Research Contribution

Almost Boolean Algebraic Computation of LALR(1) Look-Ahead Sets

HIROYUKI ANZAI*

In order to obtain an LALR(1) parser from a given grammar, it is necessary to construct an LR(0) automaton and to compute LALR(1) Look-Ahead sets. This paper presents a new method for doing so, based on the linear algebra-like approach instead of the traditional graph theoretical one.

After pointing out that the regular language generated from the empty set is isomorphic to Boolean algebra, this paper shows that the above problems can be solved by partially reducing them to problems of this simplest language, Boolean algebra, in such a manner that unknown sets are defined by simultaneous equations each of whose coefficients is a Boolean number.

For a given BNF, equations of this kind defining the state transition of an LR(0) automaton are given and solved. Follow sets are similarly defined and solved. Each solution is a formula for computing the desired sets, whose form is the product of the closure of a Boolean matrix and a vector of either Boolean numbers or sets of symbols. Finally, each Look-Ahead set is obtained as a union of properly selected Follow sets.

Until now, this kind of computation has been done in an iterative manner of set computation on the equations until the sets obtained become unchanged. Instead, this paper gives a formula for computing them by almost Boolean matrix computation.

1. Introduction

For a given grammar, in order to construct an LALR(1) parser, which is one of the most practical and popular kinds of parser, it is necessary to construct an LR(0) automaton and to compute LALR(1) Look-Ahead sets from the automaton. Many researchers have devised methods for doing so [1, 2, 3]. An efficient method was shown by DeRemer and Pennello [1] in 1982.

The traditional approach to solving the above problems is the graph theoretical one. Instead, this paper applies a linear algebra-like approach to the problems.

This approach to graphs was, however, shown to be a field of graph theory by Carrè [4] and Gondran and Minoux [5]. In Japan, this approach has been applied to finite automaton theory since the study by Utagawa, Inagaki, and Tange in 1965 [6]. Nozaki showed in 1967 that this approach is based on the algebraic system as a semi-ring [7]. We also studied applying the approach to finite automaton theory [8]. After that, we used the approach for generation of recursive-descent syntax-directed translators [9, 10].

This approach was first applied to context-free languages by Tixier [11]. For a given extended BNF, he gave a parametric representation equivalent to it, that is, a system of simultaneous right linear equations, and called it the Standard Right Linear equation or SRL equation. In the equations, each coefficient is a set of terminal and nonterminal symbols and each constant term is either the empty set ϕ or the empty string set λ. From the equation, he gave a parser of the top-down type.

As an application of this approach, we present a new method of obtaining an LR(0) automaton and its LALR(1) Look-Ahead sets.

This paper also points out that the language space generated from the empty alphabet is isomorphic to Boolean algebra, and shows that the above problems can be solved by partially reducing them to problems of this simplest language, Boolean algebra, in such a manner that unknown finite sets are defined in the form of simultaneous equations each of whose coefficients is a Boolean number. Formulas to compute the sets are then derived as the solutions of the equations.

For Tixier's SLR equations derived from a given BNF, equations of this kind defining the state transition of an LR(0) automaton are given and solved, and the solutions, that is, the state transitions, are represented as the product of a Boolean vector representing a state and a Boolean matrix representing the state transition function. First sets and Follow sets are also defined in the form of the above kinds of equation, respectively. Formulas to compute the sets are given as the solutions, whose form is the product of the closure of a Boolean matrix and a vector of symbol sets. Finally, each Look-

*Kyushu Institute of Technology, Faculty of Engineering, Kitakyushu-shi, 804, Japan.

Ahead set is obtained as a union of some Follow sets.

Until now, concrete computation of the above sets has been done by iteratively computing sets on recursively defined equations until the sets obtained become unchanged. Instead, this paper gives formulas for computing them in a manner that depends mainly on Boolean matrix computation.

Section 2 prepares, for a given alphabet Σ, basic matters necessary for computation of sets of strings generated from Σ. First, an algebra on language spaces generated from Σ is given as a semi-ring with idempotency in addition. It is pointed out that the language space generated from the empty alphabet is isomorphic to Boolean algebra. Then, a few matters are shown concerning regular languages, finite automata, and simultaneous right linear set-defining equations of the fixed point form, that is, the recursivly defined form, concerned with sets of strings as unknowns.

In Section 3, in order to treat context-free languages, the matters in Section 2 are extended on the assumption that Σ has the structure of $T \cup N$, where T is a set of terminals and N a set of nonterminals. The equations given in Section 2, each of whose coefficients is a subset of Σ and each of whose constant terms is either ϕ or λ, are extended so that Σ is $T \cup N$, and result into Tixier's SLR equations defining nonterminals in such a manner that they are derived from BNF defining the nonterminals.

In Section 4, for each nonterminal symbol X, the first symbol set First (X), a set of symbols appearing at the first position in sentential forms derived from X, is defined. Then, simultaneous equations for all First sets as unknowns are derived and solved. The solution is of the form $\Gamma^* \boldsymbol{d}$, a column vector each of whose constituents is a First set, where Γ^* is the closure of a matrix Γ each of whose elements is λ or ϕ, and $\boldsymbol{d}$ is a column vector each of whose constituents is a set of symbols.

Section 5 presents two methods of constructing an LR(0) automaton. The first uses the traditional approach to automaton theory, and the second uses the linear algebra-like one. In the latter, each state is represented as a Boolean vector and each state-transition as the product of a Boolean matrix and the Boolean vector.

In Section 6, for each nonterminal-transition in the LR(0) automaton, its Follow sets, a set of terminal symbols that cause state-transition from a state reached by the nonterminal-transition, are defined. Simultaneous equations concerned with the Follow sets as the unknowns are then derived and solved in the same manner as for Follow sets.

Finally, in Section 7, for each pair of a reduced state and the rule reduced there, the LALR(1) Look-Ahead set is given as a union of properly selected Follow sets.

2. Language Semi-Ring and Finite Automaton

This section prepares basic matters concerned with computation of sets of strings necessary for the following sections.

An alphabet, a set of symbols, is written Σ. The set of all strings generated from Σ is denoted by Σ^*. For any strings $u, v \in \Sigma^*$, the string uv is called the concatenation of u and v, which again belongs to Σ^* (closure). For any u, v, and $w \in \Sigma^*$, it holds that $(uv)w = u(vw)$ (associativity). The empty string ε is defined so that $\varepsilon w = w\varepsilon = w$ for any $w \in \Sigma^*$ (unit).

For a given set S, the cardinal number of S, i.e. the number of all elements in S, is written $\#(S)$ and the power set of S, i.e. the set of all subsets of S, is represented as PS. The empty set is written ϕ and the empty string set $\{\varepsilon\}$ λ.

For sets x and y, $x = y$ indicates an equivalence relation as a set, and $x \subseteq y$ indicates an inclusion relation. Furthermore, addition $x + y$ and multiplication $x \cdot y$ (usually written xy) are defined as follows:

$$x + y = \{w \mid w \in x \text{ or } w \in y\}$$

$$x \cdot y = \{uv \mid u \in x,\ v \in y\}$$

From the above and set theory, we have the following lemma.

Lemma 2.1 For any x, y and z in $P\Sigma^*$, the following relations (1) ~ (10) are valid.

(1) $x + y \in P\Sigma^*$ (closure in addition)
(2) $(x + y) + z = x + (y + z)$ (associativity in addition)
(3) $x + \phi = \phi + x = x$ (unit in addition)
(4) $x + y = y + x$ (commutativity in addition)
(5) $xy \in P\Sigma^*$ (closure in multiplication)
(6) $(xy)z = x(yz)$ (associativity in multiplication)
(7) $x\lambda = \lambda x = x$ (unit in multiplication)
(8) $x(y + z) = xy + xz$ (left distributivity)
(9) $(x + y)z = xz + yz$ (right distributivity)
(10) $x + x = x$ (idempotency in addition)

The algebra $\mathscr{A}$ $(P\Sigma^*, +, \cdot\phi, \lambda)$ defined by the above relations as axioms is a semi-ring with the idempotent law in addition. We call it a "language semi-ring." For this semi-ring we take the set theoretical approach to Σ^* through out the paper, for practical reasons. This approach allows us to use set theoretical concepts directly. Otherwise, we would have to use extra definitions.

