正規右辺文法のLALRパーサの新しい実現法

森本真一†

本稿では，正規右辺文法に対するLALR解析方法として従来よりも簡単な方法を述べる．本方式ではスタック競合への対応として，スタックシフト時に対応する構文規則の記号をスタックにプッシュするだけであり，従来の方法のようにlook back状態の算出やスタックの要素に関するカウンタ操作が不要である．本稿では，まず本方式の内容を必要な定義とともに述べ，次に本方式での動作を特徴的な場合を例として説明する．さらに本方式の課題である還元時の動作の複雑さを減少させた方式を述べる．

Yet Another Generation of LALR Parsers for Regular Right Part Grammars

SHIN-ICHI MORIMOTO†

A simple method for building LALR parsers for regular right part grammars is given. No grammar transformation is required. No extra data structures such as counters are required. At stack shift states (states in which a parser reads the first symbol of some production rule), the parser pushs a symbol that corresponds to the production rule. At reduce state the parser pops to the symbol that corresponds to the reduced production rule. A method to simplify the action in the reduce states is also given.

1. はじめに

本稿では正規右辺文法に対する新しいLR構文解析法を述べる．正規右辺文法とは，文脈自由文法の構文規則の右辺の記述に構文記号の正規表現を許したものである．正規右辺文法に対するLR構文解析では還元時の動作が問題になる．つまり正規右辺文法では通常の文脈自由文法と異なり構文規則の右辺の長さが定まらないため還元時にスタックのどの位置までポップするかが問題になる．

本稿の方法は，正規右辺文法から（他の等価の文法に変換しないで）直接LRパーサを生成するが，この方法と同様にパーサを直接生成する方法として従来提案されている方法[1)~4)]では，この問題に対して

(a) lookback状態という状態までポップする[1)]

(b) ポップする要素数に関するカウンタを設けカウンタの値に基づいてポップする[2),3)]

(c) path numberに関するデータ構造を設けそのデータの値に基づいてポップする[4)]

というやりかたで対処している．

本稿の方法では，スタックシフト時に構文規則に対応する記号をスタックにプッシュし，還元時に還元すべき構文規則の記号の位置までポップする．この方法は(a)と異なりlookback状態の算出が不要であり，(b)や(c)と異なりカウンタ等の特別なデータ構造も不要である．

本稿はこの方法の課題とそれに対応した方法も述べる．

2. 術語の定義

ここでは以後の議論に必要な定義を行う．本稿では，正規右辺文法の構文規則の右辺は正規表現に対応する有限状態オートマトン（FSA）で表現されているとする．

定義 2.1（正規右辺文法） 正規右辺文法Gは，7つ組 $(V_N, V_T, S, Q, \partial, F, P)$ で定義される．ただし

V_N は非終端記号の集合，V_T は終端記号の集合

S は開始記号

Q は構文規則右辺のFSAの状態の集合

∂ は構文規則右辺のFSAの状態遷移関数

$(\partial : Q \times V \to Q \quad \text{ただし } V = V_N \cup V_T)$.

F は構文規則右辺のFSAの終端状態の集合

P は構文規則の集合

† 日本電気マイコンソフト開発環境研究所
NEC Microcomputer Software Engineering Laboratories

$p \in P$ の左辺の記号を $A \in V_N$，p の右辺の FSA の初期状態を $q \in Q$ とするとき，p を (A, q) と表すことがある．

定義 2.2（↓） p, $q \in Q$ に対して，

$p \downarrow q$ iff $\exists A$ s.t. $\partial(p, A) \in Q$, $(A, q) \in P$.

注：↓ の reflexive transitive closure を $\downarrow^*$ で表し，↓ の nonreflexive transitive closure を $\downarrow^+$ で表す．

定義 2.3（**closure**） $R \subseteq Q$ に対して，

$\text{closure}(R) = \{q \mid \exists p \in R \text{ s.t. } p \downarrow^* q\}$.

定義 2.4（**LR オートマトン**） 右辺正規文法 G に対する LR オートマトンは，以下の初期状態 $\mathbf{q}_0$ と，遷移関数 Next を持つ．

$\mathbf{q}_0 = \text{closure}(\{q \mid \exists (S \to q) \in P\})$

$\text{Next}(\mathbf{q}, X) = \text{closure}(\text{succ}(\mathbf{q}, X))$

ただし，$\text{succ}(\mathbf{q}, X) = \{\partial(q, X) \mid q \in \mathbf{q}\}$.

注：LR オートマトンの状態は太字で表し，LR オートマトンの状態の集合を $\mathbf{Q}$ で表す．

定義 2.5（**kernel, nonkernel**） $\mathbf{q} \in \mathbf{Q}$ に対して，$\text{kernel}(\mathbf{q})$, $\text{nonkernel}(\mathbf{q})$ を次のように定める．

$\text{kernel}(\mathbf{q}_0) = \phi$, $\text{nonkernel}(\mathbf{q}_0) = \mathbf{q}_0$

$\text{kernel}(\text{Next}(\mathbf{q}, X)) = \text{succ}(\mathbf{q}, X)$

$\text{nonkernel}(\mathbf{q}) = \{q \mid \exists p \in \text{kernel}(\mathbf{q}) \text{ s.t. } p \downarrow^+ q\}$.

定義 2.6（**ELALR (1) 文法**） 次の条件を満たす正規右辺文法を ELALR (1) 文法という[1]．

(1) LR オートマトンにおける競合は lookahead 集合により解決できる．

(2) 還元時のハンドルは一意に決定できる．

本稿では，ELALR (1) 文法を対象とする．

定義 2.7（→） $q_1 \in \mathbf{q}_1$, $q_2 \in \mathbf{q}_2$ に対して，$\mathbf{q}_2 = \text{Next}(\mathbf{q}_1, X)$, $q_2 = \partial(q_1, X)$ が成り立つとき，$(\mathbf{q}_1, q_1) \to^X (\mathbf{q}_2, q_2)$ とする．

$q_1 \in \text{kernel}(\mathbf{q}_1)$ の場合は

$(\mathbf{q}_1, q_1) \to^{X/K} (\mathbf{q}_2, q_2)$ または $\mathbf{q}_1 \to^{X/K} \mathbf{q}_2$

