情報処理学会研究報告
IPSJ SIG Technical Report

Vol.2009-NL-192 No.1
2009/7/22

ダブル配列により状態遷移を高速化したLR解析表の実現

重越秀美[†1]　蔵満琢麻[†1]　望月久稔[†1]

LR 解析は，C 言語や Pascal 等のプログラミング言語のコンパイラや，自然言語処理，音声認識など幅広い分野に使用されており，時間的，領域的に効率的な LR 解析表の実現が求められる．本論文では，トライ構造を効率的に実現するダブル配列の遷移を拡張し，従来手法より高速に状態遷移できる LR 解析表の実現法を提案する．実験により，構文解析器生成系の 1 つである Bison に対して，解析時間を ANSI C で 15.66%，Pascal で 15.15%削減した．

An Implementation of LR Parsing Table Using Double-Array for Speeding Up State Transition

Hidemi Shigekoshi,[†1] Takuma Kuramitsu[†1]
and Hisatoshi Mochizuki[†1]

LR parsers are used widely, such as programing language compilers, natural language processing and speech recognition. So it is important to implement efficient LR parsing tables. In this paper, we present an efficient implementation of LR parsing table using extended double-array that is an efficient trie data structure. Our experiment show that proposal method can decrease the parsing time about 15% in comparison with parser generator Bison.

1. はじめに

シフト還元構文解析の一種である LR 解析[1] は，文脈自由文法の一種である LR 文法の構文解析に用いられる手法で，あらかじめ文法に対して前処理を行って解析表を作成することで高速な構文解析を実現する．LR 解析は，C 言語や Pascal などのプログラミング言語のコンパイラ，情報抽出[2]，テキストの誤り訂正[3]，自然言語の構文解析[4]，音声認識[5] など，様々な分野に用いられており，時間的，領域的に効率的な LR 解析表の実現法が求められる．現在，構文解析器生成系として一般的に用いられる Yacc[6] や Bison[7] は，Aho らの提案した表圧縮法[1] を用いて，記憶領域のコンパクト性と解析速度の高速性をあわせもつ LR 解析表を生成する．

本論文では，トライ構造を効率的に実現するダブル配列[8)–10)] を導入して，LR 解析表の状態遷移に要する計算量を抑制し，従来手法よりも解析速度が高速な LR 解析表の実現法を提案する．以下，2 章で LR 解析表とダブル配列について述べ，3 章で提案手法について述べる．4 章で提案手法を実験により評価し，5 章で本論文の総括と今後の課題を述べる．

2. LR 解析表とダブル配列

本章では，LR 解析表の構成と，Aho らの表圧縮法を述べた後，提案手法で用いるダブル配列について説明する．以下，LR 解析表作成時に登録する文法 G に含まれる規則の数を $|G|$，規則 g の規則番号を I_g と表記し，文法中に存在する終端記号の集合を T，非終端記号の集合を N，それぞれの要素数を $|T|$，$|N|$ と表記する．

2.1 LR 解析表の構成

LR 解析表は，ACTION 関数と GOTO 関数からなる．ACTION 関数は，状態 x と終端記号 a を引数とし，shift(t)，reduce(I_g)，accept，error のいずれかのアクションを返す関数である．ここで，shift(t) は，状態 t をスタックにプッシュ（状態 t へ遷移）し，新しい終端記号をトークン列から取得するアクションで，reduce(I_g) は規則 g を還元するアクションを表す．以下，状態 x における終端記号 a によるアクションを，ACTION(x, a) と記述する．GOTO 関数は，reduce(I_g) 実行時に呼び出される関数で，規則 g の還元で生じる非終端記号 A による遷移先 t を返す．

LR 解析表を実現する最も単純な方法は，状態と終端・非終端記号を添字とする 2 次元配列を用いることである．図 1(a) に示す文法 P に対して実現した 2 次元配列による LR 解析表を図 1(c) に示す．図 1(c) において，空のエントリは error を表す．また，図 1(c) における goto 遷移グラフを図 1(b) に示す．ここで，goto 遷移は ACTION 関数実行時において発生する状態から状態への遷移を示す．2 次元の配列構造を用いて LR 解析表を実現すると，解析時に次のアクションを $O(1)$ で決定できるが，図 1(c) に示すように配列がスパースとなり，記憶効率の面で問題がある[1]．

