# ＬＲパーサを用いた文字列置換アルゴリズム

2C−2

津田　和彦†　中村　雅巳†　青江　順一††

† 住友金属工業(株)　†† 徳島大学

## 1．はじめに

著者らは，これまで日本語文書の短縮法についての研究を行ってきた[1][2]．著者らが提案している文書短縮法は,より少ない文字数で表現できる形態素列を短縮規則として定義し，文書中より短縮規則が適用できる部分を検索・置換することで実行される．しかし，短縮規則が増加すると文書中の特定部分で複数の短縮規則が適用でき一般的なストリング・パターン・マッチング手法では，適用可能な全ての短縮規則を検出できないという問題点が生じてきた．

そこで，文脈依存文法の構文規則とこの短縮規則が類似し，また，文脈依存言語に対するＬＲパーサの還元動作と文書短縮のための置換動作が類似している点に着目して，バックトラックのない文字列置換アルゴリズムを提案する．本手法で得られた文字列置換マシンはバックトラックが不必要であるという特徴に加え，従来のコード最適化マシンにおいて不可能であったより良い規則の選択が可能となる．

## 2．文脈依存文法と置換規則

### 2．1　文脈依存文法

文脈依存文法(context-sensitive grammar)を$G=(V_N,V_T,P,S)$で表す．ここで，$V_T,V_N$はそれぞれ非終端語(terminals),終端語(nonterminals)の有限集合，Ｐは$\alpha\rightarrow\beta$なる生成規則(productions)の有限集合である．Ｓは$V_N$の要素で始記号(start symbol)を表す．$V=V_N\cup V_T$とし，Ｖ上のすべての有限長の記号列の集合を$V^*$とする．$V^+$は$V^*$から空集合$\epsilon$を除いた集合を表し，本章では特に断らない限り，V,VT*,V*の要素をそれぞれX,Y,Z;x;$\alpha,\beta,\gamma,\delta$;で表す．

### 2．2　置換規則

文脈依存文法の構文規則$A_{p1}A_{p2}\cdots A_{pnp}\rightarrow B_{p1}B_{p2}\cdots B_{pmp}$,$(0\leqq mp<np)$の左右を交換したものが置換規則と見なすことができる．このようにして得られた第ｐ番目の置換規則を，

$B_{p1}B_{p2}\cdots B_{pmp}\rightarrow A_{p1}A_{p2}\cdots A_{pnp}$，$(0\leqq mp<np)$

と定義する．

以上で定義された置換規則の集合を$P_{OPT}$で表し，$P_{OPT}$の最大値をｖとする．そして，ＬＲパーサの構成法をこの置換規則$P_{OPT}$に応用して得られたマシンを最適化マシンOPTと呼ぶ．例として以下の規則を用いる．

$P_{OPT}$={ＮＮ→ $\epsilon$，ＭＮ→ＮＭ，ＤＮ→ＮＤ，
ＮＡ→Ｓ，ＮＳ→Ａ，ＬＩＴ→Ｖ}

また，これらの規則より作成した解析表を図１に示す．

### 2．3　マシンOPTの解析表の構成法

マシンOPTの解析表の構成法を説明するために，次のようにOPT項を$[\alpha\rightarrow\beta\cdot\gamma,\delta]$と定義する．$\alpha\rightarrow\beta\cdot\gamma$はマシンOPTの解析が$\beta$まで終了し，$\gamma$を次に解析しようとしている状態を表す．また，$\delta$はW*の要素である．

（定義　1）OPT項の集合Ｊに対する閉包CLOSURE(J)は次のようにして得られる．

1）CLOSURE(J)をＪに初期設定する．
2）$[\alpha\rightarrow\beta\cdot\gamma,\delta]\in$CLOSURE(J)に対して，$\alpha'$が$\gamma\delta$の接頭辞である置換規則$\alpha'\rightarrow\beta'$が$P_{OPT}$に含まれるなら$[\alpha'\rightarrow\beta',\ \delta']$をCLOSURE(J)に加える．ただし,$\delta'$は$\gamma\delta=\alpha'\delta'$を満足する．
3）$[\alpha\rightarrow\beta\cdot\gamma,\ \delta]\in$CLOSURE(J)に対して，$\gamma\delta$が$\alpha'$の接頭辞である置換規則$\alpha'\rightarrow\beta'$が$P_{OPT}$に含まれるなら$[\alpha'\rightarrow\beta',\ \epsilon]$をCLOSURE(J)に加える．
4）上記２），３）を新たに加えるものがなくなるまで繰り返す．

最適化マシンOPTにおいてバックトラックをなくすためには，一般的ＬＲパーサのCLOSUREの定義に加えて，さらに定義１の３）の手順を追加しなくてはならない．これは，次のコード列の変化において，バックトラックを生じさせないためにある．