$q_1 \in \text{nonkernel}(\mathbf{q}_1)$ の場合は

$(\mathbf{q}_1, q_1) \to^{X/N} (\mathbf{q}_2, q_2)$ または $\mathbf{q}_1 \to^{X/N} \mathbf{q}_2$

と表す．

本方式ではスタックに，LR オートマトンの状態，構文規則の集合，構文記号をこの順序でプッシュする場合と構文記号のみをプッシュする場合がある．またスタックをポップする場合は LR オートマトンの状態までポップする．そこでスタックの内容は，$\mathbf{q}_0 P_0 \beta_0 \cdots \mathbf{q}_n P_n \beta_n$ と表すことができる（$\mathbf{q}_i \in \mathbf{Q}$, $P_i \subseteq P$, $\beta_i \in V^*$）．

注：e_1 が e_2 よりスタックトップに近い位置にプッシュされているとき，e_1 は e_2 より上にあるという．

定義 2.8（**状況**） 構文解析系の状況は，スタックの内容，現在の状態，現在の入力列から構成される 3 つ組である．つまり状況は，

$(\mathbf{q}_0 P_0 \beta_0 \cdots \mathbf{q}_n P_n \beta_n,\ \mathbf{q}, z)$

と表せる．ただし，

$\mathbf{q}_i \in \mathbf{Q}$, $P_i \subseteq P$, $\beta_i \in V^*$ $(0 \le i \le n)$

$\mathbf{q} \in \mathbf{Q}$, $z \in V \times V_T^*$

定義 2.9（**パーサの動作**） 状況が，$(\mathbf{q}_0\ P_0\ \beta_0\ \cdots\ \mathbf{q}_n P_n \beta_n,\ \mathbf{q}, Xz)$ の場合に，パーサは現在の状態 $\mathbf{q}$ と入力記号 X に応じて，次の 3 つの動作のいずれかを行う．

各動作を行う条件と動作後の状況は，次のとおりである．

(1) シフト (X) （$\mathbf{q} \to^{X/K} \mathbf{q}'$ の場合）

$(\mathbf{q}_0 P_0 \beta_0 \cdots \mathbf{q}_n P_n \beta_n X, \mathbf{q}', z)$

(2) スタックシフト (X) （$\mathbf{q} \to^{X/N} \mathbf{q}'$ の場合）

$(\mathbf{q}_0 P_0 \beta_0 \cdots \mathbf{q}_n P_n \beta_n \mathbf{q} P_{n+1} X, \mathbf{q}', z)$

ただし，

$P_{n+1} = \{(A, q) \mid \exists q' \in \mathbf{q}' \text{s.t.} (\mathbf{q}, q) \to^{X/N} (\mathbf{q}', q')\}$

(3) 還元（$\mathbf{q}$ は，構文規則 (A, q'') の終端状態を含み，$X \in \text{lookahead}((A, q''), \mathbf{q})$ の場合）

$(\mathbf{q}_0 P_0 \beta_0 \cdots \mathbf{q}_k P_k \beta_k, \mathbf{q}_{k+1}, AXz)$

ただし，$\beta_{k+1} \ldots \beta_n$ は (A, q'') のハンドル，P_{k+1} は (A, q'') を含み最も上にあるもの

(1) と (2) の条件がともに成立する場合をスタック競合という．特に (2) の P_{n+1} の要素を p とするとき p に関するスタック競合という場合もある．スタック競合の場合は，(2) の動作を行うとする．

本方式ではシフト時とスタックシフト時の LR オートマトンの動作は定義 2.6 の (1) の条件から一意に決まり，還元時にスタックを戻す位置も (2) の条件からハンドルが一意に決まるので決定できる．このため，本方式は ELALR (1) 文法に対して正しく解析できる．

3. 本方式による解析例

ここでは，**図 1** で表される文法 G1 に対する本方式による解析例を示す．

G1 に対する LR オートマトンは，**図 2** で表される[4]．図 2 で | の左側の数字は kernel の要素を，右側の数字は nonkernel の要素を表す．また ss (x) はスタックシフト (x) を，s (x) はシフト (x) を表す．状態 **3** や状態 **4** で入力が c の場合に（#1 に関する）スタッ

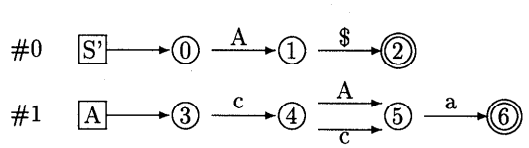


図 1 正規右辺文法 G1 の構文規則

Fig. 1 Representation of regular right part grammar G1.

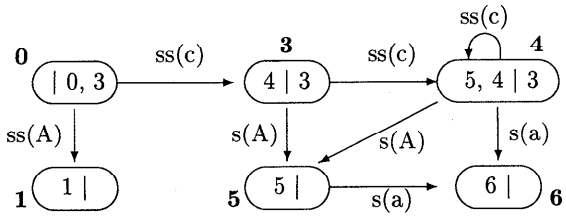


図2 文法G1のLRオートマトン
Fig. 2 LR automaton for G1.

スタック	状態	入力	動作
0{1}c3{1}c	4	caa$	スタックシフト (c)
0{1}c3{1}c4{1}c	4	aa$	シフト (a)
0{1}c3{1}c4{1}ca	6	a$	#1(A:cca) で還元
0{1}c	3	Aa$	シフト (A)
0{1}cA	5	a$	シフト (a)
0{1}cAa	6	$	#1(A:cAa) で還元
	0	A$	

図3 cccaaに対する解析動作
Fig. 3 Parse process for cccaa.

ク競合が生じる.

G1の記号列 cccaa に対する本方式による解析（の一部）を**図3**に示す. 図3で {1} は {#1} を表す.

4. 本方式の課題

本方式の課題は, 還元時の動作（スタックを戻す位置の決定）が複雑なことである. 本方式では, 還元される構文規則を p とすると

(1) p をプッシュしている
(2) それより上にプッシュされている記号列が p の右辺のFSAで受理される

という条件を満たし最も上にある状態の位置を決定する必要がある. たとえば, G1 の cccaa に対する解析で最初に状態**6**で#1による還元を行う場合を考える. この場合では#1がプッシュされた状態のうち最も上にある状態は状態**4**であるが, 状態**4**より上でプッシュされている記号列 ca は, #1の右辺のFSAで受理されない. #1がプッシュされた状態で状態**4**の次に上にある状態は状態**3**であり, 状態**3**より上でプッシュされている記号列 cca は, #1の右辺のFSAで受理される. そこでスタックは状態**3**の位置までポップすればよいことが分かる.