†1 大阪教育大学
Osaka Kyoiku University

Vol.2009-NL-192 No.1
2009/7/22

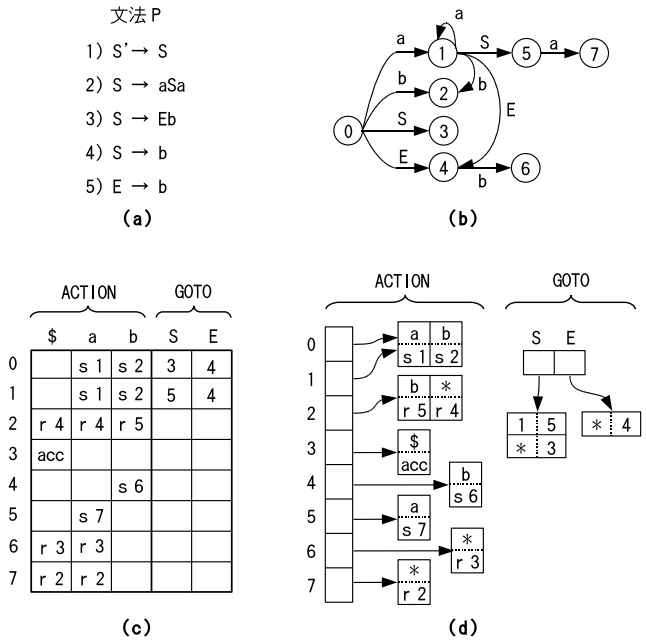

図 1 文法 P に対する goto 遷移グラフと LR 解析表

2.2 解析表の圧縮

構文解析の速度を多少犠牲にすれば，図 1(d) に示すように，終端記号とアクションを対とするリスト構造により ACTION 関数を管理し，遷移元の状態番号と goto 遷移先の状態番号を対とするリスト構造により GOTO 関数を管理することで，記憶領域を大幅に抑制できる[1)]．

また，各状態におけるアクションの中で最も多く現れる還元規則（以下，デフォルト還元規則）をデフォルトのアクションとして管理し，アクションが全て等しい状態のアクションを共有する[1)] ことでさらに記憶領域を抑制できる．GOTO 関数に関しては，各非終端記号において，遷移先として最も多く現れる状態をデフォルトの goto 遷移先として取り扱うことで記憶領域を抑制できる．図 1(d) は，状態 0，1 における ACTION 関数を共有し，デフォルトのアクションを記号「＊」との対で管理する．

配列構造の高速性とリスト構造のコンパクト性をあわせもつ表圧縮法としては，1 次元の配列 Base，Check，Action を用いて，式 (1) により ACTION 関数におけるすべてのエントリを 1 次元配列 Action に格納する手法[1)] がある．式 (1) は，ACTION(x，a) の格納位置を，状態 x の Base 値と終端記号 a の内部表現値の和により決定する．

$$\begin{cases} entry = \mathrm{Base}[x] + a \\ \mathrm{Check}[entry] = a \\ \mathrm{Action}[entry] = \mathrm{ACTION}(x,\ a) \end{cases} \tag{1}$$

以上で説明した表圧縮法は，Yacc[6)] や Bison[7)] が出力する LR 解析表にも利用されている．

2.3 ダブル配列

本節では，提案手法で用いるダブル配列について説明する．青江らは，初期状態を除く各状態の入次数が 1 である木構造の goto 遷移表を効率的に実現する手法としてダブル配列[8)] を提案した．ダブル配列[10)] は 2 本の配列 Base と Check を用いて，状態 x から遷移種 a による遷移先 t を式 (2) により定義する手法で，配列の添字が状態番号に対応する．

$$\begin{cases} t = \mathrm{Base}[x] + a \\ \mathrm{Check}[t] = a \end{cases} \tag{2}$$

ダブル配列は，式 (1) により実現する状態遷移表と比較すると，配列 Action が不要となる分，記憶効率の面で優れている．また，Base 値と遷移種の和が遷移先となり，間接参照せずに遷移先へ直接に遷移できるため，時間効率の面でも優れている．しかし，ダブル配列は遷移元の Base 値と遷移種により遷移先の状態番号が一意に定まり，遷移の数だけ異なる遷移先状態が定義される[9)] ため，複数の遷移を特定の状態へ直接定義することができず，一般の有限オートマトンにおける遷移表を作成できない．

