# 自然言語解析の新しい方法 − LR 表工学の提案 (1)

# A New Method of Natural Language Processing -LR Table Engineering (1)

田中穂積，李輝，徳永健伸
Hozumi Tanaka, Hui Li and Takenobu Tokunaga

東京工業大学情報理工学研究科
Tokyo Institute of Technology

**Abstract**

A new method of natural language analysis is proposed. The method is based on GLR parsing algorithm that can handle CFG. The paper discusses that CFG often suffers from describing local constraints such as constraints between adjacent words. The connection constraint is often expressed by a connection matrix. In case of Japanese morphological analysis, it is a usual way to use a connection matrix to check the connection between adjacent words in a sentence, and then use CFG to do syntactic parsing with conventional parsing algorithms, Chart and GLR. Consequently, the parsing is divided into two separate phases, morphological analysis and syntactic analysis. The paper gives a new method to merge two separate phases. According to our method we can incorporate the connection constraints into a LR table to generate more compact LR table. Using the compressed LR table indicates that GLR parsing makes use of two constraints, the conection matrix and CFG, at the same time. The advantages of our method and future problems are discussed.

## 1 まえがき

自然言語の統語解析用のアルゴリズムとして最近良く用いられるものに，チャート法 [Kay 80] と一般化 LR(GLR) 法がある．いずれも一般の文脈自由文法 (CFG) を扱うが，これらは音声認識にも応用されるようになってきた．GLR 法は，先読み語のもつ品詞 (前終端記号; preterminal) 情報を利用しながら解析を進めるもので，経験的にもっとも効率の良いアルゴリズムであるとされている [Tomita 86][Aho 86][Shippu 90][Kipps 91][Shann 91][Tomita 91].

本稿では，GLR 法のベースになる LR 表を修正することで，自然言語の解析過程の柔軟な制御が可能になることを説明する．それにより，従来のやり方では不可能であったことが可能になったり，可能であるにしても煩雑で困難であったことが，容易に実現できることがある．このように，LR 表を修正して統語解析過程を制御する手法を「LR 表工学」とよぶ．このとき，GLR 法それ自体に大幅な変更を加える必要はない．これまでの GLR 法による解析をほぼそのまま踏襲することができる．

2 章では，LR 表を生成後に，隣接する終端記号間の接続に関する制約を LR 表に組み込むための基本的な考え方を説明する．第 3 章では制約伝播という考え方を導入する．それにより 2 章で組み込んだ制約を LR 表にさらに伝播させることができる．これは Tanaka ら [Tanaka 95] [Tanaka 94] のアイデアを統合し洗練させたものである．4 章では，LR 表を生成するアルゴリズムに手を加え，LR 表生成時に，隣接する終端記号間の接続に関する制約を LR 表に組み込む方法を説明する．

## 2 接続表の制約の LR 表への組み込–基本的な考え方

自然言語の統語解析で用いる文法的な枠組みとして，これまで CFG がよく使われてきた．一方形態素解析では，形態素間の接続可能性という局所的な制約がよく使われてきた．この局所的な制約は，後述する接続表として表

現することができる．二つの記号間の接続可能性という極めて局所的な制約は，CFG の枠組みで原理的に記述可能であるにしても，いくつかの問題点がある．これらを解決する二つの方法を本章と第 4 章で説明する．いずれも GLR 法の枠組みを用いるが，それにより，これまで異なる解析レベルであるとみなされていたもの，たとえば形態素解析と統語解析とを，GLR 法の枠組みで統一的に扱うことが可能となる [Tanaka 95]．

なお本章以降ではギリシャ文字の$\alpha$, $\beta$,... などは次のいずれかの記号列を表す: 終端記号の列，非終端記号の列，終端記記号と非終端記号が混在した記号列．

