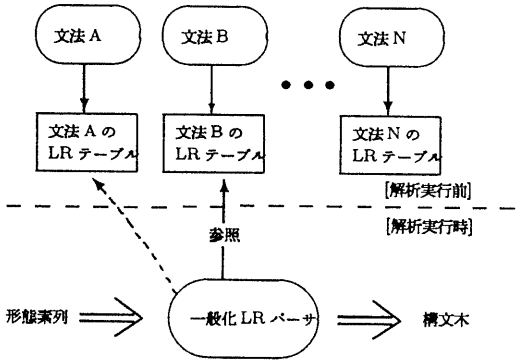

図 1: 文法を変更した解析

同一の CFG 規則から部分列を解析する LR テーブル作成するために，CFG 規則の開始記号や生成規則を変更した CFG 規則を作成する．CFG 規則の開始記号だけを変更して作った LR テーブルの場合，その文法能力は構文解析で利用する文法と同程度になり，より頑健な部分列解析が行なえる．

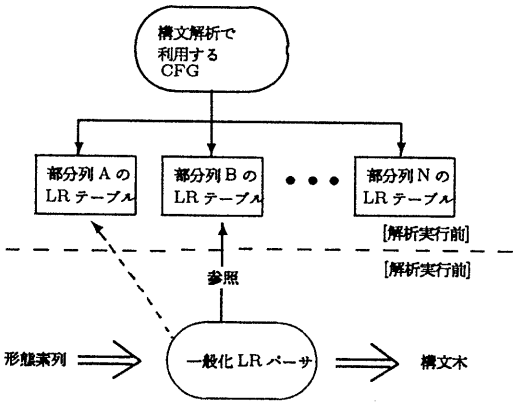

図 2: 複数の LR テーブルを利用した解析

以下に開始記号を変更した LR テーブルの作成とそれを用いた部分列解析方法，次に生成規則を変更した LR テーブルの作成とそれを用いた部分列解析方法，最後に解析の重複処理を避けるための解析経過の保持方法を示す．

```
文法1)  ((<文> <==> <連用句> <動詞>)                    extra_funct_1)
文法2)  ((<文> <==> <連用句> <文>)                      extra_funct_2)
文法3)  ((<文> <==> <従属句> <文>)                      extra_funct_3)
文法4)  ((<文> <==> <名詞句> <助動詞>)                  extra_funct_4)
文法5)  ((<従属句> <==> <文> <接続助詞>)                extra_funct_5)
文法6)  ((<連体修飾句> <==> <文> )                      extra_funct_6)
文法7)  ((<連用句> <==> <名詞句> <格助詞>)              extra_funct_7)
文法8)  ((<名詞句> <==> <名詞>)                         extra_funct_8)
文法9)  ((<名詞句> <==> <連体修飾句> <名詞>)            extra_funct_9)
```

図3: 簡単な日本語文法

状態番号	動詞	助動詞	接続助詞	格助詞	名詞	end_mark
0					s6	
1		s8		s7		accept
2	s11				s6	
3					s12	
4					s6	
5			s14		r6	
6		r8		r8		r8
7	r7				r7	
8			r4		r4	
9		s8		s7		
10			s14,r2		r2,r6	
11			r1		r1	
12		r9		r9		r9
13			s14,r3		r3,r6	
14					r5	

ACTION テーブル

状態番号	文	従属句	連体修飾句	連用句	名詞句
0	5	4	3	2	1
1					
2	10	4	3	2	9
3					
4	13	4	3	2	9
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					

GOTO テーブル

図4: LR(< 名詞句 >) テーブル

3.1 開始記号の変更よるLRテーブルの作成

CFG は非終端記号 N, 終端記号 T, 生成規則 P, 開始記号 S の4つの組 (N,T,P,S) からなる. LR テーブルはこの4つの組から作成される [Aho 77].

ここで注意する点として, 開始記号は, その文法により作成する構文木のルートノードに対応する. このため, 自然言語文の解析では文を表す非終端記号を開始記号に設定して, LR テーブルを作成する. しかし, 開始記号は任意の非終端記号で構わない. その場合, 作られた LR テーブルは設定した開始記号を作る動作が記述されたものになるだけである.

本稿では上記の性質を利用して, 部分列をある非終端記号 X にまとめる要求がある場合, その非終端記号 X を開始記号とした LR テーブル (以下これを LR(X) テーブルと略記する.) を予め作成しておく.