そこで青江らは，複数定義される遷移先状態の集合 $\{t_1 \ldots t_m\}$ を，1 つの状態に統合して取り扱い，擬似的に入次数が 2 以上となる状態をダブル配列上に定義する手法[9)] を提案した．この手法は，遷移先状態集合 $\{t_1 \ldots t_m\}$ から任意の状態 t_i を状態集合の代表として取り扱い，t_i 以外の状態に t_i へのポインタを定義することで状態を統合する[9)]．以下，状態集合の代表である状態を統一状態，統一状態へのポインタを管理する状態を間接状態と呼び，統一状態と間接状態を導入したダブル配列を拡張ダブル配列と呼ぶ．

3. LR 解析表における状態遷移の高速化

本章では，ダブル配列を導入して，従来の表圧縮法よりも高速に状態遷移が行える LR 解

Vol.2009-NL-192 No.1
2009/7/22

析表の実現法を提案する．以下，ダブル配列による LR 解析表の実現法について述べた後，既存の表圧縮法を提案手法に適用する方法を説明する．その後，提案手法の解析アルゴリズムを解説する．

3.1 ダブル配列による LR 解析表の実現

LR 解析表における GOTO 関数は統一状態と間接状態を導入した拡張ダブル配列を用いることで容易に実現できる．本節では拡張ダブル配列上に還元状態を新たに導入して LR 解析表を実現する方法を提案する．

提案手法では，状態 x が終端記号 a により文法規則 g を還元するとき，状態 x から終端記号 a による遷移先状態 t を作成し，Base[t] に文法規則 g を表す値を格納する．以下，この状態 t を還元状態と呼び，還元状態を導入したダブル配列を LRDA と呼ぶ．

還元状態の導入により，LRDA 上には，統一状態，間接状態を含む 3 種類の状態が存在する．そこで，統一状態 α，間接状態 β，還元状態 γ の Base 値を，それぞれ以下の式 (3)，式 (4)，式 (5) で定義する．式中，統一状態 α における遷移先状態の集合を $S(\alpha)$，遷移ラベルの集合を $L(\alpha)$ とする．ここで，$|S(\alpha)| = |L(\alpha)|$，$0 < I_g \leq |G|$ である．

$$\mathrm{Base}[\alpha] = base, \qquad (base \geq 0,\ S(\alpha) = \{base + a;\ a \in L(\alpha)\}) \tag{3}$$

$$\mathrm{Base}[\beta] = -(|G| + t), \qquad (\beta \text{ が統一状態 } t \text{ へのポインタをもつ}) \tag{4}$$

$$\mathrm{Base}[\gamma] = -I_g, \qquad (\gamma \text{ が文法規則 } g \text{ の規則番号 } I_g \text{ をもつ}) \tag{5}$$

式 (3)〜式 (5) により，Base[β]$< -|G| \leq$Base[γ]$< 0 \leq$Base[α] が常に成り立つため，Base 値を参照することで，状態の種類を判別できる．

この性質を利用し，提案手法は統一状態 x において，終端記号 a によるアクションを遷移先状態の種類により決定する．統一状態 x から終端記号 a による遷移先の状態を t とすると，状態 t が統一状態であれば shift(t)，間接状態であれば shift($-$(Base[t] + $|G|$))，還元状態であれば reduce($-$Base[t]) を行う．ただし，提案手法では，開始規則の還元を構文受理（accept）とし，遷移先が存在しない場合を構文エラー（error）と定義する．

終端記号 ‘$’，‘a’，‘b’ の内部表現値をそれぞれ 0，1，2，非終端記号 ‘S′’，‘S’，‘E’ の内部表現値をそれぞれ 3，4，5 とし，文法 P を登録した LRDA を図 2 に示す．図 2 において，添字を丸で囲んでいる要素が統一状態，菱形で囲んでいる要素が間接状態，四角で囲んでいる要素が還元状態である．また，添字を囲んでいない要素は未使用の要素（以下，未使用要素）であり，Base 値に 0，Check 値に遷移種として用いない値 ϵ $(= |N| + |T|)$ を格納して他の状態と区別する．図 1 の LR 解析表における状態と LRDA の統一状態との対応表を表 1 に示す．図 1 の解析表における状態 1，2 は LRDA の統一状態 2，3 に対応する．提案手法は，図 1 の ACTION(1, ‘b’) = shift(2) に該当するアクションを，LRDA 上の間接状態 8（Base[2]+‘b’）における Base 値に $-(|\mathrm{P}| + 3)$ を格納することで定義し，また，図 1 の ACTION(2, ‘b’) = reduce(5) に該当するアクションを，LRDA 上の還元状態 14（Base[3]+‘b’）における Base 値に -5 を格納することで定義する．