## 2.1 CFG と接続表

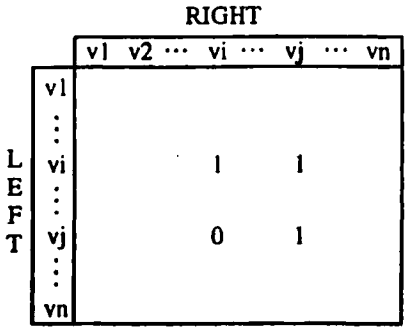



図 1: 接続表の例

局所的な文法的制約の一つに，終端記号の集合 Vt(= {v1, v2, ..., vn}) の各要素間の接続可能性についての制約がある．この制約は，日本語など，単語と単語との間に空白を置かない言語の形態素解析によく用いられている．たとえば，ら行五段活用の動詞の語幹 (決ま) には，ら行五段活用の動詞語尾 (ら，り，る，れ) は付くが，か行五段活用の動詞語尾 (か，き，く，け) は付かないなどという接続可能性に関する制約である．このような記号 (要素) 間の接続可能性は n 行 n 列の表 (connect) として表すことができる．これを接続表とよび，以下のように定義する．

(1) 記号 vi と記号 vj が，この順に接続可能なら，connect[vi,vj] = 1

(2) 記号 vi と記号 vj が，この順に接続不可能なら，connect[vi,vj] = 0

二つの記号間の接続可能性の制約は CFG 規則を用いてももちろん記述できる．たとえば CFG 規則の右辺に，文法カテゴリーの接続可能性を直接記述する．

$$X \rightarrow \alpha \text{ vi vj } \beta$$

この規則の右辺は，vi と vj がこの順に隣接可能であることを直接記述している．

次の CFG を考えてみる．

(1) V → γ vi　　(5) X → α V
(2) V → δ vj　　(6) Y → Wβ
(3) W → vi ζ　　(7) Z → X Y
(4) W → vj η

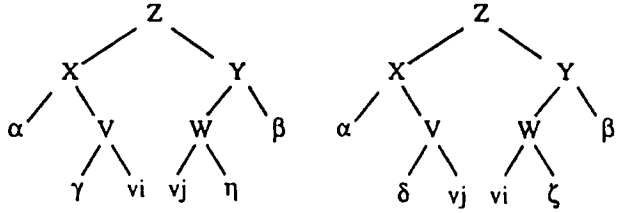



図 2: CFG と接続可能性制約

この場合，図 2の各 CFG 規則の右辺のどこにも，vi と vj(または vj と vi) の接続可能性が記述されていない．

vi と vj がそれぞれ異なる CFG 規則の右辺に現れているので，たとえば vj と vi がこの順に接続不可能である，という制約が CFG 規則の右辺に直接記述されていない．この場合，vj と vi がこの順に接続不可能であり，それ以外の接続 (vivi，vivj，vjvj) が可能であるという制約を CFG でどのように記述したらよいだろうか．

新しい非終端記号を導入し図 3に示す CFG にすることが考えられる．新しい非終端記号を導入して，vi と vj の接続可能性を，{Xi，Xj} の要素と {Yi，Yj} の要素間の接続可能性を記述した (9) から (11) の規則で置き換えるのである．

図 3の CFG を一瞥すると，新しい非終端記号の導入により CFG 規則の数が組み合わせ数的に増大し，CFG の本質が不透明になっていることが分かる．これは文法記述の立場から好ましくない．解析の立場からも，CFG 規則数の増大は時間的，空間的な消費をもたらすので好ましくない．

(1) Vi → γ vi
(2) Vj → δ vj
(3) Wi → vi ζ
(4) Wj → vj η
(5) Xi → α Vi
(6) Xj → α Vj
(7) Yi → Wi β
(8) Yj → Wj β
(9) Z → Xi Yi
(10) Z → Xi Yj
(11) Z → Xj Yj

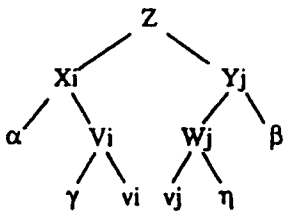
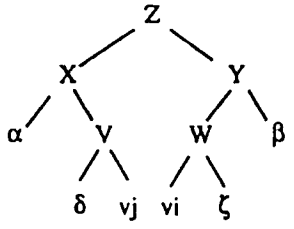



図 3: 接続可能性を考慮した CFG

新しい非終端記号を導入しない方法として，図 4の各 CFG 規則に，接続可能性の制約を付記しておく方法や，ユニフィケーション機構を利用する方法が考えられる．CFG 規則に付記する制約は，手続き付加，補強項などとよばれている [田中　89]．解析過程で CFG 規則を適用する時に，そこに付記されている制約を調べるのである．しかし，他の文法規則に書かれる補強項のことも意識した補強項の記述が複雑になり，大規模な文法規則の記述が困難になる．接続可能性の検査を行うタイミングが遅れるという欠点もある．もし vj と vi がこの順に接続不可能なら，Y という大きな構造を作成したこと自体が無駄になる．