例として, 簡単な日本語文法を考える (図3). 各文法規則は CFG の書き換え規則とその拡張部の関数名からなっている. 例えば文法1の場合, < 文 > <==> < 連用句 > < 動詞 > が CFG の書き換え規則であり, extra_funct_1 がその拡張部の関数名となっている. 拡張部の関数はその CFG の書き換え規則が発火した際, 右辺の各カテゴリに対応する部分木を引数として, その書き換え規則の発火条件を調べ, 条件を満たした時, 左辺のカテゴリに対応する部分木を作成する. 図3の文法から LR(< 名詞句 >) テーブルを作成すると図4のようになる. また, 図3の文法から作られる LR(< 文 >) テーブルが構文解析で利用される LR テーブルである.

図4の LR(< 名詞句 >) テーブルを利用して, 例えば「小田急線に接続してる駅」を解析した場合, 図5のように名詞句に解析できる. 図5では各語を読み終った後のグラフ構造化スッタクの結果を示している [田中 89]. (0) は, 初期状態の状態番号0を表し, [[A],B] はグラフ構造化スッタクのノードであり, A は部分木の名前, B は状態番号を示す. また, k1,k2,…は形態素情報を示す部分木の名前であり, どのフェーズの部分列解析でも同じであることに注意する. 図5では状態番号0から " 小田急線 (名詞)" を読み, LR(< 名詞句 >) テーブルから s6

を見て，部分木[k1]と状態6がノードに加えられる．注記してあるのは発生したリジュース処理である．例えば，”接続している(動詞)”を読んだ結果では，文法7が起動され，部分木[0]と部分木[k1]から部分木[1]が作成されたことが分かる．

```
小田急線(名詞)
  ==> (0)-[[k1],6]
に(格助詞)
  ==> (0)-[[0],1]-[[k2],7]
      cf) r8 : [0] <== [k1]
接続している(動詞)
  ==> (0)-[[1],2]-[[k3],11]
      cf) r7 : [1] <== [0][k2]
駅(名詞)
  ==> (0)-[[3],3]-[[k4],6]
      cf) r1 : [2] <== [1][k3]
          r6 : [3] <== [2]
end_mark
  ==> (0)-[[4],1]  ---> accept
      cf) r9 : [4] <== [3][k4]
```

図5: LR(< 名詞句 >)テーブルによる解析

3.2 生成規則の変更よるLRテーブルの作成

ある種の部分列解析では，開始記号の他に生成規則も変更したLRテーブルを利用する方が，解析処理上有効である．ここでは開始記号の他に生成規則も一部変更したLRテーブルの作成方法とそれを用いた部分列解析を示す．

生成規則の変更方法としては，CFG規則の右辺にある（非）終端記号列を逆転させた生成規則を作る方法と生成規則の一部を取り除く方法を示す．

3.2.1 逆生成規則

部分列解析に失敗した後，解析系としてどのような処理を行なうかは，その部分列解析を何のために利用しているかに依存する．

これまでの例であげた変換処理の場合には，部分列解析は文法外の現象に対処する非文の処理の一部とも見れるため，部分列解析に失敗した後の処理を詳細に記述することは実質意味がなく，いわゆる粗い処理を行なうのが妥当である．一例として部分列解析に失敗した後，取り出したかった句の主要部を用いて以後の処理を続ける処理がある．そのような処理を日本語の部分列解析で行なう場合，日本語では句の主要部が後置される傾向があるため，部分列の最後尾から最大の長さをもつ目的の句を取り出すのが妥当である．

しかしこれを通常の左から右方向へ解析する方法で処理すれば，図6で示されるように，解析が失敗した地点から再び目的の句をつくり出す処理を繰り返さなくてはならず効率が悪い．

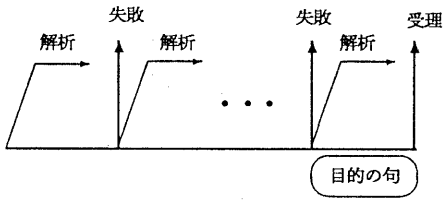

図6: 順方向への解析

しかも最後に得られた句が必ずしも最大の長さを持つ保証はない．簡単な例として以下の文法を考える．

```
S <==> a b c b
S <==> b c
S <==> c
```