	⓪	1	②	③	4	⑤	⑥	⑦	⑧	⑨	⑩	⑪	12	13	14	⑮	16	17	18	19	20	…	23
Base	1	0	6	12	−1	4	7	−7	−8	16	14	−11	−4	−4	−5	18	−3	−3	−2	−2	0	…	0
Check	ε	ε	a	b	$	S	E	a	b	b	S	E	$	a	b	a	$	a	$	a	ε	…	ε

図 2 LRDA

表 1 図 1 の解析表と LRDA との状態対応表

図 1 の解析表における状態番号	0	1	2	3	4	5	6	7
図 2 の LRDA における状態番号	0	2	3	5	6	10	9	15

3.2 表圧縮法の適用

Aho らの表圧縮法を LRDA に適用し，より効率的な LR 解析表を実現することが考えられる．以下，統一状態におけるデフォルト還元規則を定義する方法，ACTION 関数を共有する方法，各非終端記号によるデフォルトの goto 遷移先を定義する方法を順に説明する．

まず，統一状態におけるデフォルト還元規則を定義する方法について説明する．Bison[7] は状態数分の要素をもつ配列を設けて，各状態におけるデフォルト還元規則を管理する．同様に提案手法においても新たに LRDA と等しいサイズの配列を設けて統一状態におけるデフォルト還元規則を管理することが考えられるが，統一状態に該当する要素以外が未使用になり，記憶効率の面で望ましくない．そこで提案手法では，統一状態 x におけるデフォルト還元規則を，LRDA 上の隣接要素 $x + 1$ の Check 値を用いて管理する．以下，この隣接要素 $x + 1$ を統一状態 x の付属要素と呼ぶ．提案手法はデフォルト還元規則 g が存在する状態に対して付属要素を作成し，付属要素の Check 値に規則番号 I_g を負値で格納することでデフォルト還元規則を LRDA 上に定義する．

しかし，ある状態 x から，内部表現値が連続する遷移種 a と $a + 1$ による遷移が存在する場合，遷移定義式 (2) より，遷移先の状態 t と $t + 1$ が LRDA 上で隣接要素であり，状態 $t + 1$ の存在により状態 t の付属要素を作成することができない．そこで，遷移先の状態が LRDA 上で隣接しないように，LRDA の遷移を式 (6) により定義する．式 (6) と付属要素を用いることで，提案手法は任意の状態に情報を追加することができる．

Vol.2009-NL-192 No.1
2009/7/22

$$\begin{cases} t = \mathrm{Base}[x] + 2a \\ \mathrm{Check}[t] = a \end{cases} \quad (6)$$

次に，ACTION 関数を共有する方法について説明する．式 (2) により遷移を定義するダブル配列において，同じ Base 値をもつ状態は遷移を共有する[11]．LRDA においても，遷移を共有して解析表の記憶領域をさらに抑制することが考えられる．しかし，3.1 節で解説した LRDA における各状態は，ACTION 関数を定義する遷移と，GOTO 関数を定義する遷移を同じ Base 値により定義するため，異なる状態間で ACTION 関数を共有することができない．そこで，節点 x における GOTO 関数を定義する遷移を，節点 x の付属要素 $x+1$ の Base 値を用いて定義する．このように，各状態が ACTION 関数と GOTO 関数を定義するための遷移を，異なる Base 値により定義することで，ACTION 関数を定義するための遷移がすべて同じである状態の遷移を共有することができる．

最後に，各非終端記号によるデフォルトの goto 遷移先を定義する方法について説明する．提案手法では，非終端記号 A によるデフォルトの goto 遷移先 t を，式 (7) により LRDA 上に定義する．ただし，開始規則の左辺 S' による goto 遷移先は存在しないため，式 (7) において，$A \neq$ S'，$|T| < A < |T| + |N|$ である．式 (7) により，提案手法は間接参照せずに直接デフォルトの goto 遷移先へ遷移できる．

$$t = (A - |T|) \times 2 \quad (7)$$

文法 P を登録した提案手法の解析表を図 3 に示す．図 3 において，添字を破線の丸で囲んでいる要素が付属要素である．図 1 の解析表における状態と提案手法の解析表における統一状態との対応表を表 1 に示す．提案手法は，統一状態 7 のデフォルト還元規則 4 を，付属要素 8 の Check 値に負値で格納することで定義する．また，状態 0，5 は同じ Base 値を持ち，ACTION 関数を共有する．状態 2，4 はそれぞれ非終端記号 'S'，'E' によるデフォルトの goto 遷移先である．