For any $x \in P\Sigma^*$, we define $x^0 = \lambda$, $x^n = x^{n-1}x$, for $n = 0, 1, \cdots$. The closure and the positive closure of x are then defined as $x^* = \sum_{i=0}^{\infty} x^i$ and $x^+ = \sum_{i=1}^{\infty} x^i$, respectively.

For a given set S, let A be an $m \times n$ matrix with a_{ij} $(1 \leqq i \leqq m,\ 1 \leqq j \leqq n)$ in PS as its elements. We call it an S-matrix and represent it as $A = (a_{ij})$. The (i, j) element a_{ij} of A is also denoted by $[A]_{ij}$. For an $m \times n$ matrix A and an $m' \times n'$ matrix B, we write $A = B$ if and only if $m = m'$, $n = n'$, and $[A]_{ij} = [B]_{ij}$ for any i and j. If $[A]_{ij} = [B]_{ij}$ in the above definition is replaced by $[A]_{ij} \subseteq [B]_{ij}$, we write $A \subseteq B$. When and only when $m = m'$ and $n = n'$, the sum of A and B is defined as $[A + B]_{ij} = [A]_{ij} + [B]_{ij}$, and when and only when $n = m'$, the product of

A and B is defined as $[AB]_{ij}=\Sigma_{k=1}^{n}[A]_{ik}[B]_{kj}$. The matrix where all elements are ϕ is called an empty matrix and denoted by Φ. The square matrix whose diagonal elements are all λ and whose others are all ϕ is called an identity matrix and denoted by E.

For the relations from (1) to (10) that define the algebra $\mathscr{A}(P\Sigma^*, +, \cdot, \phi, \lambda)$, replace ϕ and λ by matrices Φ and E, respectively, and x, y, and z by matrices A, B, and C, respectively. If the sum and the product can be defined for them, then we can show again that all the relations remain valid [7].

For any square matrix A, we define $A^0=E$, $A^n=A^{n-1}A$, for $n=1, 2, \cdots$. Then the closure and the positive closure of A are defined as $A^*=\Sigma_{i=0}^{\infty}A^i$ and $A^+=\Sigma_{i=1}^{\infty}A^i$, respectively.

For any $a\in P\Sigma^*$ and for any Σ^*-matrix $A=(a_{ij})$, the left and the right scalar product of a and A are defined as $aA=(aa_{ij})$ and $Aa=(a_{ij}a)$, respectively. Note that if A is a λ-matrix, each of whose elements is λ or ϕ, the scalar product of a and A is commutative, that is, $aA=Aa=(a'_{ij})$, where $a'_{ij}=a$ if $a_{ij}=\lambda$ and $a'_{ij}=\phi$ if $a_{ij}=\phi$.

Vectors are all shown in Gothic letters. An n-dimensional row vector is regarded as an $n\times 1$ matrix. The transpose of a vector $\mathfrak{v}$ is written ${}^t\mathfrak{v}$. A column vector whose components are all ϕ is denoted by ϕ. Similarly, a column vector whose components are all λ is denoted by λ. A row vector whose first constituent is λ and whose others are all ϕ is denoted by $\mathfrak{e}_1$.

Let S be a given set of symbols. Then for a given symbol s, in order to know whether S contains the symbol σ or not, we define a function $\partial:\Sigma\times P\Sigma\to P\lambda$, as follows:

$$\partial_s S=\lambda \text{ if } s\in S, \tag{2.1}$$
$$=\phi \text{ otherwise.}$$

For a Σ-matrix $A=(a_{ij})$, the definition is extended as

$$\partial_s A=(\partial_s a_{ij}). \tag{2.2}$$

If a given alphabet Σ is empty, the language space generated from Σ is isomorphic to Boolean algebra under the correspondances of $\phi\leftrightarrow$false, $\lambda\leftrightarrow$true, $+\leftrightarrow$or and $\cdot\leftrightarrow$and. Therefore, the role of the function ∂ is to reduce symbol manipulation to Boolean computation. We give a simple example.

Any Σ-matrix A can be represented as follows:

$$A=\Sigma_{s\in\Sigma}\{s\}\partial_s A=\Sigma_{s\in\Sigma}\partial_s A\{s\}$$

In order to avoid a complicated description, we adopt the following notational conventions: each symbol $s\in\Sigma$ is treated as a string $s\in\Sigma^*$, and each string $w\in\Sigma^*$ is sometimes treated (i.e. interpreted) as a set $\{w\}\in P\Sigma^*$. Thus the above equation is abbreviated as follows:

$$A=\Sigma_{s\in\Sigma}s\partial_s A=\Sigma_{s\in\Sigma}\partial_s As$$

Since $\partial_s As'=s'\partial_s A$ for any s, $s'\in\Sigma$, we have

$$A^2=(\Sigma_{s\in\Sigma}s\partial_s A)(\Sigma_{s'\in\Sigma}s'\partial_{s'}A)$$
$$=\Sigma_{s\in\Sigma}\Sigma_{s'\in\Sigma}ss'\partial_s A\partial_{s'}A.$$

Note that if $\partial_s A$ is an $n\times n$ λ-matrix (i.e. Boolean matrix), then $\partial_s A\partial_{s'}A$ can be transformed into a matrix of the same type.

For instance, let a matrix A be $\begin{bmatrix}a+b & \phi\\ b & a\end{bmatrix}$; then we have

$$A=\begin{bmatrix}a & \phi\\ \phi & a\end{bmatrix}+\begin{bmatrix}b & \phi\\ b & \phi\end{bmatrix}=a\begin{bmatrix}\lambda & \phi\\ \phi & \lambda\end{bmatrix}+b\begin{bmatrix}\lambda & \phi\\ \lambda & \phi\end{bmatrix}$$

$$A^2=\left(a\begin{bmatrix}\lambda & \phi\\ \phi & \varphi\end{bmatrix}+b\begin{bmatrix}\lambda & \phi\\ \lambda & \phi\end{bmatrix}\right)\left(a\begin{bmatrix}\lambda & \phi\\ \phi & \lambda\end{bmatrix}+b\begin{bmatrix}\lambda & \phi\\ \lambda & \phi\end{bmatrix}\right)$$
$$=aa\begin{bmatrix}\lambda & \phi\\ \phi & \lambda\end{bmatrix}+(ab+ba+bb)\begin{bmatrix}\lambda & \phi\\ \lambda & \phi\end{bmatrix}$$

For a given Σ, we introduce simultaneous right linear equations each of whose coefficients is a subset of Σ and each of whose constant terms is λ or ϕ, as follows:

$$x_i=\sum_{j=1}^{n}a_{ij}x_j+c_i \tag{2.3}$$

where $a_{ij}\in P\Sigma$, $c_i\in P\lambda$, $i=1, 2, \cdots, n$.

The above simultaneous equations are represented as an n-dimensional right linear equation $\mathfrak{x}=A\mathfrak{x}+\mathfrak{c}$, where $\mathfrak{x}={}^t(x_1, x_2, \cdots, x_n)$, $\mathfrak{c}={}^t(c_1, c_2, \cdots c_n)$, $A=(a_{ij})$, $1\leqq i, j\leqq n$. It is easy to show that the equation has the solution $A^*\mathfrak{c}$. Namely, the right part of the equation $=A\mathfrak{x}+\mathfrak{c}=AA^*\mathfrak{c}+\mathfrak{c}=(AA^*+E)\mathfrak{c}=A^*\mathfrak{c}=$the left part of the equation. We call the equation a Σ-righty linear equation system, or Σ-RLES for short.

It is well-known that, for a given regular expression R generated from a given alphabet Σ, R can be unfolded by introducing appropriate intermediate parameters x_1, $x_2, \cdots, x_n$, which results in obtaining a Σ-RLES whose first constituent of the solution $A^*\mathfrak{c}$ is equivalent to R (i.e. $R=x_1=\mathfrak{e}_1A^*\mathfrak{c}$) [6, 8]. For the Σ-RLES, we can give a finite automaton that can accept the language defined by the regular expression R, as follows:

$$\mathscr{A}(S, \Sigma, x_1, F, \tau) \tag{2.4}$$

where

S: a set of states: $\{x_1, x_2, \cdots, x_n\}$,
Σ: a set of input symbols,
x_1: the initial state,
F: a set of final states: $\{x_i|[\mathfrak{c}]_i=\lambda\}$,
τ: the transition function: $S\times\Sigma\to S$
: $\tau(x_i, s)=x_j$ iff $s\in[A]_{ij}$ iff $[\partial_s A]_{ij}=\lambda$.