この動作は他の動作（スタックに要素をプッシュする等）に比べると複雑である. そこで次章では, 還元時にスタックを戻す位置の決定がより簡単な方式を述べる.

5. 改良方式

本章ではスタックを戻す位置の決定がより簡単な方式を述べる. この方式の基本的な考え方は次のとおりである.

還元時にスタックを戻す位置を決定するためには, 還元すべき構文規則を p とするとき, スタックのトップから何番目の p までポップするか（何個の p を読みとばすか）が分かればよい. 読みとばされる p は, ハンドルの解析の途中でスタック競合状態のスタックシフト動作でプッシュされたものである. つまり p で還元されるときにスタックを戻す位置を決定するには, ハンドルを解析中に発生した p に関するスタック競合の数が分かればよい.

p のハンドルを $v_1 \ldots v_n$ とするとき, p の右辺のFSAで $v_1 \ldots v_n$ に対応する状態列をハンドルのパス, p のハンドルの解析中に発生する p に関するスタック競合を自己競合というとする.

ハンドルのパスが自己競合を発生するループ（これを自己競合ループという）を含まない場合は, ハンドル解析中に発生する自己競合の数をLRオートマトンの作成時に決定できる. ハンドルのパスが自己競合ループを含む場合は, ハンドル解析中に発生する自己競合の数は（ハンドル解析時にループをまわる回数に依存するため）LRオートマトンの作成時に決定できない. しかしループを1回まわる間に発生する自己競合の数はLRオートマトンの作成時に一意に定まる.

そこで

- ハンドルのパス中のループでない部分で発生した自己競合の数をLRオートマトンの状態に保持する
- ハンドルのパス中のループをまわるごとに, そこで発生した自己競合の数に対応する情報をスタックにプッシュする

ことにより, ハンドルの解析中に発生した自己競合の数を, ループでない部分で発生した自己競合の数（LRオートマトンの状態から求めることができる）と, ループで発生した自己競合の数（スタックにプッシュされた情報から求めることができる）の和として求めることができる.

ハンドルのパス中のループでない部分で発生した自己競合の数をLRオートマトンの状態に保持するために, 以下で述べる方式では, 構文規則右辺のFSAの状態の列とその状態列に対応するハンドルの解析中に発生した自己競合の数の対の集合をLRオートマトンの状態とする.

以下に, この方法を必要な定義とともに述べる.

定義 5.1（$Q^{\dagger}$） 構文規則右辺のFSAの状態の空でない列の集合を Q^{+} で表す．$q \in Q^{+}$ に対して，q の長さ（要素の個数）を $\mathrm{len}(q)$ で表し，q の i 番目の要素を q^{i} で表す．つまり $q = q^{1} \cdots q^{\mathrm{len}(q)}$ とおける．また q の末尾の要素 $q^{\mathrm{len}(q)}$ を $\mathrm{tail}(q)$ で表す．$\mathrm{tail}(q)$ を右辺に持つ構文規則を q の構文規則という．

定義 5.2（$\Downarrow$） $p \in Q^{+}$，$q \in Q$ に対して，

$p \Downarrow q$ iff $\exists A$ s.t. $\partial(\mathrm{tail}(p), A) \in Q, (A, q) \in P$.

注：$\Downarrow$ の reflexive transitive closure を $\Downarrow^{*}$ で表し，$\Downarrow$ の nonreflexive transitive closure を $\Downarrow^{+}$ で表す．

定義 5.3（$\overline{\mathrm{closure}}$） $R \subseteq Q^{+}$ に対して，

$\overline{\mathrm{closure}}(R) = \{q \mid \exists p \in R \text{ s.t. } p \Downarrow^{*} q\}$.

定義 5.4（**オートマトン**） 以下の初期状態 $\overline{\mathbf{q}_0}$ と，遷移関数 $\overline{\mathrm{Next}}$ を持つオートマトンを考える．

$\overline{\mathbf{q}_0} = \overline{\mathrm{closure}}(\{q \mid \exists (S \to q) \in P\})$

$\overline{\mathrm{Next}}(\overline{\mathbf{q}}, X) = \overline{\mathrm{closure}}(\overline{\mathrm{succ}}(\overline{\mathbf{q}}, X))$

ただし，

$\overline{\mathrm{succ}}(\overline{\mathbf{q}}, X) = \{\overline{\mathrm{appnd}}(q, \partial(\mathrm{tail}(q), X)) \mid q \in \overline{\mathbf{q}}\}$

$\overline{\mathrm{appnd}}(q, r)$

$= qr$ （$r \neq q^{i}\ (1 \leq i \leq \mathrm{len}(q))$ の場合）

$= q^{1} \cdots q^{l}$ （$r = q^{l}$ の場合）

注：定義5.4のオートマトンの状態の集合を $\overline{\mathbf{Q}}$ で表す．

定義 5.5（**kernel, nonkernel**） $\overline{\mathbf{q}} \in \overline{\mathbf{Q}}$ に対して，$\mathrm{kernel}(\overline{\mathbf{q}})$，$\mathrm{nonkernel}(\overline{\mathbf{q}})$ を次のように定める．

$\mathrm{kernel}(\overline{\mathbf{q}_0}) = \phi$ $\quad \mathrm{nonkernel}(\overline{\mathbf{q}_0}) = \overline{\mathbf{q}_0}$

$\mathrm{kernel}(\overline{\mathrm{Next}}(\overline{\mathbf{q}}, X)) = \overline{\mathrm{succ}}(\overline{\mathbf{q}}, X)$

$\mathrm{nonkernel}(\overline{\mathbf{q}}) = \{q \mid \exists p \in \mathrm{kernel}(\overline{\mathbf{q}}) \text{ s.t. } p \Downarrow^{+} q\}$.