	⓪	1	②	3	④	⑤	⑥	⑦	⑧	⑨	10	11	⑫	⑬	⑭	⑮	16	…	22
Base	3	0	11	0	8	3	1	6	0	12	−5	−1	0	0	0	0	0	…	0
Check	ε	ε	S	ε	E	a	ε	b	−4	S	b	$	b	−3	a	−2	ε	…	ε

図 3 提案手法の LR 解析表

表 2 図 1 の解析表と提案手法の解析表との状態対応表

図 1 の解析表における状態番号	0	1	2	3	4	5	6	7
図 3 の提案手法における状態番号	0	5	7	2	4	9	12	14

3.3 ダブル配列を応用した LR 解析表による解析アルゴリズム

本節では，提案手法の解析アルゴリズム（図 4）を述べる．提案手法は，ある状態 x において，終端記号 a による遷移先状態 t が存在するときは状態 t の種類を判別してアクションを決定し，遷移先状態が存在しないときは状態 x の付属要素 $x+1$ に定義したデフォルトのアクションを行う．解析アルゴリズム中で使用するスタックの最上段の状態を n_{sp} とし，shift(t)，reduce(I_g)，goto(t) を以下に定義する．

- **goto(t)** ： スタックに状態 t をプッシュする．
- **shift(t)** ： goto(t) を行い，終端記号 a に次のトークンを読み込む．
- **reduce(I_g)**：規則番号 I_g が開始規則 1 ならば構文受理として解析を終了する．そうでなければ，まず，規則 g の左辺の非終端記号 A と右辺の長さ l を取得し，スタックから l 個の状態をポップする．次に，Base[$n_{sp}+1$] > 0 のとき，遷移先状態 t=Base[$n_{sp}+1$]$+2A$ を求める．Check[t] $\neq A$ のときはデフォルト以外の遷移先状態が定義されていないため，goto($(A - |T|) \times 2$) により非終端記号 A のデフォルトの遷移先状態へ遷移する．Check[t]=A のときは，Base[t] ≥ 0 ならば状態 t が統一状態であるため goto(t) を行い，Base[t] < 0 ならば状態 t が間接状態であるため goto($-$(Base[t] $+ |G|$)) を行う．Base[$n_{sp}+1$] ≤ 0 のときはデフォルト以外の遷移先状態が定義されていないため，goto($(A - |T|) \times 2$) により非終端記号 A のデフォルトの遷移先状態へ遷移する．

図 3 を用いて，入力トークン列 'a''b''a''$' を解析する例を説明する．まず，手順 1 で初期状態 0 をスタックに積む．次に，手順 2-1 でスタックの最上段の状態 0 と終端記号 'a' により遷移先状態 5（Base[0]+2×'a' = 5）を求め，Check[5]='a' で遷移先状態 5 が存在するため，手順 2-2 で状態 5 の種類によりアクションを決定する．Base[5] ≥ 0 で遷移先が統一状態であるため shift(5) を行う．同様に，状態 5 と終端記号 'b' により shift(7) を行う．次に，手順 2-1 でスタックの最上段の状態 7 と終端記号 'a' による遷移先状態 8（Base[7]+2×'a' = 8）の有無を確認する．Check[8]≠'a' で遷移先状態 8 が存在しないため，手順 2-3 で状態 7 の付属要素 8（7+1）を参照してデフォルトのアクションを行う．Check[8]< 0 で付属要素にデフォルト還元規則が存在するため reduce(4) を行う．以下，同様に shift(14)，reduce(2)，reduce(1) を順に実行し，開始規則を還元して解析を終了する．