The language accepted by $\mathscr{A}$ is defined as

$$T(\mathscr{A})=\{s_1s_2\cdots s_r|\tau(\cdots\tau(\tau(x_1, s_1), s_2), \cdots, s_r)\in F, r\geqq 0\}.$$

In the above, $r=0$ means that $s_1s_2\cdots s_r=\varepsilon$ and $\tau(x_1, \varepsilon)=x_1$.

Lemmas and a theorem used in the following sections are introduced below. Proofs are given in the Appendix.

Lemma 2.2 $\tau(\cdots\tau(\tau(x_i, s_1), s_2), \cdots, s_r)=x_j$
iff $[\partial_{s_1}A\partial_{s_2}A\cdots\partial_{s_r}A]_{ij}=\lambda$. □

Lemma 2.3 $s_1s_2\cdots s_r\in T(\mathscr{A})$

iff $e_1\partial_{s_1}A\partial_{s_2}A\cdots\partial_{s_r}Ac=\lambda$. □

Theorem 2.1 $T(\mathscr{A})=e_1A^*c$. □

For a set $\Sigma'\subseteq\Sigma$ and states x_i and x_j, it is said that x_i (or x_j) is Σ'-accessible to x_j (or from x_i) if there exist $s_1, s_2, \cdots, s_r\in\Sigma'$, $r\geqq 0$, such that $\tau(\cdots((x_i, s_1), s_2), \cdots, s_r)=x_j$. When $\Sigma'=\Sigma$, "Σ-accessible" becomes simply "accessible". As regards, A and Σ', a λ-matrix C' is defined as $C'=\sum_{s\in\Sigma'}\partial_sA$ and called a Σ'-adjacent matrix of A. In the following, C', C'' and C''' are the Σ'-, the Σ''- and the Σ'''-adjacent matrices, respectively.

Lemma 2.4 x_i is Σ'-accessible to x_j iff $[C'^*]_{ij}=\lambda$. □

If state x_i is Σ'-accessible from the initial state (or to at least one of the final states), then x_i is said to be initially (or finally) Σ'-accessible. State x_i is initially (or finally) Σ'-accessible if and only if it holds that $[e_1C'^*]_i=\lambda$ (or $[C'^*c]_i=\lambda$).

Lemma 2.5 For sets Σ', Σ'', $\Sigma'''\subseteq\Sigma$ and for any symbols s, $s'\in\Sigma$, there exists a string $w'sw''s'w'''\in T(\mathscr{A})$ such that $w'\in\Sigma'^*$, $w''\in\Sigma''^*$ and $w'''\in S'''^*$, if and only if it holds that

$$e_1C'^*\partial_sAC''^*\partial_{s'}A_XC'''^*c=\lambda. \square$$

Lemma 2.6 For any s, $s'\in\Sigma$ and $\Sigma'\subseteq\Sigma$, if there exists a string $wsw's'w''$ in $T(\mathscr{A})$ such that w, $w''\in\Sigma^*$ and $w'\in\Sigma'^*$, then it holds that

$$e_1C^*\partial_sAC'^*\partial_{s'}A\lambda=\lambda. \square$$

Lemma 2.7 For any $s\in\Sigma$ and $\Sigma'\subseteq\Sigma$, if there exists a string wsw' in $T(\mathscr{A})$ such that $w\in\Sigma^*$ and $w'\in\Sigma'^*$, then it holds that

$$'\lambda\partial_sAC'^*c=\lambda. \square$$

When each state of $\mathscr{A}$ is initially and finally accessible, $\mathscr{A}$ is said to be reduced. If $\mathscr{A}$ is reduced, Lemmas 2.6 and 2.7 are not only necessary but sufficient.

3. Context-Free Language and Multiple Finite Automata System

Context-free languages are treated in this section by extending the reasoning in the previous section.

A set of terminals and a set of nonterminals are denoted by T and N, respectively. A union of T and N is written V. For notational conventions, we use t as a terminal in T; X, Y, and Z as nonterminals in N; and s as an element (a terminal or a nonterminal) of V.

The right part of each BNF or EBNF can be regarded as a regular expression generated from an alphabet V. Thus, as shown in the previous section, for each nonterminal X, the right part R_X of the X-defining BNF can be unfolded by introducing appropriate intermediate parameters $x_{X1}, x_{X2}, \cdots, x_{Xn_X}$, which results in an n_X-dimensional V-RLES, as follows:

$$X=x_{X1}$$

$$x_{Xi}=\sum_{j=1}^{n_X}a_{Xij}x_{Xj}+c_{Xi} \qquad (3.1)$$

$$i=1, 2, \cdots, n_X,$$

or

$$x_X=A_Xx_X+c_X, \qquad (3.2)$$

and it holds that

$$X(=R_X)=x_{X1}=[A_X^*c_X]_1\subseteq V^*, \qquad (3.3)$$

where $[A_X^*c_X]_1$ is the first constituent of the vector $A_X^*c_X$, the (minimal) solution of Eq. (3.2), and the equal sign "=" means the equivalence relation for sets of strings generated from the alphabet V. The simultaneous equation system for all $X\in N$ as the unknowns composed of Eq. (3.3) for each $X\in N$ is considered to be satisfied by the intrinsic solution on T^*.

Tixier gave this kind of equation first and called it the standard right linear (abr. SLR) equation [11].

Example 1. An example grammar given by Tremblay and Sorenson [2], which is not SLR(1) but LALR(1), is shown in Fig. 1 in the form of BNF. This form is easily transformed into a kind of graphic representation, as shown in Fig. 2, which can be interpreted as a system of

$$\begin{aligned} S &\rightarrow G\ \# \\ G &\rightarrow E = E \mid f \\ E &\rightarrow T \mid E + T \\ T &\rightarrow f \mid T * f \end{aligned}$$

Fig. 1 An example grammar.

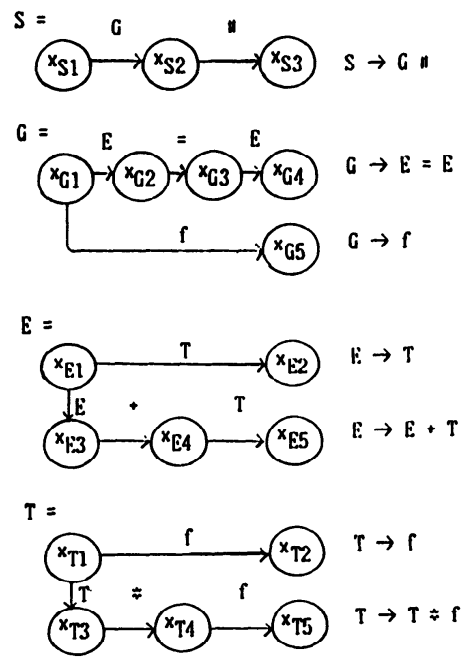

Fig. 2 Multiple finite automata derived from Fig. 1.

$$A_S=\begin{bmatrix}\phi & G & \phi\\ \phi & \phi & \|\\ \phi & \phi & \phi\end{bmatrix}\qquad c_S=\begin{bmatrix}\phi\\ \phi\\ \lambda\end{bmatrix}$$

$$A_G=\begin{bmatrix}\phi & E & \phi & \phi & f\\ \phi & \phi & = & \phi & \phi\\ \phi & \phi & \phi & E & \phi\\ \phi & \phi & \phi & \phi & \phi\\ \phi & \phi & \phi & \phi & \phi\end{bmatrix}\quad c_G=\begin{bmatrix}\phi\\ \phi\\ \phi\\ \lambda\\ \lambda\end{bmatrix}$$

$$A_E=\begin{bmatrix}\phi & T & E & \phi & \phi\\ \phi & \phi & \phi & \phi & \phi\\ \phi & \phi & \phi & + & \phi\\ \phi & \phi & \phi & \phi & T\\ \phi & \phi & \phi & \phi & \phi\end{bmatrix}\quad c_E=\begin{bmatrix}\phi\\ \lambda\\ \phi\\ \phi\\ \lambda\end{bmatrix}$$

$$A_T=\begin{bmatrix}\phi & f & T & \phi & \phi\\ \phi & \phi & \phi & \phi & \phi\\ \phi & \phi & \phi & \div & \phi\\ \phi & \phi & \phi & \phi & f\\ \phi & \phi & \phi & \phi & \phi\end{bmatrix}\quad c_T=\begin{bmatrix}\phi\\ \lambda\\ \phi\\ \phi\\ \lambda\end{bmatrix}$$

Fig. 3 Coefficient matrices and constant terms of V-RLES derived from Fig. 1 or Fig. 2.

state transition diagrams representing a system of multiple finite automata where each automaton co-operates with the others to accept the language defined by the grammar given in Fig. 1. It is easy to construct V-RLES from Fig. 1 or Fig. 2. Coefficient matrices and constant terms of the V-RLES are shown in Fig. 3.