$\beta\ \ \beta'$ ・・・・・（1）
$\beta\ \ \alpha'$ ・・・・・（2）
$\beta\ \ \gamma\delta$ ・・・・・（3）
$\alpha\ \ \delta$ ・・・・・・（4）

$\beta'$が$\alpha'$に置換され，$\alpha'=\gamma\delta$であり，$\beta\gamma$が$\alpha$に置換されることを表す．状態から状態への遷移を考えるために$[\alpha\rightarrow\beta\cdot A\gamma,\delta]$を含んだ閉包から$[\alpha\rightarrow\beta A\cdot\gamma,\delta]$を含んだ閉包へのＡによる遷移を次のように定義する．

（定義　2）　ＪをOPT項の集合，Ａ∈Ｗとしたとき，次のGOTO関数を定義する．

GOTO(J,A)=CLOSURE({[$\alpha\rightarrow\beta A\cdot\gamma,\delta$]|[$\alpha\rightarrow\beta\cdot A\gamma,\delta$]∈Ｊ})

置換規則$P_{OPT}$に対するすべての状態集合$K_{OPT}$は次の手順により求まる．

1）$K_{OPT}$を{CLOSURE({[$\alpha_1\rightarrow\beta_1,\epsilon$], [$\alpha_2\rightarrow\beta_2,\epsilon$], …, [$\alpha_v\rightarrow\beta_v,\epsilon$]})}と初期設定する．
2）以下を新たな集合が$K_{OPT}$に加えられなくなるまで繰り返す．

任意のＪ∈$K_{OPT}$とＡ∈Ｗに対しGOTO(J,A)が空集合でなく$K_{OPT}$に入っていなければ，それを$K_{OPT}$に加える．

定義２の１）で初期化される状態は初期状態と呼ばれ$J_0$で表す．置換規則$P_{OPT}$には定義２の１）のようにすべての規則に対する状態を加える必要がある．

置換規則$P_{OPT}$に対して状態集合$K_{OPT}$が求まれば，マシンOPTの動作を決定できるが，$K_{OPT}$においてもＬＲパーサと同様にR·R矛盾をもつ状態が存在する．ＬＲパーサではこの矛盾を先読みにより解決しているが，マシンOPTでは次の関数EFF(efficientの略)を定義し，この矛盾の解決法を与える．

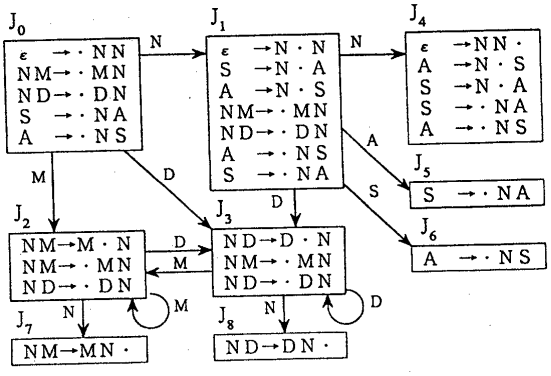


図１　マシンOPTの解析表

---

A String Replacement Algorithm by Using LR Parser.

Kazuhiko TSUDA†, Masami NAKAMURA†, Jun-ichi AOE††

† Sumitomo Metal Industries,LTD. †† University of Tokushima

$\alpha \rightarrow \beta \in P_{OPT}$ならば，$\sharp\beta$は$\beta$に含まれるWの要素数である．OPT項の第２項を省略して説明する．一般的にR･R矛盾をもつ状態Jは，$[\gamma \rightarrow \beta \cdot]$なる１つのOPT項とこの$\beta$を右辺に接尾辞をもつ規則$\gamma_i \rightarrow \gamma'_i \beta$, $1 \leqq i \leqq n$によるOPT項$[\gamma_i \rightarrow \gamma'_i \beta \cdot]$から成り立っていると考えられる．またR･S矛盾をもつ状態Jは，シフト動作に関係した$[\alpha_j \rightarrow \alpha'_j \cdot \delta_j]$, $1 \leqq j \leqq m$と還元動作に関係した$[\gamma_i \rightarrow \gamma'_i \beta \cdot]$, $1 \leqq i \leqq n$から成ると考えらる．従って，以後の説明では図２の(a)と(b)のR･R矛盾とR･S矛盾の一般形を用いる．

ＬＲパーサでは，これらの矛盾に対して先読みにより一意的に正しい動作を決定されなければならないが最適化マシンにおいてはより効率的なコードを得るようにこれらの矛盾に対して動作を決定しなければならない．すなわち，この関数EFFは局所的にコードが最短になる動作を決定するために使用される．