図7: 順向きの解析では不都合な文法

図7の文法と一般化LRパーサを用いて”abcbc”を解析すると，2番目の”c”を読み込んだ時点で解析が失敗し，以下解析を続行させても，”c”という句しか出てこない．しかし最後尾からの最大幅の目的の句は”bc”である．

ここでは上記の問題点から，逆向きに解析するためのLRテーブルを作成し逆方向へ解析する手法を提案する．解析はまず逆向きに解析する．受理されたらそれが求める結果となる．また解析が途中で失敗した場合，最大の長さを持つ目的の句は，その失敗した地点から最後尾までの間から探せば良い．探索の手法はいくつか考えられるが，実質は以下の2点を併用させれば十分である．

- 逆向きの解析中には，最後尾から最大の長さをもつ目的の句を常に更新・保持しておく．
- 逆向きの解析が失敗した地点から順向きに解析する．

注意として順向きに解析する際は逆向きの解析との重複した処理は避けられる．これは次節で説明する．

逆向きに解析するためのLRテーブル（以下，逆LRテーブルと呼ぶ）は，文法のCFG規則の右辺にある（非）終端記号列を逆転させた生成規則$Rev(P)$を作り，$(N, T, Rev(P), S)$から作成する．ここでも開始記号は任意の非終端記号で構わない．拡張部

```
文法1)  ((<文> <==> <動詞> <連用句>)          extra_funct_1)
文法2)  ((<文> <==> <文> <連用句>)            extra_funct_2)
文法3)  ((<文> <==> <文> <従属句>)            extra_funct_3)
文法4)  ((<文> <==> <助動詞> <名詞句>)        extra_funct_4)
文法5)  ((<従属句> <==> <接続助詞> <文>)      extra_funct_5)
文法6)  ((<連体修飾句> <==> <文> )            extra_funct_6)
文法7)  ((<連用句> <==> <格助詞> <名詞句>)    extra_funct_7)
文法8)  ((<名詞句> <==> <名詞>)               extra_funct_8)
文法9)  ((<名詞句> <==> <名詞> <連体修飾句>) extra_funct_9)
```

図8: 逆向き日本語文法

状態番号	動詞	助動詞	接続助詞	格助詞	名詞	end_mark
0					s2	
1						accept
2	s6	s5	r8	r8		r8
3			r9	r9		r9
4			s10,r6	s9,r6		r6
5					s2	
6				s9		
7			r2	r2		r2
8			r3	r3		r3
9					s2	
10	s6	s5				
11			r4	r4		r4
12			r1	r1		r1
13			r7	r7		r7
14			s10,r5	s9,r5		r5

ACTION テーブル

状態番号	文	従属句	連体修飾句	連用句	名詞句
0					1
1					
2	4		3		
3					
4		8		7	
5					11
6				12	
7					
8					
9					13
10	14				
11					
12					
13					
14		8		7	

GOTO テーブル

図9: 逆 LR(< 名詞句 >) テーブル

の関数は呼びだし側で引数の並びを逆転させて利用する.

図3の文法を逆向きにした文法を図8, その文法の解析記号を < 名詞句 > として作られる逆 LR(< 名詞句 >) テーブルを図9に示す.

以下図9の逆 LR(< 名詞句 >) テーブルを用いた例文6の解析例を示す.

例文6: 彼女がどうしても知りたいって言うんでちょっと聞いてみたいのですけど小田急線に乗れる駅までいくらですか

ここでも「〜までいくらですか」の「〜」の部分を名詞句として解析する処理を仮定し, 「彼女がどうしても知りたいって言うんでちょっと聞いてみたいのですけど小田急線に乗れる駅」を名詞句として解析する場合を考える.

部分列解析の成功・失敗にかかわらず, 最後尾からの最大の長さを持つ名詞句 (この場合「小田急線に乗れる駅」) だけを取り出したいとする.

図9で示される逆 LR(< 名詞句 >) テーブルを用いて文末から逆に解析した場合, 図10のようになる.

```
駅 (名詞)
  ==> (0)-[[k1],2]
乗れる (動詞)
  ==> (0)-[[k1],2]-[[k2],6]
に (格助詞)
  ==> (0)-[[k1],2]-[[k2],6]-[[k3],9]
小田急線 (名詞)
  ==> (0)-[[k1],2]-[[k2],6]-[[k3],9]-[[k4],2]
けど (接続助詞)  ==>
  ==> (0)-[[k1],2]-[[2],4]-[[k5],10]
      cf) r8 [0] <== [k4]
          r7 [1] <== [k3] [0]
          r1 [2] <== [k2] [1]
          r6 [3] <== [2]
          r9 [4] <== [k1] [3] <保持する名詞句>
です (助動詞)
  ==> (0)-[[k1],2]-[[2],4]-[[k5],10]-[[k6],5]
の (格助詞)
  ==> fail
```