Vol.2009-NL-192 No.1
2009/7/22

解析アルゴリズム
引数：解析表を構成する配列 Base と Check，入力トークン列 $X = a_1 a_2 \cdots a_n\$$

手順 1：初期設定
スタックに初期状態 0 をプッシュし，終端記号 a に最初のトークンを読み込む．
手順 2：スタック最上段の状態 n_{sp} と終端記号 a によるアクションの決定
開始規則を還元するか，構文エラーを発見するまで以下の処理を繰り返す．
手順 2-1：遷移先状態 t の有無の確認
遷移先状態 t（Base$[n_{sp}]+2a$）を計算し，Check$[t]$=a で遷移先状態 t が存在するときは手順 2-2，Check$[t] \neq a$ で遷移先状態 t が存在しないときは手順 2-3 を行う．
手順 2-2：遷移先状態 t の種類によるアクションの決定
Base$[t] \geq 0$ ならば状態 t が統一状態であるため shift(t)，Base$[t] < -|G|$ ならば状態 t が間接状態であるため shift$(-(\text{Base}[t]+|G|))$ を行う．$-|G| \leq \text{Base}[t] < 0$ ならば状態 t が還元状態であるため reduce$(-\text{Base}[t])$ を行う．
手順 2-3：付属要素 $n_{sp}+1$ によるデフォルトのアクション
Check$[n_{sp}+1] < 0$ ならば付属要素にデフォルトの還元規則が存在するため，reduce$(-\text{Check}[n_{sp}+1])$ を行う．そうでなければ構文エラーとして解析を終了する．

図 4 提案手法の解析アルゴリズム

4. 評価実験

本章では，提案手法の解析時間と記憶領域，解析表の構築時間について，Bison[7](version 2.3) を比較手法として評価する．以下，各手法における goto 遷移の計算量について説明した後，各手法を実験により評価する．

提案手法の解析表には，従来の LR 解析表における状態に 1 対 1 対応する統一状態と，統一状態へのポインタを保持する間接状態が存在する．提案手法における goto 遷移は統一状態から次の統一状態までの状態遷移である．以下，統一状態から直接に統一状態へ遷移することを直接遷移，統一状態から間接状態のポインタを参照して統一状態へ遷移することを間接遷移と呼ぶ．直接遷移は間接遷移と比較して，間接参照する必要がない分，goto 遷移に要する計算量を抑制できる．

表 3 実験で用いた文法と解析表の構築時間

	規則数	終端記号数	非終端記号数	構築時間	変換時間
ANSI C	238	90	69	0.046	0.002
Pascal	254	135	67	0.040	0.002

一方，Bison におけるすべての goto 遷移は，遷移先へのポインタを参照する間接遷移である．よって，直接遷移により goto 遷移の総計算量を抑制できる提案手法は，Bison よりも解析時間を抑制できるといえる．

Bison(version 2.3) との比較実験を Intel core2 Duo 2.93GHz，Fedora8 上で行った．実験では，ANSI C と Pascal の文法を使用して LR 解析表を作成し，C 言語のトークン列 6.6M と Pascal のトークン列 6.3M をそれぞれ解析した．本実験において提案手法の解析表は，Bison が出力した解析表を提案手法の定義式に従って変換することで作成した．

表 3 に，各文法における規則数，文法中の終端・非終端記号の数，Bison が解析表を構築する時間（表中，構築時間），提案手法の解析表への変換時間（表中，変換時間）を示す．表 4 に，解析時間，シフト回数，還元回数，直接遷移回数，間接遷移回数，各配列上の未使用要素を除く使用要素数の合計（表中，使用要素数），未使用要素数の合計（表中，未使用要素数），各配列サイズの合計,（表中，配列サイズ合計），記憶領域を示す．ここで，解析時間はトークン列を構文解析する時間とし，解析表の構築や入力トークン列の読み込みに要する時間を含めない．また，記憶領域を各配列サイズの合計と配列 1 要素あたりの記憶領域（2byte）との積とする．

まず，解析時間について評価する．表 4 に示すように，Bison が goto 遷移を行う際に必ず間接遷移するのに対し，提案手法は，大半の goto 遷移を遷移計算量が比較的少ない直接遷移で行う．これにより，提案手法は，解析時間を Bison に対して，ANSI C の文法で 15.66%，Pascal の文法で 15.15%削減した．

次に，記憶領域について評価する．表 4 に示すように，Bison の使用要素数に対して，提案手法は，ANSI C の文法で 3.75%，Pascal の文法で 3.88%削減した．これは，Bison がデフォルト還元規則を状態数分の要素をもつ配列により管理するのに対し，提案手法は付属要素を用いて必要に応じた要素数でデフォルト還元規則を定義できるためである．しかし，提案手法は配列上に存在する未使用要素の数が Bison より増加し，配列サイズの合計が Bison に対して，ANSI C の文法でほぼ同等，Pascal の文法で 1.33 倍になった．これは，付属要素を LRDA 上に追加したことで，各状態における Base 値の影響範囲[11] が拡大