For each nonterminal X, the X-defining BNF is transformed into an automaton as shown in Fig. 2. We call it automaton $\mathscr{A}_X$, which is described as follows:

$$\mathscr{A}_X(S_X,\ V,\ x_{X1},\ F_X,\ \tau_X),\ X\in N \tag{3.4}$$

where

S_X: a set of states $\{x_{X1}, x_{X2}, \cdots, x_{Xn}\}$,
V: a set of input symbols (terminals and nonterminals),
x_{X1}: the initial state,
F_X: a set of final states, which the rules to be reduced are attached to,
τ_X: the transition function: $S_X\times V\to S_X$.

The state transition caused by a nonterminal is called a nonterminal-transition, and when the nonterminal is Y, the automaton $\mathscr{A}_Y$ is regarded as called there in the manner of the recursive-call in normal programming languages. Furthermore, each final state is regarded as having a kind of output called "reduction," which reduces the rule attached to the final state just when the return operation is performed there.

The behavior of the above system is generally nondeterministic. One method of making it deterministic if possible is to construct an LR(0) automaton using LALR(1) Look-Ahead sets from the system, which is described in the following sections.

For automaton $\mathscr{A}_X$, state x_{Xi} is said to be ε-accessible to state x_{Xj} if x_{Xi} is accessible to x_{Xj} via a sequence of transitions caused only by the empty string ε. A nonterminal X that derives the empty string ε ($X\overset{*}{\Rightarrow}\varepsilon$) is called the ε-generating nonterminal. A set of all the ε-generating nonterminals in N is denoted by N'. Thus in $\mathscr{A}_X$, x_{Xi} is ε-accessible to x_{Xj} if and only if x_{Xi} is N'-accessible to x_{Xj}.

As mentioned in Section 2, the N'-adjacent matrix of A_X is $C'_X=\Sigma_{Y\in N'}\partial_Y A_X$. Therefore from Lemma 2.4, it holds that x_{Xi} is ε-accessible to x_{Xj} if and only if $[C'_X{}^*]_{ij}=\lambda$. If N' is empty, $C'_X=\Phi_X$ and then $C'_X{}^*=\Phi_X{}^*=E$.

In the following discussion, for a given BNF, we assume that the reader knows the production rules in the production grammar equivalent to it, and we write P as the set of the production rules. Furthermore, we use V-RLES, the SLR equation Eq. (3.2), where λ-vectors e_1, ϕ, and λ associated with Eq. (3.2) are represented as i_X, ϕ_X, and λ_X respectively in order to clarity their dimensions.

4. First Symbol Sets and Lmost Symbol Sets

For a given production grammar $G(N, T, X_0, P)$, this section gives two kinds of symbol set called First sets and Lmost sets. The former are sets of terminals used in Section 6 and the latter are sets of nonterminals used in Section 5.

For a nonterminal $X\in N$, a set of terminals appearing in the first position of sentential forms derived from X is defined as

$$\mathrm{First}(X)=\{t\in T\mid X\overset{*}{\Rightarrow}tw,\ w\in V^*\} \tag{4.1}$$

and is called First (symbol) set.

First, a method of computing First sets is given. Initially, we define a function $\gamma: N\times V\to P\lambda$, as follows:

$$\begin{aligned}\gamma_{Xs}=\lambda\quad & \text{if there exists a rule } X\to\xi sw \text{ in } P\\ & \text{such that } \xi\in N'^* \text{ and } w\in V^*.\\ =\phi\quad & \text{otherwise,}\end{aligned} \tag{4.2}$$

where N' is the set of all ε-generating nonterminals. $j\in N'^*$ means that there is a case where $\xi\overset{*}{\Rightarrow}\varepsilon$.

For our V-RLES $x_X=A_Xx_X+c_X$, from Lemma 2.6 and using the N'-adjacent matrix $C'_X=\Sigma_{Z\in N'}\partial_Z A_X$, the above γ_{Xs} is given as follows:

$$\gamma_{Xs}=i_XC'_X{}^*\partial_s A_X\lambda_X \tag{4.3}$$

Thus, we have the following relation:

$$\mathrm{First}(X)=\sum_{Y\in N}\gamma_{XY}\,\mathrm{First}(Y)+\sum_{t\in T}\gamma_{Xt}\{t\} \tag{4.4}$$

Now, to solve Eq. (4.4), we define two $\#(N)$-dimensional column V-vectors $u=(u_X)$, $u_X=\mathrm{First}(X)$, and $d=(d_X)$, $d_X=\Sigma_{t\in T}\gamma_{Xt}\{t\}$, and a $\#(N)\times\#(N)$ λ-matrix

$\Gamma=(\gamma_{XY})$. Then, Eq. (4.4) becomes

$$u=\Gamma u+d \tag{4.5}$$

and has the solution

$$u=\Gamma^* d \tag{4.6}$$

For $t\in T$, the definition of the First set is extended as

$$\text{First}(t)=\{t\}. \tag{4.7}$$

Next, for each $X\in N$, we define Lmost(X) as a set of nonterminals appearing in the first (i.e. left-most) position of sentential forms derived from X without application of ε-generation. A function $\gamma':N\times N\to P\lambda$ is then defined as follows:

$$\begin{aligned}\gamma'_{XY}=\lambda \quad & \text{if there exists a rule } X\to Yw \text{ in } P \\ & \text{such that } Y\in N \text{ and } w\in V^*. \qquad (4.8)\\ =\phi \quad & \text{otherwise.}\end{aligned}$$

For our V-RLES $x_X=A_X x_X+c_X$, the above γ'_{XY} is given as follows:

$$\gamma'_{XY}=i_X\partial_Y A_X\lambda_X \tag{4.9}$$

Then for each $X\in N$, we define a set of nonterminals as follows:

$$\text{Lmost}(X)=\sum_{Y\in N}\gamma'_{XY}\,\text{Lmost}(Y)+\sum_{Y\in N}\gamma'_{XY}\{Y\} \tag{4.10}$$

Here, we define two $\#(N)$-dimensional column V-vectors $u=(u_X)$, $u_X=\text{Lmost}(X)$, and $d=(d_X)$, $d_X=\sum_{Y\in N}\gamma'_{XY}\{Y\}$, and a $\#(N)\times\#(N)$ λ-matrix $\Gamma'=(\gamma'_{XY})$. Eq. (4.10) then becomes

$$u=\Gamma' u+d=\Gamma'^* d. \tag{4.11}$$

Note that γ_{Xs} and γ'_{XY} is obtained not only from V-RLES but also from P. In order to compute Γ^* and Γ'^*, we have Warshall's famous theorem [12], of which a λ-matrix representation is shown in the appendix.

When N' is empty, we have $\gamma_{XY}=\gamma'_{XY}$. Therefore, in this case, we can have First sets and Lmost sets at once, as follows:

$$u=\Gamma u+d=\Gamma^* d \tag{4.12}$$

$$\text{where } u=(u_X),\ u_X=\text{Lmost}(X)\cup\text{First}(X) \tag{4.13}$$

$$d=(d_X),\ d_X=\sum_{s\in V} i_X\partial_s A_X\lambda_X\{s\}=i_X A_X\lambda_X \tag{4.14}$$

$$\Gamma=(\gamma_{XY}),\ \gamma_{XY}=i_X\partial_Y A_X\lambda_X=\partial_Y(i_X A_X\lambda_X)=\partial_Y d_X. \tag{4.15}$$

Example 2. Computation of Lmost sets and First sets in Example 1.