図10: 逆 LR(< 名詞句 >) テーブルによる解析

状態番号	動詞	助動詞	接続助詞	格助詞	名詞	end_mark
0					s5	
1		s7		s6		
2	s9				s5	
3					s10	
4					r6	accept
5		r8		r8		
6	r7				r7	
7					r4	r4
8					r2,r6	r2
9					r1	r1
10		r9		r9		

ACTION テーブル

状態番号	文	従属句	連体修飾句	連用句	名詞句
0	4		3	2	1
1					
2	8		3	2	1
3					
4					
5					
6					
7					
8					
9					
10					

GOTO テーブル

図 11: 部分 LR(< 文 >) テーブル

解析中は < 名詞句 > が得られた時に，その句の開始位置を調べ開始位置が文末（この場合スタート地点）である場合，現在保持している < 名詞句 > と交換する．上記の例の場合，”の”を読み込んだ段階で解析が失敗する．その時に最終的に保持している名詞句として図 10 の部分木 [4]（「小田急線に乗れる駅」の構造）が得られている．次に”です”から順向きに図 4 で示される LR(< 名詞句 >) テーブルを用いて解析する．テーブルの情報から名詞まで読み飛ばし”小田急線”を読み込む．この時点で保持している名詞句の方が解析する部分列よりも長くなるので，保持している名詞句を結果として返す．これは，最初から LR(< 名詞句 >) テーブルを用いて順方向に解析した場合と比べて，”の”よりも前方の解析を省いている分だけ効率的である．

3.2.2 部分生成規則

本稿で提案する部分列解析の 1 つの特徴として部分列解析の文法能力が構文解析時に利用する文法能力と同程度であるという点がある．

しかしある種の部分列解析では，それほど大きな文法能力は不必要だし，ある構造を排除した句を取り出したい場合も多い．

例えば，図 3 に示される文法を用いて，

何回〜ても〜できない

というパターンの最初の空所部分をカテゴリー < 文 > に部分列解析する場合を考える．このとき最初の空所部分に従属句のある文が来るとは考えられないので，文法 3 と文法 5 は不必要である．

このように不必要な文法があると，解析の効率や精度の面で好ましくない．

そこで本来の生成規則から不必要な規則を除いた生成規則 P_{sub} を作り，(N, T, P_{sub}, S) から LR テーブルを作成する．ここでも開始記号は任意の非終端記号で構わない．

例として，図 3 の文法から文法 3 と文法 5 を除いた文法を作り，その文法の解析記号を < 文 > として作られる部分 LR(< 文 >) テーブルを図 11 に示す．

図 11 に示される LR テーブルからは，「駅に着くと電車が来た」は解析できないが，「駅に着く」や「電車が来た」は解析できることが確かめられる．

3.3 重複解析の回避

本手法による各 LR テーブルは，同一の文法から作成したものなので，文法番号の統一がとられている．このため，部分列の解析の際に，発火した文法規則とその引数となった部分木の番号およびその結果（文法適用の失敗あるいは文法適用により作成された部分木の番号）を保持しておけば，別フェーズの（部分列）解析で重複した処理が避けられる．

例を示す．

例文 5: 子供だと向ヶ丘遊園までいくらですか

ここでも「〜までいくらですか」の「〜」の部分を名詞句として解析する処理を仮定し，「子供だと向ヶ丘遊園」を名詞句として解析する場合を考える．LR テーブルは図 4 の LR(< 名詞句 >) テーブルを利用する．図 12 に解析経過を示す．

解析経過は図 13 のように，解析中，リジュースが起こった時の文法番号，引数の部分木番号および文法適用後の部分木番号（適用失敗の場合は fail）を記録しておく．

部分列の解析に失敗した後，解析系としてどのような処理を行なうかは様々である．しかし，例えば「子供だと」の部分をカテゴリー < 従属句 > にまとめる処理が行なわれる場合には，必ず図 13 に示された経過を辿ることになり，文法発火時にこの表を参照すれば重複した処理が避けられる．