定義5.4のオートマトンに対して，定義2.7と同じく $\to$，$\to^{X}$，$\to^{X/K}$，$\to^{X/N}$ を定義する．また2章と同じくスタック競合を定義する．

定義 5.6（**ループの末尾**） $\overline{\mathbf{q}_i} \in \overline{\mathbf{Q}}$，$q_i \in \overline{\mathbf{q}_i}$，$X_i \in V\ (1 \leq i \leq n)$ が

- $(\overline{\mathbf{q}_1}, q_1) \to^{X_1/N} (\overline{\mathbf{q}_2}, q_2)$
- $(\overline{\mathbf{q}_i}, q_i) \to^{X_i/K} (\overline{\mathbf{q}_{i+1}}, q_{i+1})\ (1 < i < n)$
- $\exists l (1 < l < n)$ s.t. $(\overline{\mathbf{q}_n}, q_n) \to^{X_n/K} (\overline{\mathbf{q}_l}, q_l)$

を満たすとき，$(\overline{\mathbf{q}_i}, q_i, X_i)\ (l \leq i \leq n)$ はループをなすといい，$(\overline{\mathbf{q}_n}, q_n, X_n)$ や $(\overline{\mathbf{q}_n}, X_n)$ をループの末尾という．

定義5.4のオートマトンから，本方式のLRオートマトンを定義する．

定義 5.7（**LR オートマトン**） 本方式のLRオートマトンは，以下の初期状態 $\mathbf{q}_0$ と，遷移関数 Next を持つ．

$\mathbf{q}_0 = \{\langle q, 0\rangle \mid q \in \overline{\mathbf{q}_0}\}$

$\mathrm{Next}(\mathbf{q}, X) = \{\langle q', i\rangle \mid q' \in \overline{\mathrm{Next}}(\mathrm{Item}(\mathbf{q}), X)\}$

ただし，$\mathrm{Item}(\mathbf{q}) = \{q \mid \langle q, j\rangle \in \mathbf{q}\}$

$\langle q', i\rangle$ の i は，次のように定義される．

(1) $q' \notin \overline{\mathrm{succ}}(\mathrm{Item}(\mathbf{q}), X)$ の場合：

$i = 0$

(2) 次の条件を満たす $\langle q, j\rangle \in \mathbf{q}$ が存在する場合：

- $q' = \overline{\mathrm{appnd}}(q, \partial(\mathrm{tail}(q), X))$
- $(\mathrm{Item}(\mathbf{q}), q, X)$ はループの末尾でない
- $\mathrm{Item}(\mathbf{q})$ は5.4のオートマトンで入力が X の場合に q の構文規則に関するスタック競合が生じない

$i = j$

(3) 次の条件を満たす $\langle q, j\rangle \in \mathbf{q}$ が存在する場合：

- $q' = \overline{\mathrm{appnd}}(q, \partial(\mathrm{tail}(q), X))$
- $(\mathrm{Item}(\mathbf{q}), q, X)$ はループの末尾でない
- $\mathrm{Item}(\mathbf{q})$ は5.4のオートマトンで入力が X の場合に q の構文規則に関するスタック競合が生じる

$i = j + 1$

(4) 次の条件を満たす $\langle q, j\rangle \in \mathbf{q}$，$\mathbf{q}'$，$\langle q', j'\rangle \in \mathbf{q}'$ が存在する場合：

- $(\mathrm{Item}(\mathbf{q}), q) \to^{X} (\mathrm{Item}(\mathbf{q}'), q')$
- $q' = \overline{\mathrm{appnd}}(q, \partial(\mathrm{tail}(q), X))$
- $(\mathrm{Item}(\mathbf{q}), q, X)$ はループの末尾である

$i = j'$

注：LRオートマトンの状態は太字で表し，LRオートマトンの状態の集合を $\mathbf{Q}$ で表す．

注：LRオートマトンの状態の要素 $\langle q, j\rangle$ の j は，q の解析中に発生する自己競合の数を表す．

定義 5.8（**先頭状態**） $\mathbf{q}_i \in \mathbf{Q}$，$q_i \in \mathbf{q}_i$，$X_i \in V\ (1 \leq i \leq n)$ が

- $(\mathrm{Item}(\mathbf{q}_1), q_1) \to^{X_1/N} (\mathrm{Item}(\mathbf{q}_2), q_2)$
- $(\mathrm{Item}(\mathbf{q}_i), q_i) \to^{X_i/K} (\mathrm{Item}(\mathbf{q}_{i+1}), q_{i+1})\ (1 < i < n)$

を満たすとき，$\mathbf{q}_1$ を $\mathbf{q}_i$ の q_i に関する先頭状態という．

注：$\mathbf{q} \in \mathbf{Q}$，$q \in \mathbf{q}$ が $q \in F$ を満たす場合は文献1)で定義されたlookback状態（$\mathrm{LB}(\mathbf{q}, q)$）に対して

$\mathrm{LB}(\mathbf{q}, q) = \{\mathbf{q}' \mid \mathbf{q}'$ は $\mathbf{q}$ の q に関する先頭状態$\}$

が成り立つ．

定義 5.9（**ループ内競合，ループ内生成列**） $\langle q_i, j_i\rangle \in \mathbf{q}_i$，$X_i \in V (1 \leq i \leq n)$，$p \in P$ が

- $(\mathrm{Item}(\mathbf{q}_i), q_i, X_i)\ (1 \leq i \leq n)$ はループをなす
- $(\mathrm{Item}(\mathbf{q}_n), q_n, X_n)$ はループの末尾
- $\mathbf{q}_1$ から $X_1 \cdots X_n$ の解析中に p に関する自己競合が発生する

を満たすとき，$(p, j_n - j_1 + j)$ を $\mathbf{q}_n$ と X_n におけるループ内競合という．

ただし,

$j=0$ (Item$(\mathbf{q}_n)$ は入力が X_n で p に関するスタック競合が発生しない場合),

$j=1$ (Item$(\mathbf{q}_n)$ は入力が X_n で p に関するスタック競合が発生する場合).

このとき (Item$(\mathbf{q}_n), q_n, X_n$) や (Item$(\mathbf{q}_n), X_n$) を自己競合ループの末尾という.

さらに, $q'_1 \in \mathbf{q}_1$, $q'_n \in \mathbf{q}_n$ が,

- (Item$(\mathbf{q}_n), q'_n) \rightarrow^{X_n/K}$ (Item$(\mathbf{q}_1), q'_1$)
- $\mathbf{q}_1$ の q'_1 に関する先頭状態は $\mathbf{q}_i$ $(1 \leq i \leq n)$ のいずれかである

を満たすとき, q'_n の構文規則を p' とすると q'_n をループ内生成列 (または p' に関するループ内生成列) という.