There is no ε-generating nonterminal in Fig. 1. Therefore, from Fig. 3 and Eqs. (4.14) and (4.15), we have d and Γ, as follows:

$$d=\begin{bmatrix}d_S\\ d_G\\ d_E\\ d_T\end{bmatrix}=\begin{bmatrix}\{\ G\ \}\\ \{E, f\}\\ \{E, T\}\\ \{T, f\}\end{bmatrix},\qquad \begin{aligned}\Gamma&=(\gamma_{XY})=(\partial_Y d_X)\\ &=(\partial_S d\ \partial_G d\ \partial_E d\ \partial_T d)\end{aligned}$$

$$\Gamma=\begin{bmatrix}\phi & \lambda & \phi & \phi\\ \phi & \phi & \lambda & \phi\\ \phi & \phi & \lambda & \lambda\\ \phi & \phi & \phi & \lambda\end{bmatrix},\quad \Gamma^*=\begin{bmatrix}\lambda & \lambda & \lambda & \lambda\\ \phi & \lambda & \lambda & \lambda\\ \phi & \phi & \lambda & \lambda\\ \phi & \phi & \phi & \lambda\end{bmatrix},$$

Then from Eq. (4.12), the desired sets are obtained as

$$u=\begin{bmatrix}\text{Lmost}(S)\cup\text{First}(S)\\ \text{Lmost}(G)\cup\text{First}(G)\\ \text{Lmost}(E)\cup\text{First}(E)\\ \text{Lmost}(T)\cup\text{First}(T)\end{bmatrix}=\Gamma^* d=\begin{bmatrix}\{G, E, T, f\}\\ \{\ \ E, T, f\}\\ \{\ \ E, T, f\}\\ \{\ \ \ \ T, f\}\end{bmatrix}.$$

5. LR(0) Automaton

The traditional method of deriving an LR(0) automaton from a given grammar uses the dot notation on production rules and is considered to be an extended variation of the subset construction method used in making a nondeterministic finite automaton deterministic. This method is shown first by the usual automata theoretical approach and then by a linear algebra-like one.

Automata Theoretical Approach

For the multiple finite automata system given in Section 3, an LR(0) automaton $\mathscr{A}_{\text{LR}}$ is given on the asumption that it is a simulator of $\mathscr{A}_X$, $X\in N$, cooperating with each other to analyze a given input string. The LR(0) automaton $\mathscr{A}_{\text{LR}}$ has a finite number of states $I_0, I_1, \cdots, I_K$. Each state I_k, the so-called LR(0) term, is given as a set of all I_{kX} for $X\in N$, where I_{kX} is a subset of the set of states S_X for each automaton $\mathscr{A}_X$.

For a given input $s\in V$, the state transition of the LR(0) automaton is defined as follows: First, for each automaton $\mathscr{A}_X$, the transition function $\Upsilon_X:PS_X\times V\to PS_X$ is given as

$$J_{kX}=\Upsilon_X(I_{kX}, s)=\{x'\mid{}^3x\in I_{kX},\ x'\in\tau_X(x, s)\}. \tag{5.1}$$

Put $S_{\text{LR}}=\cup_{X\in N}S_X$; then for $\mathscr{A}_{\text{LR}}$, the preparatory transition function Trans: $PS_{\text{LR}}\times V\to PS_{\text{LR}}$ is given as a union of all J_{kX}, $X\in N$, as follows:

$$J_k=\text{Trans}(I_k, s)=\cup_{X\in N}J_{kX}=\cup_{X\in N}\Upsilon_X(I_{kX}, s). \tag{5.2}$$

Next, the other preparatory functions $\Psi_X:PS_{\text{LR}}\to PS_X$, $X\in N$, are defined such that the values $J'_{kX}=\Psi_X(J_k)$ are given as the solutions of the following simultaneous equations:

$$J'_{kX}=J_{kX}\cup\{x_{X1}\in S_X\mid{}^3Y\in N,\ {}^3y\in J'_{kY},\ {}^3y'\in S_Y:\ y'=\tau_Y(y, X)\}. \tag{5.3}$$

For any J'_{kX}, $X\in N$, that satisfies the above recursively defined simultaneous equations, we can expect that for $X\in N$, if there exists Y such that there is a state $y\in J'_{kY}$ from which a state transition caused by the nonterminal X can start, then J'_{kX} contains x_{X1}, the initial state of $\mathscr{A}_X$.

Instead of Eq. (5.3), the following definition is available:

$$J'_{kX}=\Psi_X(J_k)$$
$$=J_{kX}\cup\{x_{X1}\in S_X\mid {}^3Y\in N,\ {}^3y\in J_{kY},\ {}^3y'\in S_Y:$$
$$[y'=\tau_Y(y, X) \text{ or } {}^3Z\in N: X\in \text{Lmost}(Z),$$
$$y'=\tau_Y(y, Z)]\}. \quad (5.4)$$

This means that for $X\in N$, if there exists Y such that there is a state $y\in J_{kY}$ from which can start a state transition caused by the nonterminal X or by the nonterminal Z whose Lmost set contains X, then J'_{kX} contains x_{X1}. It will be shown in the next approach that Eq. (5.4) is the solution of Eq. (5.3).

A function Closure: $PS_{LR}\to PS_{LR}$ is then constructed as a union of all J'_{kX}, as follows:

$$\text{Closure}(J_k)=\cup_{X\in N}J'_{kX}=\cup_{X\in N}\Psi_X(J_k). \quad (5.5)$$

Finally, the desired transition function of $\mathscr{A}_{LR}$ Goto: $PS_{LR}\times V\to PS_{LR}$ is synthesized as shown below:

$$\text{Goto}(I_k, s)=\text{Closure}(\text{Trans}(I_k, s)). \quad (5.6)$$

For the starting nonterminal X_0, the initial state I_0 of $\mathscr{A}_{LR}$ is given for the initial state $x_{X_0 1}$ of $\mathscr{A}_{X_0}$ as follows:

$$I_0=\text{Closure}(\{x_{X_0 1}\}). \quad (5.7)$$

If a state I_k contains a final state $x_{Xi}{}^{\#}$concerned with a production rule $X\to\alpha$, the state I_k is, in $\mathscr{A}_{LR}$, a reduced state for the rule $X\to\alpha$.

Example 3. The LR(0) automaton $\mathscr{A}_{LR}$ obtained from the finite automaton system given in Fig. 2 is shown in Figs. 4 and 5.

The above method can be seen to construct and LR(0) automaton $\mathscr{A}_{LR}$ from the finite automata $\mathscr{A}_X$, $X\in N$, so as to embed each of them into $\mathscr{A}_{LR}$ one or more times. In the above example, this is shown by the "Embedding table" in Fig. 6. In this table, each column is indexed by a nonterminal-transition (I_l, Y) which corresponds to an occurrence of embedding automaton $\mathscr{A}_Y$ from I_l, and each row is indexed by state I_k of $\mathscr{A}_{LR}$. If a state x_{Yi} of $\mathscr{A}_Y$ embedded from I_l is contained in I_k, the state x_{Yi} is written in the $[I_k, (I_l, Y)]$-th element of the talbe. If $\mathscr{A}_Y$ is embedded from I_l, the $[I_l, (I_l, Y)]$-th element contains x_{Y1}, the initial state of $\mathscr{A}_Y$.