また，これは部分列解析の順序によらない．例えば，例文 5 の全体を < 文 > して解析する処理（こ

れはLR(＜文＞)テーブルによる解析）を先に行ない，その解析が失敗した後，「子供だと向ヶ丘遊園」を名詞句として解析する場合でも，先の解析で「子供だと」の部分をカテゴリー＜従属句＞にまとめる処理が行なわれていれば，その処理は重複して行なわれず，保持してある解析経過から直接，結果の部分木を得られる．

```
子供(名詞)
  ==> (0)-[[k1],6]
だ(助動詞)
  ==> (0)-[[0],1]-[[k2],8]
      cf) r8 : [0] <== [k1]
と(接続助詞)
  ==> (0)-[[1],5]-[[k3],14]
      cf) r4 : [1] <== [0][k2]
向ヶ丘遊園(名詞)
  ==> (0)-[[2],4]-[[k4],6]
      cf) r5 : [2] <== [1][k3]
end_mark
  ==> fail
      cf) r8 : [3] <== [k4]
```

図12: 部分列解析結果

文法番号	引数の部分木番号	発火結果
8	(k1)	0
4	(0,k2)	1
5	(1,k3)	2
8	(k4)	3

図13: 解析経過の保持

4 まとめ

本稿では，構文解析で利用する文法を併用することによって部分列解析を効率的に行なう手法を提案した．具体的には，構文解析として一般化LRパーサを利用することを前提にし，構文解析で利用するCFG規則から開始記号や生成規則を変更して複数のLRテーブルを作成しておき，適切なLRテーブルと構文解析で利用する一般化LRパーサとを用いて部分列を解析する手法である．これによって，部分列解析のための新たな規則やパーサを作成することなく，部分列解析が行なえる．また，本手法による部分列解析の文法能力は，構文解析で利用する文法と同程度にすることも可能であり，より頑健な部分列解析が行なえる．また，各LRテーブルは同一の文法から作成されたものなので，解析中に発火した文法規則，その引数となった部分木の番号，およびその結果の3つを保持しておけば，別フェーズの（部分列）解析で重複した処理が避けられ効率的である．

LRテーブルを複数持つことは，メモリの効率面では良くないが，以下の2点からこの問題には対処したい．

- LRテーブルは一種のスパース行列であるため，様々な圧縮法[青江 90]が利用できる．またLRテーブルの特質を利用した圧縮法も存在する[Aho 77].
- パーサでなんらかの工夫を行なえば，開始記号だけを変更して作った各LRテーブルは1つのLRテーブルで代表できると予想している．この点は今後の課題である．

今後は発話文解析の非文の処理に本手法を取り入れたい．発話文には，言い直し，間投詞，省略，助詞落ち，文節の倒置など通常の文法では対応するのが困難な現象がよく現れ，非文と判断されてしまう文が多い．非文の処理では，ある部分列から予想した句の情報を取り出し，判明している部分的な情報から，なんらかの意味をつくり出す必要がある．このため本稿で示した手法が有効に利用できることが期待できる．

また本稿で例とした変換処理を実現するために，入力文が用意しているパターンと同じ[1]であることを効率的に判断する方法も考えたい．

なお，本研究は財団法人新世代コンピュータ技術開発機構からの受託(発仕番号7302号)により行なわれたものである．

参考文献

[Aho 77] A.V.Aho, J.D.Ullman :『Principles of Compiler Design』, Addison-Wesley,1977. (邦訳『コンパイラ』, 土居範久訳, 培風館,1989).

[Tomita 87] M.Tomita : "An Efficient Augmented Contex-Free Parsing Algorithm", Computational Linguistics, 13,1-2,31-46,1987.

[青江 90] 青江順一 :「スパース行列の圧縮法」, bit, vol.22, No.6, 658-665, 1989.

[大場 89] 大場健司, 元吉文男, 井佐原均, 横山晶一, 石崎俊, 板橋秀一 :「未定義語を含む文の多段階構文解析法」, 情報処理学会自然言語処理研究会, NL70-4, 1989.

[田中 89] 田中穂積 :『自然言語解析の基礎』, 産業図書,1989.

[古瀬 90] 古瀬蔵, 隅田英一郎, 飯田仁 :「変換主導型機械翻訳の実現手法」, 情報処理学会自然言語処理研究会, NL80-8, 1989.

[1] ここでは文字列の一致による処理の他，「～までいくらですか」と「～までいくらなの」などを同じとする処理も考えている．