Z → X Y
{ vj と viの接続可能性の検査 }

図 4: 接続可能性検査のタイミング

文法記述の立場から解決すべき問題を列挙すると以下のようになる．

(1) 文法記述者は，接続可能性を考慮した新たな非終端記号の導入を行うことなしに，CFG の記述が可能であること，

(2) 接続可能性を表す接続表の記述者は，CFG 規則とは無関係に接続表の記述が可能であること，

(3) 局所的な制約である接続可能性の検査のタイミングは，早ければ早いほどよい．

(1) と (2) は，文法記述と接続表記述とを独立させたいということである．

GLR 法の枠組みで，(1) から (3) の条件を満足する解決法を 2.2と 4で説明する．

## 2.2 接続表の制約のLR表への組み込み［1］

終端記号の集合 Vt = {v1, v2, ..., vn} の各要素を先読み語としよう．そして vj と vi がこの順に接続不可能で，vivi，vivj，vjvj という接続が可能であるとしよう．

connect[vi,vi] = 1, connect[vi,vj] = 1
connect[vj,vi] = 0, connect[vj,vj] = 1

次の CFG を考える．

(0) S → Z
(1) V → γ vi
(2) V → δ vj
(3) W → vi ζ
(4) W → vj η
(5) X → α V
(6) Y → W β
(7) Z → X Y

(1) から (7) に与えた CFG から，次の LR 表が得られたと仮定する．

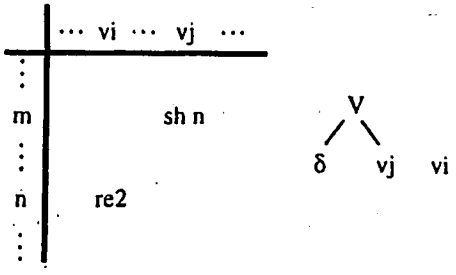



図 5: LR 表の例 1

状態 m で vj をシフトするアクション shn を実行して到達した状態 n において，先読み語が vi の時に行うアクション re2 は，V → δ vj という規則の適用を意味している (図 5)．この場合にはシフトした語 vj と先読み語 vi とがこの順に接続することになるが，接続表からこの接続は許されていないので，re2 (状態 n で先読み語が vi の時に実行するアクション) を LR 表から削除することができる．

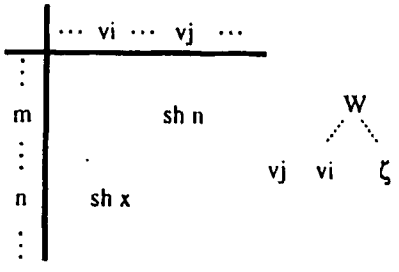



図 6: LR 表の例 2

一方，たとえば W → vi ζを将来適用するために，状態 m で shn を実行し，vj をシフトして到達した状態 n で，次の先読み語が vi の時に実行するアクションを考えてみよう (6)．ζを作り出すためには，次に vi をシフトする必要がある．ところが vj と vi がこの順に接続不可能であるから，状態 n で vi をシフトするアクションは削除してよい．

以上に述べた二つのアクションの削除の例から，アクションを削除する手続きを次のように述べることができる．

アクション Act の直前に実行するアクションがシフトアクション Sh であるようなすべての Act に対して, Sh の先読み語が v, Act の先読み語が w であるとき, connect[v,w] = 0 なら Act を削除する.

# 3 制約伝播によるアクションの削除

2.2で述べた方法により, 接続表で許されないアクションを LR 表からすべて削除する. その結果, LR 表の他のアクションの削除が連鎖的に可能になることがある. これを制約伝播とよぶ. たとえば, 2.2の最後に述べたアクションの削除により, LR 表上で, アクションがすべて削除された状態 (このような状態を空状態とよぶ) ができてしまうことがある. 空状態にシフトするアクションがあるとき, 空状態へシフトしてもその後の解析が続けられないので意味がない. そこで空状態へシフトするシフトアクションは削除することができる.