In Example 3, $\mathscr{A}_S$ derived from $S\to G\#$ is initially given as $I_0 \overset{G}{\text{---}} I_1 \overset{\#}{\text{---}} I_5$. For the nonterminal-transition (I_0, G) there, $\mathscr{A}_G$ derived from $G\to E=E\mid f$ is embedded as $I_0 \overset{E}{\text{---}} I_2 \overset{=}{\text{---}} I_6 \overset{E}{\text{---}} I_9$ and $I_0 \overset{f}{\text{---}} I_4$. Because there are two nonterminal-transitions (I_0, E) and (I_6, E) in the embed-

$$\begin{aligned}
J_0 &= \{ x_{S1} \} \\
&=>\{ x_{S1}, x_{G1}, x_{E1}, x_{T1} \} && = I_0 \\
\text{Trans}(I_0,G) &= \{ x_{S2} \} && = \text{Goto}(I_0,G) = I_1 \\
\text{Trans}(I_0,E) &= \{ x_{G2}, x_{E3} \} && = \text{Goto}(I_0,E) = I_2 \\
\text{Trans}(I_0,T) &= \{ x_{E2}{}^{\#}, x_{T3} \} && = \text{Goto}(I_0,T) = I_3{}^{\#} \\
\text{Trans}(I_0,f) &= \{ x_{G5}{}^{\#}, x_{T2}{}^{\#} \} && = \text{Goto}(I_0,f) = I_4{}^{\#} \\
\text{Trans}(I_1,\#) &= \{ x_{S3}{}^{A} \} && = \text{Goto}(I_1,\#) = I_5{}^{A} \\
\text{Trans}(I_2,=) &= \{ x_{G3} \} \\
&=>\{ x_{G3}, x_{E1}, x_{T1} \} && = \text{Goto}(I_2,=) = I_6 \\
\text{Trans}(I_2,+) &= \{ x_{E4} \} \\
&=>\{ x_{E4}, x_{T1} \} && = \text{Goto}(I_2,+) = I_7 \\
\text{Trans}(I_3,*) &= \{ x_{T4} \} && = \text{Goto}(I_3,*) = I_8 \\
\text{Trans}(I_6,E) &= \{ x_{G4}{}^{\#}, x_{E3} \} && = \text{Goto}(I_6,E) = I_9{}^{\#} \\
\text{Trans}(I_6,f) &= \{ x_{T2}{}^{\#} \} && = \text{Goto}(I_6,f) = I_{10} \\
\text{Trans}(I_7,T) &= \{ x_{E5}{}^{\#}, x_{T3} \} && = \text{Goto}(I_7,T) = I_{11}{}^{\#} \\
\text{Trans}(I_8,f) &= \{ x_{T5}{}^{\#} \} && = \text{Goto}(I_8,f) = I_{12} \\
\text{Trans}(I_6,T) &= \{ x_{E2}{}^{\#}, x_{T3} \} && = \text{Goto}(I_6,T) = I_3{}^{\#} \\
\text{Trans}(I_7,f) &= \{ x_{T2}{}^{\#} \} && = \text{Goto}(I_7,f) = I_{10}{}^{\#} \\
\text{Trans}(I_9,+) &= \{ x_{E4} \} && = \text{Goto}(I_9,+) = I_7 \\
\text{Trans}(I_{11},*) &= \{ x_{T4} \} && = \text{Goto}(I_{11},*) = I_8
\end{aligned}$$

Fig. 4 Transition function of the LR(0) automaton obtained from Fig. 2 in Example 1.

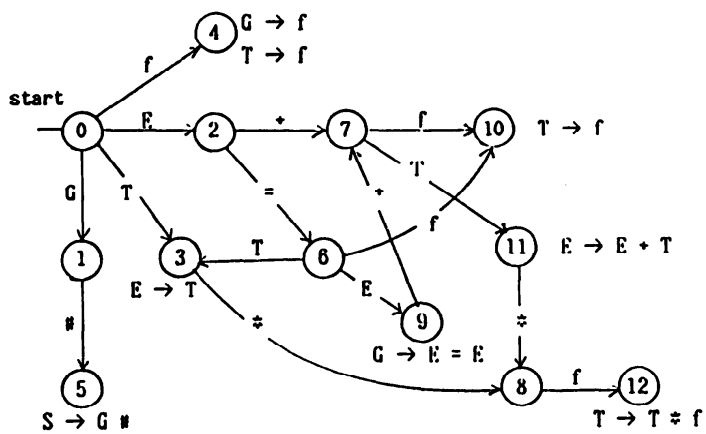

Fig. 5 State transition diagram of the LR(0) automaton in Fig. 4.

		(I_ℓ, Y)					
I_k	(I_0, S)	(I_0, G)	(I_0, E)	(I_0, T)	(I_6, E)	(I_6, T)	(I_7, T)
I_0	x_{S1}	x_{G1}	x_{E1}	x_{T1}	-	-	-
I_1	x_{S2}	-	-	-	-	-	-
I_2	-	x_{G2}	x_{E3}	-	-	-	-
I_3#	-	-	x_{E2}#	x_{T3}	x_{E2}#	x_{T3}	-
I_4#	-	x_{G5}#	-	x_{T2}#	-	-	-
I_5A	x_{S3}A	-	-	-	-	-	-
I_6	-	x_{G3}	-	-	x_{E1}	x_{T1}	-
I_7	-	-	x_{E4}	-	x_{E4}	-	x_{T1}
I_8	-	-	-	x_{T4}	-	x_{T4}	x_{T4}
I_9#	-	x_{G4}#	-	-	x_{E3}	-	-
I_{10}#	-	-	-	-	-	x_{T2}#	x_{T2}#
I_{11}#	-	-	x_{E5}#	-	x_{E5}#	-	x_{T3}
I_{12}#	-	-	-	x_{E5}#	-	x_{E5}#	x_{T5}#

Fig. 6 Embedding Table.

ding, $\mathscr{A}_E$ derived from $E \to T \mid E + T$ is embedded for (I_0, E) as $I_0 - \overset{I}{-} I_3$ and $I_0 - \overset{E}{-} I_2 - \overset{+}{-} I_7 - \overset{I}{-} I_{11}$ and for (I_6, E) as $I_6 - \overset{I}{-} I_3$ and $I_6 - \overset{E}{-} I_9 - \overset{+}{-} I_7 - \overset{I}{-} I_{11}$. In the former occurrence of embedding $\mathscr{A}_E$, we have three nonterminal-transitions (I_0, T), (I_0, E), and (I_7, T) and similarly in the latter, we have (I_6, T), (I_6, E), and (I_7, T). However, for (I_0, E) and (I_6, E), $\mathscr{A}_E$ has been already embedded. Accordingly, for the only three nonterminal-transitions (I_0, T), (I_6, T), and (I_7, T), it is necessary to embed $\mathscr{A}_T$ derived from $T \to f \mid T * f$. Thus, $\mathscr{A}_T$ is embedded for (I_0, T) as $I_0 - \overset{f}{-} I_4$ and $I_0 - \overset{I}{-} I_3 - \overset{*}{-} I_8 - \overset{f}{-} I_{12}$, for (I_6, T) as $I_6 - \overset{f}{-} I_{10}$ and $I_6 - \overset{I}{-} I_3 - \overset{*}{-} I_8 - \overset{f}{-} I_{12}$, and furthermore for (I_7, T) as $I_7 - \overset{f}{-} I_{10}$ and $I_7 - \overset{I}{-} I_{11} - \overset{*}{-} I_8 - \overset{f}{-} I_{12}$. Because there is no more new nonterminal-transition, no more embedding occurs. (If we replace the word "embed" by "generate" in the above, one more method of constructing the LR(0) automaton will be found.)

Linear algebra-like Approach

Our linear algebra-like approach follows a similar course. For each $\mathscr{A}_X$ associated with the X-defining V-RLES $x_X = A_X x_X + c_X$, each subset I_k of S_X is represented as the so-called set characteristic vector, i.e. an n_X-dimensional bit vector such that for each state $x_{Xi} \in S_X$, if I_{kX} contains the state x_{Xi}, then the i-th bit of the vector is λ, otherwise ϕ. This bit vector is also denoted here by I_{kX}. Each state I_k of $\mathscr{A}_{LR}$ is specified here by a list of all I_{kX}, $X \in N$, as shown below:

$$I_k = (I_{kX_0} \cdots I_{kX} \cdots I_{kZ}), \quad k = 0, 1, \cdots, K \tag{5.9}$$

That is, I_k is an n_{LR}-dimensional bit vector, where $n_{LR} = \Sigma_{X \in N} n_X$. Here we denote the n-dimensional bit vector space by $b^{(n)}$. Two functions, Trans: $b^{(n_{LR})} \times V \to b^{(n_{LR})}$ and Closure: $b^{(n_{LR})} \to b^{(n_{LR})}$, are defined by using two kinds of functions, Υ_X: $b^{(n_X)} \times V \to b^{(n_X)}$ and Ψ_X: $b^{(n_{LR})} \to b^{(n_X)}$, respectively, as follows:

$$\text{Trans}(I_k, S) = J_k = (J_{kX_0} \cdots J_{kX} \cdots J_{kZ}) \tag{5.10}$$

$$\text{where } J_{kX} = \Upsilon_X(I_{kX}, s) = I_{kX} \partial_s A_X, \tag{5.11}$$

$$\text{Closure}(J_k) = J'_k = (J'_{kX_0} \cdots J'_{kX} \cdots J'_{kZ}) \tag{5.12}$$

$$\text{where } J'_{kX} = \Psi_X(J_k) = J_{kX} + \Sigma_{Y \in N} J'_{kY} \partial_X A_Y \lambda_Y i_X \tag{5.13}$$