注:定義5.9の $\mathbf{q}_n$ と X_n におけるループ内競合 (p, j) の j は, $\mathbf{q}_1$ から $X_1 \cdots X_n$ の解析に対応するループで発生する自己競合の数を表す.

この方式ではスタックに, LR オートマトンの状態, 構文規則の集合, 構文記号の他にループ内競合の集合もプッシュするので, スタックの内容は $(\mathbf{q}_0\overline{P}_0P_0\beta_0\mathbf{q}_1 \cdots \mathbf{q}_n\overline{P}_nP_n\beta_n)$ と表すことができる ($\mathbf{q}_i \in \mathbf{Q}$, $\overline{P}_i$ はループ内競合の集合, $P_i \subseteq P$, $\beta_i \in V^*$).

定義 5.10 (状況) 構文解析系の状況は, スタックの内容, 現在の状態, 現在の入力列から構成される3つ組である. つまり状況は,

$$(\mathbf{q}_0\overline{P}_0P_0\beta_0 \cdots \mathbf{q}_n\overline{P}_nP_n\beta_n,\ \mathbf{q},\ z)$$

と表せる.

ただし, $\mathbf{q}_i \in \mathbf{Q}$, $\overline{P}_i$ はループ内競合の集合, $P_i \subseteq P$, $\beta_i \in V^*$ $(0 \leq i \leq n)$, $\mathbf{q} \in \mathbf{Q}$, $z \in V \times V_T^*$

定義 5.11 ($D_p(j)$, $C_p(j)$) 状況 $(\mathbf{q}_0\overline{P}_0P_0\beta_0 \cdots \mathbf{q}_n\overline{P}_nP_n\beta_n, \mathbf{q}, z)$ が, $\exists(q, m) \in \mathbf{q}_n$ s.t. tail(q) は構文規則 p の終端状態を満たすとき, この状況に対する $D_p(j)$, $C_p(j)$, $Sig_p(j)$ $(0 \leq j \leq n)$ を次のように定義する.

$D_p(n+1) = m$

$D_p(j) = D_p(j+1) + C_p(j) - Sig_p(j)$

$C_p(j) = l$

($\exists(p, l) \in \overline{P}_j$ で, a) または b) が成り立つ)

a) Item$(\mathbf{q}_j)$ は p に関するループ内生成列を含まない

b) $D_p(j+1) > M_p$ ただし M_p は $\{x | \langle q, x\rangle \in \mathbf{q}_j,\ q$ は p に関するループ内生成列$\}$ の要素 x の最大値

$C_p(j) = 0$ (上の条件が成り立たない場合)

$Sig_p(j) = 1$ ($p \in P_j$ の場合)

$Sig_p(j) = 0$ ($p \notin P_j$ の場合)

注:定義5.11の条件を満たす状況で $D_p(j) \geq 0$ を満たす j に対して, $D_p(j)$ は $\mathbf{q}$ の q に関する先頭状態から $\mathbf{q}_j$ ($j = n+1$ の場合は $\mathbf{q}$) までのLRオートマトンのパスのうちループでない部分で発生した自己競合の数を表す (証明は付録参照).

定義 5.12 (パーサの動作) 状況が, $(\mathbf{q}_0\overline{P}_0P_0\beta_0 \cdots \mathbf{q}_n\overline{P}_nP_n\beta_n, \mathbf{q}, Xz)$ の場合に, パーサは現在の状態 $\mathbf{q}$ と入力記号 X に応じて, 次の5つの動作のいずれかを行う.

各動作を行う条件と動作後の状況は, 次のとおりである.

(1) シフト (X) (Item$(\mathbf{q}) \rightarrow^{X/K}$ Item$(\mathbf{q}')$ で, (Item$(\mathbf{q}), X$) は自己競合ループの末尾でない場合)

$$(\mathbf{q}_0\overline{P}_0P_0\beta_0 \cdots \mathbf{q}_n\overline{P}_nP_n\beta_nX,\ \mathbf{q}',\ z)$$

(2) キャンセル (X) (Item$(\mathbf{q}) \rightarrow^{X/K}$ Item$(\mathbf{q}')$ で, (Item$(\mathbf{q}), X$) は自己競合ループの末尾の場合)

$$(\mathbf{q}_0\overline{P}_0P_0\beta_0 \cdots \mathbf{q}_n\overline{P}_nP_n\beta_n\mathbf{q}\overline{P}_{n+1}\phi X,\ \mathbf{q}',\ z)$$

(3) スタックシフト (X) (Item$(\mathbf{q}) \rightarrow^{X/N}$ Item$(\mathbf{q}')$ で, (Item$(\mathbf{q}), X$) は自己競合ループの末尾でない場合)

$$(\mathbf{q}_0\overline{P}_0P_0\beta_0 \cdots \mathbf{q}_n\overline{P}_nP_n\beta_n\mathbf{q}\phi P_{n+1}X,\ \mathbf{q}',\ z)$$

(4) スタックキャンセル (X) (Item$(\mathbf{q}) \rightarrow^{X/N}$ Item$(\mathbf{q}')$ で, (Item$(\mathbf{q}), X$) は自己競合ループの末尾の場合)

$$(\mathbf{q}_0\overline{P}_0P_0\beta_0 \cdots \mathbf{q}_n\overline{P}_nP_n\beta_n\mathbf{q}\overline{P}_{n+1}P_{n+1}X, \mathbf{q}', z)$$

(5) 還元 ($\exists q \in$ Item$(\mathbf{q})$ s.t. tail(q) は構文規則 (A, q'') の終端状態, $X \in$ lookahead$((A, q''), \mathbf{q})$ の場合)

$$(\mathbf{q}_0\overline{P}_0P_0\beta_0 \cdots \mathbf{q}_k\overline{P}_kP_k\beta_k,\ \mathbf{q}_{k+1},\ AXz)$$

ただし, k は $D_{(A,q'')}(k+1) < 0$ を満たす最大値

ただし, (3), (4) の P_{n+1} は,

$\exists q' \in$ Item$(\mathbf{q}')$ s.t. (Item$(\mathbf{q}), q) \rightarrow^{X/N}$ (Item$(\mathbf{q}'), q'$) を満たす構文規則 (A, q) の集合, (2), (4) の $\overline{P}_{n+1}$ は $\mathbf{q}$ と X におけるループ内競合の集合.