これを一般化して述べると次のようになる. Act とよばれるアクションを実行した直後に実行すべきアクションが一つもなければ, Act の実行直後にエラーが生じるのは明らかである. したがって, この Act を実行しても解析が続けられないのであるから, これを削除してよい. 一方, Act を実行する直前に実行すべきアクションが一つもなければ, この Act は永久に実行されない無効なアクションであるということになる. したがってこのアクションを削除しても, 解析過程になんら影響しないので削除してよい. これらを以下にまとめてみよう. なお以下の手続き (a) と (b) は [Tanaka 94] を一般化し洗練したものである.

(a) LR 表の初期化手続き:アクション Act に対して, Act の直前に実行するアクションがシフトアクション Sh であるようなすべての Act に対して,

Sh の先読み語が v, Act の先読み語が w であるとき, connect[v, w] = 0 なら Act を削除する.

(b) 制約伝播手続き:削除すべきアクションがなくなるまで, 以下を繰り返す.

LR 表中の各アクション Act について, Act の直前または直後に実行するアクションが一つもなければ, Act を削除する.

```
begin
    手続き (a) を実行する;
    手続き (b) を実行する;
    空の行を圧縮する;
    空の列を圧縮する
end;
```

図 7: 制約伝播のアルゴリズム (その 1)

実験によれば, 制約伝播により LR 表の状態数, レデュース数, シフト数が大幅に減少する [Tanaka 94]. 次章では, 制約伝播手続き (b) の中核部分の Prolog による定義を与える.

# 4 LR 表への接続表の制約の組み込み [2]

本章では正準 LR 表を作成するアルゴリズムそのものを修正することにより, 効率よく接続表の制約を組み込む第二の方法を説明する. これは LR(1) のアイテム (正準アイテム) に関する GOTO グラフを作成する段階で, 接続表の制約を組み込んでしまう方法である. こうして GOTO グラフ作成段階で無駄なアイテムと状態の生成を極力抑制するとともに, 3で述べた (a) の手続きを省略することができる. GOTO グラフ中のアイテム数, 状態数とも事前に削減されるので, GOTO グラフ作成時間の短縮, 使用メモリー量の大幅な削減をはかることができる.

## 4.1 接続表の制約を利用した GOTO グラフの作成法

2.2に与えた文法に対して, 先き読み語 vi, 状態 m で rel を, 先き読み語 vi, 状態 n で re2 を実行する場合を考えてみよう. LR(1) の GOTO グラフ作成段階で次の LR(1) アイテムが, それぞれ状態 m と状態 n にできているはずである. LR(1) アイテム中の記号 ";" の後には先読み語が書かれている.

m: {...., [V → γ vi · ; vi], ....}
n: {...., [V → δ vj · ; vi], ....}

前者の状態 m の場合には，vi に vi が後続している．接続表によりこの接続は許されるものとしよう．後者の状態 n の場合には，vj に vi がこの順に接続している．接続表によりこの接続が許されないとすれば，アイテム [V → δ vj · ; vi] を削除することができる．

しかし熟考すれば状態 n のアイテム [V → δ vj · ; vi] が生成される以前の状態 (たとえば状態 p) で次のアイテム，

p: {...., [V → · δ vj; vi], ....}

が生成されているはずである．接続表より，vj と vi がこの順に接続しないとすれば，その時点でこのアイテム [V → · δ vj; vi] の生成を禁止することができる．こうして，アイテム [V → · δ vj; vi] から派生したり状態推移してできるはずの新しい LR(1) アイテム (たとえば [ V → δ · vj; vi] や [ V → δ vj · ; vi] など) の生成が自動的に抑制される．

以上に述べた接続表の制約を利用した GOTO グラフ作成時のアイテム削除法を一般化することができる [Li 95]．これを以下で説明する．

[右向き矢印 (→) の直後にドット記号 (•) をもつアイテム生成手続き]

与えられた CFG と接続表とを用いて，次の 1) または 2) のいずれかの場合にアイテムを生成する．ただし，記号 X を根とする解析木の右分枝の先端 (葉) に現れる記号の集合は，関数 last(X) を実行して得られるものとする．特に記号 X が終端記号の場合には，last(X) = {X} である．