以下に$K_{OPT}$からマシンOPTの動作を与えるための定義を与える．関数ACTIONと同様に使用されるが，動作の矛盾をもつ状態Jに対してのみ，先読みコード列の要素の集合LOOK(J)を定義し，LOOK(J)に先読みが含まれる場合の動作をFIRST(J)，そうでない場合の動作をSECOND(J)で表す．

（定義　３）

１）　状態Jが動作矛盾をもたない場合

ａ）$[\alpha \rightarrow \beta \cdot \gamma, \delta] \in$ Jかつ$\gamma \neq \varepsilon$ならば

ACTION(J)＝shiftを定義する．

ｂ）$[\alpha \rightarrow \beta \cdot, \delta]$かつ$\alpha \rightarrow \beta$が第ｐ番目の規則ならば，

ACTION(J)＝reduce ｐを定義する．

２）状態JがR･R矛盾をもつ場合（図２の(a)参照）

$\gamma_1 \rightarrow \gamma'_1 \beta, \gamma_2 \rightarrow \gamma'_2 \beta, \cdots, \gamma_n \rightarrow \gamma'_n \beta$の中で関数EFFの値が最大のものを$\gamma_i \rightarrow \gamma'_i \beta$, $1 \leqq i \leqq n$と仮定し，$\gamma \rightarrow \beta$を第ｐ番目，$\gamma_i \rightarrow \gamma'_i \beta$を第ｑ番目の規則と仮定する．

ａ）　$EFF(\delta_1 \rightarrow \delta_2 \gamma\ \delta) > EFF(\gamma_i \rightarrow \gamma'_i \beta)$の場合

ACTION(Ｊ)＝conflict，LOOK(Ｊ)＝{$\delta$}，

FIRST(Ｊ)＝reduce ｐ，

SECOND(Ｊ)＝reduce ｑと定義する．

ｂ）　$EFF(\delta_1 \rightarrow \delta_2 \gamma\ \delta) \leqq EFF(\gamma_i \rightarrow \gamma'_i \beta)$の場合

ACTION(Ｊ)＝reduce ｑと定義する．

３）状態JがＲ・Ｓ矛盾をもつ場合（図２の(b)参照）

$\gamma_1 \rightarrow \delta_1, \gamma_2 \rightarrow \delta_2, \cdots, \gamma_n \rightarrow \delta_n$の中で関数EFFの値が最大のものを$\gamma_i \rightarrow \delta_i$，$1 \leqq i \leqq n$と仮定し，この規則番号を第ｐ番目とする．また，$\alpha_1 \rightarrow \beta_1 \delta_1, \alpha_2 \rightarrow \beta_2 \delta_2, \cdots, \alpha_m \rightarrow \beta_m \delta_m$は関数EFFの値の大きい順にソーティングされているものと仮定する．

ａ）$EFF(\alpha_1 \rightarrow \beta_1 \delta_1) > EFF(\gamma_i \rightarrow \delta_i) \geqq EFF(\alpha_{l+1+1} \rightarrow \beta_{l+1} \delta_{l+1})$，$1 \leqq l \leqq m$の場合

ACTION(Ｊ)＝conflict，LOOK(Ｊ)＝{$\delta_1, \delta_2, \cdots, \delta_e$}

FIRST(Ｊ)＝shift，SECOND(Ｊ)＝reduce ｐと定義する．

ｂ）$EFF(\gamma_i \rightarrow \delta_i) \geqq EFF(\alpha_1 \rightarrow \beta_1 \delta_1)$の場合

ACTION(Ｊ)＝reduce ｐと定義する．

図２(a)の状態Ｊに対して，n+1種類の還元動作が可能であるが，本論文で新しく提案する矛盾解決法は，局所的に短いコードを生成する還元動作を選び出すことである．最も局所的な矛盾解決法は，$\gamma_1 \rightarrow \gamma'_1 \beta, \gamma_2 \rightarrow \gamma'_2 \beta, \cdots, \gamma_n \rightarrow \gamma'_n \beta$中で関数EFFの値が最大の規則（$\gamma_i \rightarrow \gamma'_i \beta$）による還元動作を選び出すことである．すなわち，この選択はＲ・Ｒ矛盾に関係した規則の情報だけに依存している．