6. 解析例

6.1 ハンドルのパスに自己競合ループがない場合

自己競合ループがハンドルのパスに存在しない文法として図1で定義された文法G1を考える.

G1に対する本方法でのLRオートマトンは**図4**で表される. 図4では各状態の $\langle q, 0\rangle$ の要素を q と表す.

G1の記号列 cccaa に対する本方式による解析 (の一部) を**図5**に示す. 状態**7**で還元するときは状態**7**の要素が $\langle 3456, 1\rangle$ なので, 還元すべき構文規則 (#1) に関するスタック競合 (自己競合) がハンドル解析中

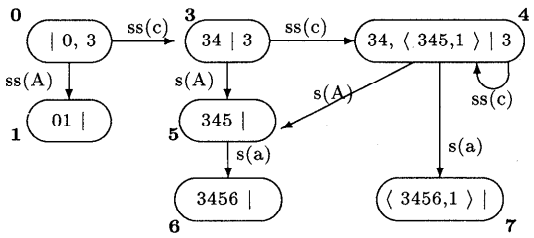


図4 文法G1のLRオートマトン
Fig. 4 LR automaton for G1.

スタック	状態	入力	動作
0ϕ{1}c3ϕ{1}c	4	caa$	スタックシフト (c)
0ϕ{1}c3ϕ{1}c4ϕ{1}c	4	aa$	シフト (a)
0ϕ{1}c3ϕ{1}c4ϕ{1}ca	7	a$	#1(A:cca) で還元
0ϕ{1}c	3	Aa$	シフト (A)
0ϕ{1}cA	5	a$	シフト (a)
0ϕ{1}cAa	6	a$	#1(A:cAa) で還元
	0	A$	

図5 cccaaに対する解析動作
Fig. 5 Parse process for cccaa.

に1回発生したことが分かる．そこで#1をプッシュした上から2番めの状態の位置までスタックを戻す．状態 **6** で還元するときは状態 **6** の要素が $\langle 3456, 0\rangle$ なので，自己競合がハンドル解析中に発生しなかったことが分かる．そこで#1をプッシュした最も上の状態の位置までスタックを戻す．

6.2 ハンドルのパスに自己競合ループがある場合

自己競合ループがハンドルのパスに存在する文法として**図6**で定義された文法G2を考える．

G2に対する本方法でのLRオートマトンは**図7**で表される．図7でc(x)はキャンセル(x)を表す．

状態 **5** でcを読み込んだ場合は自己競合が発生する．次にdを読み込むと，(Item(**5**), 3457, c) と (Item(**6**), 345, d) はループをなすので (**6**, d) は自己競合ループの末尾となる．ゆえに状態 **6** でdを読み込んだ場合はループ内競合 ($\langle$#1, 1$\rangle$) をスタックにプッシュする．G2の記号列cbdcbdcdcaaに対する本方式による解析（の一部）を**図8**に示す．状態 **8** で還元するときは，状態の要素は $\langle 3456, 1\rangle$ だが，スタックに $\langle$#1, 1$\rangle$ がプッシュされ状態 **6** にループ内生成列は含まれないから，自己競合がハンドル解析中に2回発生したことが分かる．そこで#1をプッシュした上から3番めの状態の位置までスタックを戻す．

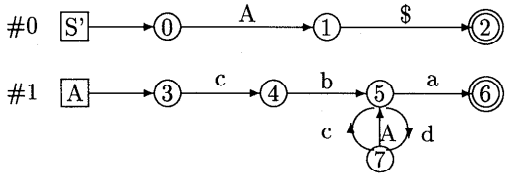


図6 正規右辺文法G2の構文規則
Fig. 6 Representation of regular right part grammar G2.

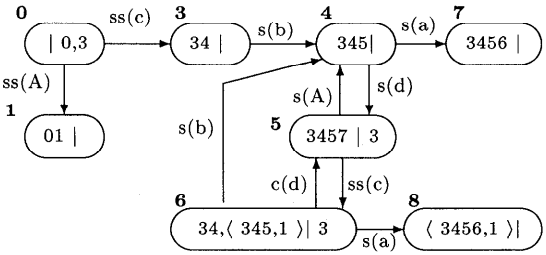


図7 文法G2のLRオートマトン
Fig. 7 LR automaton for G2.

7. まとめ

正規右辺文法のLRパーサの新しい実現法を述べた．

2章の方法は，解析時にカウンタの操作を必要としない，還元時に4章の(2)の条件を満たす最も上にある状態の位置までスタックを戻す必要があるという点で文献1)と類似点がある．しかし文献1)の方法は，lookback状態の算出処理が必要であり，また定義2.6で定義されたELALR(1)文法でも解析できないものがあるという問題がある．たとえば，G1は文献1)の方法では解析できない[4]．これに対して2章の方法はlookback状態の算出処理が不要であり，2章で述べたようにELALR(1)文法全体に適用できる．

5章の方法は還元時のスタックの位置をスタック競合の回数に基づいて決定する点では文献3)と類似点があるが，文献3)の方法と異なり，スタック競合の回数に関するカウンタやその操作が不要である．

参考文献

1) Nakata, I. and Sassa, S.: Generation of Efficient LALR Parsers for Regular Right Part Grammars, *Acta Inf.*, Vol.23, pp.149–162 (1986).
2) 佐々政孝，中田育男：正規右辺文法のLRパーサの簡単な実現法，情報処理学会論文誌，Vol.27, No.1, pp.124–127 (1986).
3) 張 又普，中田育男，佐々政孝：正規右辺文法の効率の良いLRパーサの簡単な実現法，情報処理学会論文誌，Vol.28, No.11, pp.162–167 (1987).