1) アイテム [V → · ξ X; vi] に対して，last(X) に属す要素と vi の対のうち，この順に接続可能な対が存在する場合に限り，このアイテムを生成する，

2) 記号 X をシフトして到達した新状態に存在するアイテム [V → · vi ξ ; w] に対して，last(X) に属す要素と vi の対のうち，この順に接続可能な対が存在する場合に限り，このアイテムを生成する．

接続表は一般にスパースであるため接続が禁止されているペアが多く，上記したアイテム生成手続きにより，GOTO グラフ作成段階で，大量のアイテムの生成が抑制される．このような大量のアイテムの削減は，LR 表の大量の状態数の削減につながる．こうして使用記憶空間を大幅に削減し，最終的な修正済みの LR 表を得る時間を一桁以上短縮できることが報告されている [Li 95]．LR 表を作成する上での大きな問題は，LR 表のサイズにある．最悪の場合の状態数は，CFG 規則数の指数オーダになることが知られている．CFG の規模が小さければ，これはさして大きな問題にはならない．CFG 規則数が数千の規模になる場合には，本節で説明した方法は，制約を早い時点で利用して状態数の少ない LR 表を作成することができるという利点が生きてくる．

本節の方法で得た LR 表は，3で述べた制約伝播のアルゴリズム (その 1) の手続き (a) が既に施されたものになっていることに注意．制約伝播の核となる部分は制約伝播のアルゴリズム (その 1) の手続き (b) の部分である．そこで，Act の直前または直後に実行するアクションを求めるアルゴリズムを，Prolog の述語 succ により定義してみる．

```
succ(S0,LookAhead0,[sh,State],State,LookAhead,Action) :-
        lr(S0,LookAhead0,[sh,State]),
        lr(State,LookAhead,Action).

succ(S0,LookAhead0,[re,Rule0],State,LookAhead0,Action) :-
        lr(S0,LookAhead0,[re,Rule0]),
        rule(Rule0,ProdList),
        reverse(ProdList,RevProdList),
        succ1(RevProdList,S0,State,LookAhead0,Action).

succ1([NT],ToState,State,LA,Action) :-
        goto(ToState,NT,State),
```

```
        lr(State,LA,Action).

succ1([NT|NTs],ToState,State,LA,Action) :-
        (goto(FromState,NT,ToState)
         ;
         lr(FromState,NT,[sh,ToState])),
        succ1(NTs,FromState,State,LA,Action).

reverse(X,Y):-reverse1(X,[],Y).
reverse1([],Z,Z).
reverse1([X|Y],Y1,Z) :- reverse1(Y,[X|Y1],Z).
```

述語 succ の第一引数から第三引数にそれぞれ現在の状態，現在の先読み記号，現在のアクションを与え，第四引数から第六引数までを変数にして実行すると，第四引数から第六引数にそれぞれ次に推移する状態，次の先読み語，次に実行するアクションが返される．逆に第四引数から第六引数にアクションに関する情報 (値) を与え，第一引数から第三引数までを変数にして述語 succ を実行すると，直前に実行したアクションが得られる．したがって，制約伝播のアルゴリズム (その 2) では，当該アクションの直前と直後に実行するアクションが述語 succ により一つも得られなければ (fail すれば)，当該アクションを削除することになる．

ここで以下の注意をしておきたい．LR 表の各エントリーについては，アクション部は述語 lr(State, Lookahead, Action) として，GOTO 部は述語 goto(State, Nonterminal, ToState) としてあらかじめアサートされているものとする．ただし Action は，shj なら [sh, j] の形式で，rej なら [re, j] の形式で与えるものとする．また述語 rule は第一引数に CFG 規則の番号を与えると，第二引数にその規則の左辺の記号と右辺の記号列とを cons したものを計算する．たとえば，2.2のはじめに与えた CFG の 7 番目の規則に対して rule(1,L) を実行すれば，変数 L に [s,x,y,z] が返される (prolog では大文字で始まる記号は変数を表すので，記号 S,X,... の代わりに記号 s,x,... を用いる)．述語 reverse は第一引数のリストの逆順のリストを第二引数に返す述語である．以上のことから，制約伝播のアルゴリズム (その 2) は図 8に示すものになる．

```
begin
   LR 表中のすべてのアクションについて，新たな制約伝播が起き
   なくなるまで以下を繰り返す:
     各状態 S に対して，先読み記号 LA の各アクション
   Act について，succ(S,LA,Act,S1,LA1,Act1) が fail す
   れば (Act の直前または直後に実行するアクションが
   一つもなければ)，Act を LR 表より削除する;
   空の行と列を圧縮する
end;
```