In order to solve Eq. (5.12), put

$$H = (h_{XY}), \quad h_{XY} = \partial_X A_Y \lambda_Y \tag{5.14}$$

$$D = (d_{XY}), \quad d_{XY} = i_Y \text{ if } X = Y, \quad = \phi_Y \text{ otherwise} \tag{5.15}$$

Then, from Eqs. (5.12) and (5.13), we have

$$J'_k = J_k + J'_k HD \tag{5.16}$$

and have the solution

$$J'_k = J_k (HD)^* = J_k + J_k H \Gamma'^* D \tag{5.17}$$

$$\text{where } \Gamma' = DH = (\gamma'_{XY}), \quad \gamma'_{XY} = i_X \partial_Y A_X \lambda_X \tag{5.18}$$

Note that γ'_{XY} is the same as Eq. (4.9). Thus, Eqs. (5.16) and (5.17) correspond to Eqs. (5.3) and (5.4), respectively.

The transition function Goto: $b^{(n_{LR})} \times V \to b^{(n_{LR})}$ is given as

$$\text{Goto}(I_k, s) = \text{Closure}(J_k) = J_k + J_k H \Gamma'^* D \tag{5.19}$$

$$\text{where } J_k = \text{Trans}(I_k, s) = I_k \partial_s G \tag{5.20}$$

$$G = (g_{XY}), \quad g_{XY} = A_X \text{ if } X = Y, \quad = \Phi_{X,Y} \text{ otherwise.}$$

The initial state of $\mathscr{A}_{LR}$ is defined as

$$I_0 = J_0 + J_0 H \Gamma'^* D \tag{5.21}$$

where $J_0 = (i_{X_0} \phi_{X'} \cdots \phi_X \cdots \phi_Z)$.

If a state I_k contains I_{kX} such that

$$I_{kX} c_X = \lambda \tag{5.22}$$

then the state I_k is a reduced state in $\mathscr{A}_{LR}$.

Example 4. $\mathscr{A}_{LR}$ in Example 3 is computed from V-RLES in Fig. 3. From Eqs. (5.14) and (5.15) and Example 2, we can compute H, Γ'^* and $H\Gamma'^* D$, as shown in Fig. 7. Then, from Eqs. (5.19), (5.20), and (5.21), the desired $\mathscr{A}_{KR}$ is computed as shown in Fig. 8.

In this approach, the Embedding table shown in Fig. 6 is changed into a λ-matrix in such a manner that if its element contains a state, it is replaced by λ. This λ-matrix is called an Embedding matrix and denoted by $M = (\mu_{I_k,(I_l,Y)})$.

Note that for each nonterminal-transition (I_l, Y), $\mathscr{A}_Y$ is embedded from state I_l once and only once. That is, for each nonterminal-transition (I_l, Y), we have only one column (I_l, Y). In the column, it holds that $\mu_{I_k,(I_l,Y)} = \lambda$ and $I_{lY} i_Y = \lambda$ (i.e. the first bit of I_{lY} is λ), because I_l contains the initial state of $\mathscr{A}_Y$. Furthermore, if it holds

$$H = \begin{bmatrix} \phi & \lambda & \phi & \phi \\ \phi & \phi & \phi & \phi \\ \phi & \phi & \phi & \phi \\ \phi & \phi & \lambda & \phi \\ \phi & \phi & \phi & \phi \\ \phi & \phi & \lambda & \phi \\ \phi & \phi & \phi & \phi \\ \phi & \phi & \phi & \phi \\ \phi & \phi & \lambda & \phi \\ \phi & \phi & \phi & \phi \\ \phi & \phi & \phi & \phi \\ \phi & \phi & \phi & \lambda \\ \phi & \phi & \phi & \phi \\ \phi & \phi & \phi & \lambda \\ \phi & \phi & \phi & \phi \\ \phi & \phi & \phi & \phi \\ \phi & \phi & \phi & \phi \end{bmatrix}, \quad \Gamma^{\mp} = \begin{bmatrix} \lambda & \lambda & \lambda & \lambda \\ \phi & \lambda & \lambda & \lambda \\ \phi & \phi & \lambda & \lambda \\ \phi & \phi & \phi & \lambda \end{bmatrix},$$

$$H\Gamma^{\mp}D = \begin{bmatrix} \phi & \lambda & \lambda & \lambda \\ \phi & \phi & \phi & \phi \\ \phi & \phi & \phi & \phi \\ \phi & \phi & \lambda & \lambda \\ \phi & \phi & \phi & \phi \\ \phi & \phi & \lambda & \lambda \\ \phi & \phi & \phi & \phi \\ \phi & \phi & \phi & \phi \\ \phi & \phi & \lambda & \lambda \\ \phi & \phi & \phi & \phi \\ \phi & \phi & \phi & \phi \\ \phi & \phi & \phi & \lambda \\ \phi & \phi & \phi & \phi \\ \phi & \phi & \phi & \lambda \\ \phi & \phi & \phi & \phi \\ \phi & \phi & \phi & \phi \\ \phi & \phi & \phi & \phi \end{bmatrix} \begin{bmatrix} i_S & \phi_G & \phi_E & \phi_T \\ \phi_S & i_G & \phi_E & \phi_T \\ \phi_S & \phi_G & i_E & \phi_T \\ \phi_S & \phi_G & \phi_E & i_T \end{bmatrix}$$

Fig. 7 Computation of $H\Gamma^{*}D$ for V-RLES in Example 1.

$I_k = (\ I_{kS} \quad I_{kG} \quad I_{kE} \quad I_{kT}\ )$

$J_0 = (\lambda\phi\phi\ \ \phi\phi\phi\phi\phi\ \ \phi\phi\phi\phi\phi\ \ \phi\phi\phi\phi\phi\phi)$

$\Rightarrow (\lambda\phi\phi\ \ \lambda\phi\phi\phi\phi\ \ \lambda\phi\phi\phi\phi\ \ \lambda\phi\phi\phi\phi\phi) = I_0$

$\text{Trans}(I_0, G) = (\phi\lambda\phi\ \ \phi\phi\phi\phi\phi\ \ \phi\phi\phi\phi\phi\ \ \phi\phi\phi\phi\phi) = \text{Goto}(I_0, G) = I_1$

$\text{Trans}(I_0, E) = (\phi\phi\phi\ \ \phi\lambda\phi\phi\phi\ \ \phi\phi\lambda\phi\phi\ \ \phi\phi\phi\phi\phi) = \text{Goto}(I_0, E) = I_2$

$\text{Trans}(I_0, T) = (\phi\phi\phi\ \ \phi\phi\phi\phi\phi\ \ \phi\lambda\phi\phi\phi\ \ \phi\phi\lambda\phi\phi) = \text{Goto}(I_0, T) = I_3^{\#}$

$\text{Trans}(I_0, f) = (\phi\phi\phi\ \ \phi\phi\phi\phi\lambda\ \ \phi\phi\phi\phi\phi\ \ \phi\lambda\phi\phi\phi) = \text{Goto}(I_0, f) = I_4^{\#}$

$\text{Trans}(I_1, \#) = (\phi\phi\lambda\ \ \phi\phi\phi\phi\phi\ \ \phi\phi\phi\phi\phi\ \ \phi\phi\phi\phi\phi) = \text{Goto}(I_1, \#) = I_5^{A}$

$\text{Trans}(I_2, =) = (\phi\phi\phi\ \ \phi\phi\lambda\phi\phi\ \ \phi\phi\phi\phi\phi\ \ \phi\phi\phi\phi\phi)$

$\Rightarrow (\phi\phi\phi\ \ \phi\phi\lambda\phi\phi\ \ \lambda\phi\phi\phi\phi\ \ \lambda\phi\phi\phi\phi) = \text{Goto}(I_3, =) = I_6$

$\text{Trans}(I_2, +) = (\phi\phi\phi\ \ \phi\phi\phi\phi\phi\ \ \phi\phi\phi\lambda\phi\ \ \phi\phi\phi\phi\phi)$