Vol.2009-NL-192 No.1
2009/7/22

表 4 解析実験結果

	ANSI C		Pascal	
	提案	Bison	提案	Bison
解析時間 (s)	1.67	1.98	0.84	0.99
シフト回数 (M)	6.56	6.56	6.32	6.32
還元回数 (M)	53.27	53.27	21.07	21.07
直接遷移回数 (M)	53.24	0	23.36	0
間接遷移回数 (M)	6.59	59.83	4.03	27.39
使用要素数 (K)	3.08	3.20	1.98	2.06
未使用要素数 (K)	0.99	0.86	0.80	0.03
配列サイズ合計 (K)	4.08	4.06	2.78	2.09
記憶領域 (KB)	8.15	8.12	5.55	4.19

し，各状態のダブル配列上における位置を決定する問題[12] が複雑になったためと考えられる．この未使用要素が増加する問題は，提案手法の解析表を作成する際に，ダブル配列に対してガベージコレクション[12] を行うことで緩和できると考えられる．

最後に構築時間について評価する．表 3 に示すように，Bison の解析表から提案手法の解析表への変換に要した時間は，ANSI C と Pascal 共に 0.002 秒であり，Bison が解析表を構築する時間に対して約 1/20 の時間であった．これにより，提案手法は十分実用的な時間で解析表を作成できるといえる．

5. おわりに

本論文では，ダブル配列の遷移を拡張し，従来手法よりも高速に状態遷移できる LR 解析表の実現法を提案した．実験の結果，提案手法は Bison に対して構文解析に要する時間を約 15%削減した．

今後の課題として，ダブル配列にガベージコレクションを実装して記憶領域を抑制することや，提案手法の解析表を文法から直接構築することが挙げられる．また，LRDA を一般化 LR 文法[2] に対応させ，自然言語の文法[4] などのより大きな文法について検証し，提案手法の応用範囲を拡げることが挙げられる．

参 考 文 献

1) Aho, A.V., Sethi, R. and Ullman, J.D.: コンパイラ I 原理・技法・ツール，サイエンス社 (1990).
2) 江里口善生，木谷　強：冨田一般化 LR パーザを用いた情報抽出，情報処理学会論文誌， Vol.38, No.1, pp.44–54 (1997).
3) Shishibori, M., Lee, S.S., Oono, M. and Aoe, J.: Improvement of the LR parsing table and its application to grammatical error correction, *Information Sciences*, Vol.148, pp.11–26 (2002).
4) 田中穂積，野呂智哉，植木正裕：GLR をベースにした自然言語処理用 MSLR パーザの改良，情報処理学会研究報告, NL-182， No.113, pp.9–14 (2007).
5) 秋葉友良，伊藤克亘：LR 表縮退法の提案と自然言語処理および音声認識への応用，信学技報, NLC101-40，pp.21–28 (2001).
6) Johnson, S.C.: YACC: Yet Another Compiler-Compiler, *Computing Sience Technical Report 32, AT&T Bell Laboratories, Murray Hill, N.J.* (1975).
7) Free Software Fundaction: Bison - GNU parser generator, http//www.gnu.org/software/bison/.
8) Aoe, J.: An Efficient Digital Search Algorithm by Using a Double-Array Structure, *IEEE Transactions on Software Engineering*, Vol.15, No.9, pp.1066–1077 (1989).
9) 青江順一：ダブル配列による有限状態機械の効率的インプリメンテーション，電子情報通信学会論文誌 D， Vol.J70-D, No.4, pp.653–662 (1987).
10) Yata, S., Oono, M., Morita, K., Fuketa, M., Sumitomo, T. and Aoe, J.: A compact static double-array keeping character codes, *Information Processing and Management*, Vol.43, No.1, pp.237–247 (2007).
11) 蔵満琢麻，松浦寛生，望月久稔：遷移先関数を拡張した効率的なパターン照合機械の設計と実装，電子情報通信学会論文誌 D， Vol.J92-D, No.3, pp.321–337 (2009).
12) 矢田　普，大野将樹，森田和宏，泓田正雄，吉成友子，青江順一：接頭辞ダブル配列における空間効率を低下させないキー削除法，情報処理学会論文誌， Vol.47, No.6, pp. 1894–1902 (2006).