図 8: 制約伝播のアルゴリズム (その 2)

## 5 あとがき

本稿では接続表の制約を LR 表に組み込む二つの方法を説明した．2.2で説明した方法の利点は，既存の標準的な LR 表作成アルゴリズムがそのまま使えることである．欠点は，CFG 規則数が増え数千の規模になると，(制約伝播を施す前の) 中間段階でできる LR 表のサイズが巨大になり実用に耐えないことである．

4章で説明した第二の方法のの利点は，CFG 規則の数が増えても中間段階でできる LR 表のサイズが大きくならずに済むことである．これは LR 表を作るアルゴリズムを修正し，LR 表を作るときに接続表の制約を組み込み無駄なアイテムの生成を抑制するからである．そのため既存の標準的な LR 表作成アルゴリズムを修正する必要

があるが，CFG 規則数が増えるにつれて，この第二の方法を用いる必要がある．実験によれば，標準的なワークステーション上での LR 表生成時間も，規則数が 1208 の規模の CFG に対して 30 秒程度と高速である [Li 95]．

本稿で説明した自然言語解析の新しい方法は，形態素解析と統語解析とを統合する方法として有効であることが実験的に示されている [Tanaka 95][植木 95]．この場合には形態素間の接続表を用いる．音声認識にも有効であることが実験的に示されている [Tanaka 94][Li 94]．この場合には異音間の接続表を用いる．

今後の課題として，複数の接続表の制約を LR 表に組み込む方法の開発がある．われわれは一つの解法をすでに見い出しているが，これについては次の発表の機会に譲りたい．

## 参考文献


[Aho 86] Aho, A., Ravi, S., Ullman, J. *Compilers, Principle, Techniques, and Tools.* Addision Wesely, (1986).

[Kay 80] Kay M. *Algorithm Schemata and Data Structure in Syntactic Processing.* TR CSL80-12, Xerox PARC, (1980).

[Kipps 91] Kipps J.N. *GLR Parsing in Time O(n3).* in (18), pp. 43-59 (1991).

[Li 94] Li H., Takezawa T., Singer H., Hayashi T., and Tanaka H. *An efficient phoneme-context-dependent LR table and its applications in continuous speech recognition.* Proc. of the Acoustic Society of Japan(ASJ) autumn meeting, 3-8-9, pp. 125–126, Nov. 1994.

[Li 95] Li H., Suresh K.G. and Tanaka H. *Incorporation of Connection Constraints into Generation Process of Allophone-Base LR Table.* 情報処理学会第 50 回全国大会講演論文集, (1995).

[Sippu 90] Sippu S. and Soisalon-Soininen E. *Parsing Theory, Vol.II: LR(k) and LL(k) Parsing.* Springer-Verlag, 303-316(1990).

[Shann 91] Shann P. *Experiments with GLR and Chart Parsing.* in (18), pp.17-14 (1991).

[田中 89] 田中穂積. 自然言語解析の基礎, 産業図書, (1989).

[Tanaka 95] Tanaka H., Tokunaga T., and Aizawa M. *Integration of Morphological and Syntactic Analysis based on LR Parsing.* Journal of Natural Language Processing, 2, 2, pp. 59-74(1995).

[Tanaka 94] Tanaka, H., Li, H. and Tokunaga, T. *Incorporation of Phoneme-Context-Dependence in LR Table through Constraint Propagation Method.* In Proceedings of AAAI94 Workshop on the Integration of Natural Language and Speech Processing, pp. 15–22 (1994).

[Tomita 86] Tomita M. *Efficient Parsing for Natural Language.* Kluwer Academic Publishers (1986).

[Tomita 91] Tomita M.(ed.). *Generalized LR Parsing Technologies.* Kluwer Academic Publishers (1991).

[植木 95] 植木正裕, 徳永健伸, 田中穂積. *EDR* 辞書を用いて形態素解析と統語解析を行なうシステム. EDR 電子化辞書利用シンポジウム論文集, pp. 33-39 (1991).