しかし，定義６の２）では，規則$\gamma \rightarrow \beta$に対する局所的な先読み情報を利用して矛盾を解決している．状態Ｊの$OPT項[\gamma \rightarrow \beta \cdot]$は，状態$J_1$の$OPT項[\delta_1 \rightarrow \delta_2 \cdot \gamma\ \delta]$の閉包から得られたものであるので$\gamma \rightarrow \beta$による還元動作は，先読みコード列$\delta$が存在すれば，$\delta_1 \rightarrow \delta_2 \gamma\ \delta$による還元動作が起こることが保証される．従って$EFF(\delta_1 \rightarrow \delta_2 \gamma\ \delta) > EFF(\gamma_i \rightarrow \gamma'_i \beta)$であって，状態Ｊの先読みコード列が$\delta$であれば，$\gamma \rightarrow \beta$による還元動作を選択すべきである．また，$EFF(\delta_1 \rightarrow \delta_2 \gamma\ \delta) < EFF(\gamma_i \rightarrow \gamma'_i \beta)$ならば$\gamma_i \rightarrow \gamma'_i \beta$による還元動作が選択されるべきである．

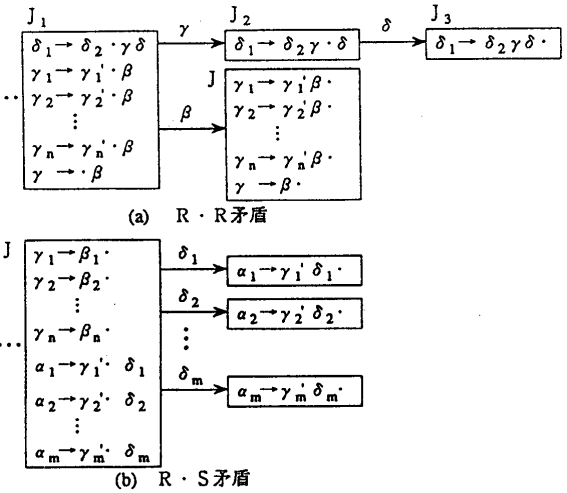


図２　R・R矛盾とR・S矛盾

図２(b)の状態Ｊにおいて，コード最適化では，$[\alpha_j \rightarrow \alpha'_j \cdot \delta_j]$, $1 \leqq j \leqq m$なる項に対するシフト動作は，後に$\alpha_j \rightarrow \alpha'_j \cdot \delta_j$による還元動作が起こることが保証されない．従って，状態Ｊに対する最も局所的な矛盾解決法は，$\gamma_1 \rightarrow \beta_1, \gamma_2 \rightarrow \beta_2, \cdots, \gamma_n \rightarrow \beta_n$の中で関数EFFの値が最も大きい規則（$\gamma_i \rightarrow \beta_i$）による還元動作を選び出すことである．

しかし，定義３の３）では，シフト動作に対する局所的な先読み情報を利用して矛盾を解決している．$EFF(\gamma_i \rightarrow \beta_i) \geqq EFF(\alpha_1 \rightarrow \beta_1 \delta_1)$ならば，先読みなしに規則$\gamma_i \rightarrow \beta_i$による還元動作が選択されるべきである．これに対し$EFF(\alpha_1 \rightarrow \beta_1 \delta_1) > EFF(\gamma_i \rightarrow \beta_i) \geqq EFF(\alpha_{l+1+1} \rightarrow \beta_{l+1} \delta_{l+1})$, $1 \leqq l \leqq m$であって$\delta_k$, $1 \leqq k \leqq l$なる先読みコード列が存在すれば，$\alpha_k \rightarrow \beta_k \delta_k$による還元動作が保証されるので，シフト動作が選択されるべきである．ただし，この$\delta_k$が存在しない場合は，$\gamma_i \rightarrow \beta_i$による還元動作が選択される．

これを実行するマシンOPTの解析は，２つのスタックを用いて行われるが，詳細はD.A.Waltersの文献(4)を参照されたい．

## ３．まとめ

本稿では，バックトラックが不必要かつより良い規則の選択が可能となる文字列置換アルゴリズムを提案した．本アルゴリズムは，局所的コード最適化にも応用可能である．今後は，本アルゴリズムを文書短縮に適用し評価を早急に進めて行きたい．また，局所的コード最適化にも応用し，A.S.Tanenbumら(3)による最適化マシンとの比較・評価も行う予定である．

## 参考文献

(1) 津田和彦,中村雅巳,青江順一：“形態素置換による文書短縮法”，信学論Vol.J75-D-II，No.3，pp.619-627，(1992-03)

(2) 津田和彦,中村雅巳,青江順一：“文書短縮のための文字列置換アルゴリズム”，情処学NL究報，85-5，(1991-09)

(3) A.S.Tanenbaum,H.Staveren and J.W.Stevenson：“Using peephole optimization on intermediate code”,ACM Trans. Prog. Lang.Syst., 4,1, pp.21-36, (1982)

(4) D.A.Walters：“Deterministic context-sensitive language: part1 part2” Inf. Control, 17, pp.14-40, pp.41-61, (1970)

(5) J.W.Davidson and A.WcFraser：“The design and application of a retargetable peephole optimizer” ACM Trans. Prog. Lang. Syst., 2, 2, pp.191-202, (1980)