スタック	状態	入力	動作
0ϕ{1}cbd**5**ϕ{1}cbd	**5**	cdcaa$	スタックシフト (c)
0ϕ{1}cbd**5**ϕ{1}cbd**5**ϕ{1}c	**6**	dcaa$	キャンセル (d)
0ϕ{1}cbd**5**ϕ{1}cbd**5**ϕ{1}c**6**{(1,1)}ϕd	**5**	caa$	スタックシフト (c)
0ϕ{1}cbd**5**ϕ{1}cbd**5**ϕ{1}c**6**{(1,1)} ϕd**5**ϕ{1}c	**6**	aa$	シフト (a)
0ϕ{1}cbd**5**ϕ{1}cbd**5**ϕ{1}c**6**{(1,1)} ϕd**5**ϕ{1}ca	**8**	a$	#1(A:cbdcdca) で還元
0ϕ{1}cbd	**5**	Aa$	シフト (A)
0ϕ{1}cbdA	**4**	a$	シフト (a)
0ϕ{1}cbdAa	**7**	$	#1(A:cbdAa) で還元
	0	A$	

図 8 cbdcbdcdcaa に対する解析動作
Fig. 8 Parse process for cbdcbdcdcaa.

4) Zhang, Y. and Nakata, I.: Generation of Path Directed LALR(k) Parsers for Regular Right Part Grammars, *J. Information processing*, Vol.14, pp.325–334 (1991).

付　　録

ここでは，次の命題が成り立つことを示す．

命題 1　定義 5.11 の条件を満たす状況で $D_p(j) \geq 0$ を満たす j に対して，$\mathbf{q}$ の q に関する先頭状態を $\mathbf{q}_I$ とすると，$D_p(j)$ は $\mathbf{q}_I$ から $\mathbf{q}_j$（$j = n+1$ の場合は $\mathbf{q}$）までの LR オートマトンのパスのうちループでない部分で発生した自己競合の数を表す

定義 1（消滅状態）　q_i, $q'_i \in \mathbf{q}_i$, $X_i \in V\,(1 \leq i \leq n)$ が

- $(\mathrm{Item}(\mathbf{q}_1), q_1) \rightarrow^{X_1/K} (\mathrm{Item}(\mathbf{q}_2), q_2)$
- $(\mathrm{Item}(\mathbf{q}_1), q'_1) \rightarrow^{X_1/N} (\mathrm{Item}(\mathbf{q}_2), q'_2)$
- $(\mathrm{Item}(\mathbf{q}_i), q_i) \rightarrow^{X_i} (\mathrm{Item}(\mathbf{q}_{i+1}), q_{i+1})$ $(1 < i < n)$
- $(\mathrm{Item}(\mathbf{q}_i), q'_i) \rightarrow^{X_i} (\mathrm{Item}(\mathbf{q}_{i+1}), q'_{i+1})$ $(1 < i < n-1)$
- $\partial(\mathrm{tail}(q'_{n-1}), X_{n-1})$ は存在しない

を満たすとき，$\mathbf{q}_n$ を $\mathbf{q}_1$ の競合の消滅状態という．

補題 1　$q_i \in \mathbf{q}_i$, $X_i \in V\,(1 \leq i \leq n)$ が

- $(\mathrm{Item}(\mathbf{q}_i), q_i, X_i)$ $(1 \leq i \leq n)$ はループをなす
- $(\mathrm{Item}(\mathbf{q}_n), q_n, X_n)$ はループ末尾
- $\mathbf{q}_k$ は，入力が X_k の場合に自己競合が発生する

を満たすとき，$\mathbf{q}_i\,(1 \leq i \leq n)$ の中に $\mathbf{q}_k$ の競合の消滅状態が存在する．

証）$\mathbf{q}_i\,(1 \leq i \leq n)$ がいずれも消滅状態でないとする．$\mathbf{q}_k$ で発生する自己競合が構文規則 p に関する競合とする．$\mathbf{q}_k$ で $X_kX_{k+1}\cdots X_nX_1\cdots X_{k-1}$ を読み込んだときに遷移する状態列であるループ $\mathbf{q}_k\mathbf{q}_{k+1}\cdots\mathbf{q}_n\mathbf{q}_1\cdots\mathbf{q}_{k-1}$ には仮定より消滅状態が存在しないから，$\beta_1 = X_1\cdots X_{k-1}$, $\beta_2 = X_{k+1}\cdots X_n$ とするとき，$X_k\beta_2\beta_1\beta_3$ と $X_k\beta_2\beta_1X_k\beta_2\beta_1\beta_3$ が p のハンドルであるような構文記号列 β_3 が存在する．ゆえに，p の左辺の構文記号を A とすると，$X_k\beta_2\beta_1X_k\beta_2\beta_1\beta_3$ は，A, $X_k\beta_2\beta_1A$ のいずれにも還元できることになり，定義 2.6 の条件に反する．

命題 1 の証明

$j = n+1$ の場合は命題は成り立つ．

$j = k+1$ の場合に命題が成り立つとする．

A) $\mathbf{q}_k$ が自己競合ループの末尾でない場合：

この場合は $\overline{P_k} = \phi$ だから $C_p(k) = 0$ となるので，$D_p(k) = D_p(k+1) - Sig_p(k)$ となり，命題は成り立つ．

B) $\mathbf{q}_k$ が自己競合ループの末尾の場合：

この自己競合ループを L とし，L を構成する状態を $\mathbf{q}'_i$（$1 \leq i \leq r$, $\mathbf{q}'_r$ はループの末尾）とする．このとき，$\mathbf{q}'_r = \mathbf{q}_k$ である．

B1) $\mathrm{Item}(\mathbf{q}_k)$ に p に関するループ内生成列を含まない場合：

この場合は，$\mathbf{q}_I$ は $\mathbf{q}'_i\,(1 \leq i \leq r)$ でない．$\mathbf{q}_k$ は L の末尾であり，$\mathbf{q}_k \rightarrow \mathbf{q}'_1$ だから，$\mathbf{q}_I$ から $\mathbf{q}_k$ までのループでないパスは $\mathbf{q}_I$ から $\mathbf{q}'_1$ までのループでないパス（path1 とする）と $\mathbf{q}'_1$ から $\mathbf{q}_k$ までのループでないパス（path2 とする）から構成される．path1 上で発生する自己競合の数は $D_p(k+1)$ だから，path2 上で発生する自己競合の数を l とすると

$\mathrm{Item}(\mathbf{q}_k)$ で自己競合が発生する場合は，

$C_p(k) = l+1$, $Sig_p(k) = 1$ であり，

$\mathrm{Item}(\mathbf{q}_k)$ で自己競合が発生しない場合は，

$C_p(k) = l$, $Sig_p(k) = 0$ である．

いずれの場合も $D_p(k) = D_p(k+1) + C_p(k) - Sig_p(k) = D_p(k+1) + l$ となり，命題は成り立つ．

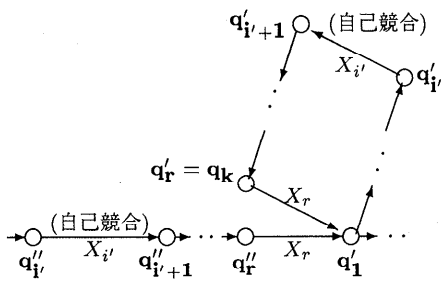


図9 B2) の場合の例
Fig. 9 Example for case B2).

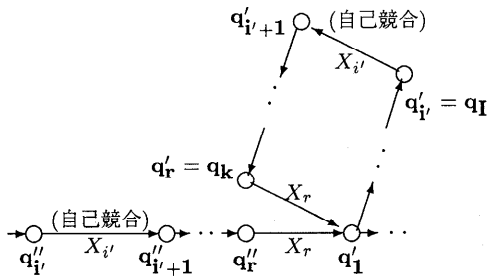


図10 B2a) の場合の例
Fig. 10 Example for case B2a).

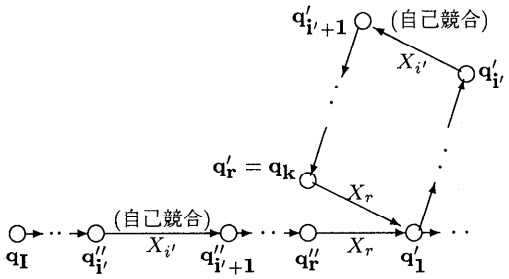


図11 B2b) の場合の例
Fig. 11 Example for case B2b).

B2) $\mathrm{Item}(\mathbf{q}_k)$ に p に関するループ内生成列を含む場合：

$\mathbf{q}_k$ の p に関するループ内生成列を q'，$\mathbf{q}_k$ の q' に関する先頭状態を $\mathbf{q}'_{i'}$ とすると，次の条件を満たす $\mathbf{q}''_t\,(i' \le t \le r)$ が存在する．

- $\mathbf{q}'_t \to^{X_t} \mathbf{q}'_{t+1}, \mathbf{q}''_t \to^{X_t} \mathbf{q}''_{t+1}\ (i' \le t < r)$
- $\mathbf{q}'_r \to^{X_r} \mathbf{q}'_1, \mathbf{q}''_r \to^{X_r} \mathbf{q}'_1$
- $\mathbf{q}'_r = \mathbf{q}_k$
- $\mathbf{q}''_t\,(i' \le t \le r)$ は，L を構成する状態でない

$\mathbf{q}_k$，$\mathbf{q}'_{i'}$，$\mathbf{q}''_{i'}$ の関係は図9のようになる．

B2a) $\mathbf{q}_I$ が $\mathbf{q}'_i\,(1 \le i < r)$ のいずれかの場合（図10）：

$\mathbf{q}_I = \mathbf{q}'_{i'}$ とする．$D_p(k)$ は $\mathbf{q}_I\,(= \mathbf{q}'_{i'})$ から $\mathbf{q}_k$ までのループでないパスで発生した自己競合の数だから，$D_p(k) \le M_p$ が成り立つ．ゆえに $C_p(k) = 0$ となる．

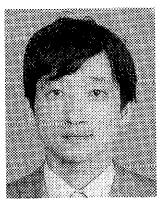
一方，$\mathbf{q}_I$ から $\mathbf{q}_k$ までのパス上に $\mathbf{q}'_1$ は存在しないから，この場合はa) と同様に $D_p(k) = D_p(k+1) - Sig_p(k)$ となり命題は成り立つ．

B2b) $\mathbf{q}_I$ が $\mathbf{q}'_i\,(1 \le i < r)$ のいずれでもない場合（図11）：

補題1より，L には $\mathbf{q}'_{i'}$ の競合の消滅状態が存在するから，$\mathbf{q}'_{i'}$ から $X_{i'} \cdots X_r X_1 \cdots$ を読み込んだ場合の状態のパス上に $\mathbf{q}'_{i'}$ の競合の消滅状態が存在する．ゆえに $\mathbf{q}''_{i'}$ から $X_{i'} \cdots X_r X_1 \cdots$ を読み込んだ場合の状態のパス上にも $\mathbf{q}''_{i'}$ の競合に関する消滅状態が存在するので，$\mathrm{Item}(\mathbf{q}_k)$ には $\mathbf{q}''_{i'}$ を先頭状態とする要素は存在しない．つまり命題の状況のスタックで $\mathbf{q}_I$ は $\mathbf{q}''_{i'}$ よりも下にあるから，$\mathbf{q}_I$ から $\mathbf{q}_k$ までのループでないパスで発生する自己競合の数（$D_p(k)$）は $\mathbf{q}''_{i'}$ から $\mathbf{q}_k$ までのループでないパスで発生する自己競合の数よりも大きい．ゆえに $D_p(k) > M_p$ が成り立つので $C_p(k) = l$ となる．

一方，$\mathbf{q}_I$ から $\mathbf{q}_k$ までのパス上に $\mathbf{q}'_1$ は存在するので，この場合はB1) と同様に，$\mathbf{q}_I$ から $\mathbf{q}_k$ までのループでないパスは $\mathbf{q}_I$ から $\mathbf{q}'_1$ までのループでないパスと $\mathbf{q}'_1$ から $\mathbf{q}_k$ までのループでないパスから構成される．ゆえにこの場合も $D_p(k) = D_p(k+1) + C_p(k) - Sig_p(k)$ となり，命題は成り立つ．

(平成 9 年 3 月 25 日受付)
(平成 10 年 1 月 16 日採録)



森本 真一（正会員）

昭和56年東京大学大学院理学系研究科情報科学専攻修士課程修了．同年日本電気（株）入社．言語処理系の研究開発および品質管理業務に従事．日本ソフトウェア科学会，JapanSIGAda 各会員．