$\Rightarrow (\phi\phi\phi\ \ \phi\phi\phi\phi\phi\ \ \phi\phi\phi\lambda\phi\ \ \lambda\phi\phi\phi\phi) = \text{Goto}(I_3, +) = I_7$

$\text{Trans}(I_3, \mp) = (\phi\phi\phi\ \ \phi\phi\phi\phi\phi\ \ \phi\phi\phi\phi\phi\ \ \phi\phi\phi\lambda\phi) = \text{Goto}(I_4, \mp) = I_8$

$\text{Trans}(I_6, E) = (\phi\phi\phi\ \ \phi\phi\phi\lambda\phi\ \ \phi\phi\lambda\phi\phi\ \ \phi\phi\phi\phi\phi) = \text{Goto}(I_6, E) = I_9^{\#}$

$\text{Trans}(I_6, f) = (\phi\phi\phi\ \ \phi\phi\phi\phi\phi\ \ \phi\phi\phi\phi\phi\ \ \phi\lambda\phi\phi\phi) = \text{Goto}(I_6, f) = I_{10}^{\#}$

$\text{Trans}(I_7, T) = (\phi\phi\phi\ \ \phi\phi\phi\phi\phi\ \ \phi\phi\phi\phi\lambda\ \ \phi\phi\lambda\phi\phi) = \text{Goto}(I_7, T) = I_{11}^{\#}$

$\text{Trans}(I_8, f) = (\phi\phi\phi\ \ \phi\phi\phi\phi\phi\ \ \phi\phi\phi\phi\phi\ \ \phi\phi\phi\phi\lambda) = \text{Goto}(I_8, f) = I_{12}^{\#}$

$\text{Trans}(I_6, T) = \text{Goto}(I_6, T) = I_3^{\#}, \quad \text{Trans}(I_7, f) = \text{Goto}(I_7, f) = I_{10}^{\#}$

$\text{Trans}(I_9, +) = \text{Goto}(I_9, +) = I_7, \quad \text{Trans}(I_{11}, \mp) = \text{Goto}(I_{11}, \mp) = I_8.$

Fig. 8 Transition function of the LR(0) automaton obtained from Fig. 3.

that $\mu_{I_k,(I_\ell,Y)}=\lambda$ and $I_{kY}c_Y=\lambda$, I_k is a reduced state in $\mathscr{A}_{LR}$.

6. Follow Sets

For a given LR(0) automaton $\mathscr{A}_{LR}$, a formula for computing Follow sets is given in the same way as in Section 4.

Let Ω be a set of all nonterminal-transitions in $\mathscr{A}_{LR}$. Then for each nonterminal-transition $(I_k, X)\in\Omega$, a set of terminals is defined as follows, and is called the Follow set of (I_k, X).

$$\text{Follow}(I_k, X)=\{t\in T \mid X_0 \overset{*}{\underset{R}{\Rightarrow}} \theta Xtw, \quad \theta \text{ accesses state } I_k,\ \theta\in V^*,\ w\in T^*\} \tag{6.1}$$

where $\overset{*}{\underset{R}{\Rightarrow}}$ indicates the rightmost derivation.

DeRemer and Pennello [1] derived from the definition two types of inclusive relations related to Follow sets and gave a method for obtaining the sets concretely, based on inclusive relations, in the manner of recursive computation of sets of symbols. The method given below first derives simulaneous equations with Follow sets as the unknown sets, and then solves them. The solution is of the product form of the closure of a λ-matrix and a T-vector.

There are generally two types of production rules:

(1) $Y\to\alpha X\xi s\beta$, $\alpha, \beta\in V^*$, $\xi\in N'^*$, (6.2)

(2) $Y\to\alpha X\xi$, $\alpha\in V^*$, $\xi\in N'^*$ (6.3)

where N' is the set of ε-generating nonterminals.

They introduce the following two types of situation in the derivation sequences:

For (1), there is $X_0 \overset{*}{\underset{R}{\Rightarrow}} \delta Yw \underset{R}{\Rightarrow} \delta\alpha X\xi s\beta w$, $\delta\in V^*$, $w\in T^*$ and for (2), there is $X_0 \overset{*}{\underset{R}{\Rightarrow}} \delta Yw \underset{R}{\Rightarrow} \delta\alpha X\xi w$, $\delta\in V^*$, $w\in T^*$.

Similarly, in the related LR(0) automaton $\mathscr{A}_{LR}$, the two types are derived as shown in Fig. 9 and Fig. 10.

For the first type, we have the following proposition.

Proposition 6.1 The following two conditions are equivalent.

—There exists a rule $Y\to\alpha X\xi s\beta$, $\alpha, \beta\in V^*$, $\xi\in N'^*$.

—There exist an occurrence of embedding $\mathscr{A}_Y$ and a state I_k of $\mathscr{A}_{LR}$ such that

$$I_{kY}\partial_X A_Y C_Y'^* \partial_s A_Y \lambda_Y=\lambda \tag{6.4}$$

where C_Y' is the N'-adjacent matrix of A_Y. Figure 9 shows this situation and its relation to the definition of Follow sets. That is, if Eq. (6.4) holds, we can expect to have the following relation:

$$\text{Follow}(I_k, X)\supseteq \text{First}(s) \tag{6.5}$$

The above relation is shown with condition (6.4) as follows:

$$\text{Follow}(I_k, X)\supseteq I_{kY}\partial_X A_Y C_Y'^* \partial_s A_Y \lambda_Y \cdot \text{First}(s) \tag{6.6}$$

Because this relation is valid for any $Y\in N$ and any $s\in V$ and any $s\in V$, it holds that

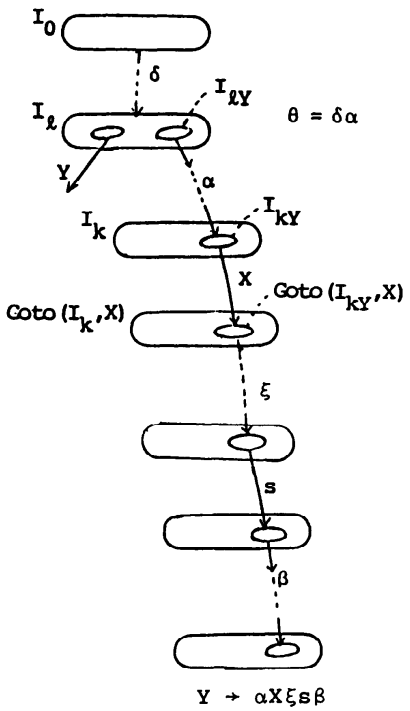

Fig. 9 Embedding of automaton Y in case 1.

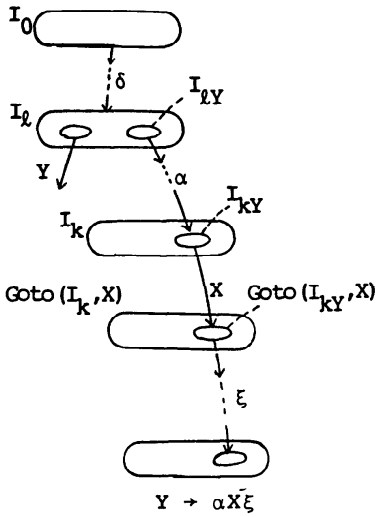

Fig. 10 Embedding of automaton Y in case 2.

$$\text{Follow}(I_k, X)\supseteq \sum_{Y\in N}\sum_{s\in V} I_{kY}\partial_X A_Y C_Y'^* \partial_s A_Y \lambda_Y \text{ First}(s)$$

Here we put

$$b_Y=\Sigma_{s\in V} C_Y'^* \partial_s A_Y \lambda_Y \text{ First}(s) \tag{6.7}$$

$$d_{(I_k,X)}=\Sigma_{Y\in N} I_{kY}\partial_X A_Y b_Y \tag{6.8}$$

$$\text{Follow}(I_k, X)\supseteq d_{(I_k,X)} \tag{6.9}$$

Note that $I_{kY}\partial_X A_Y$ is a λ-vector $\Upsilon_Y(I_{kY}, X)$ given by Eq. (5.11). Therefore, we have

$$d_{(I_k,X)}=\Sigma_{Y\in N}\Upsilon_Y(I_{kY}, X)b_Y \tag{6.10}$$

The next proposition is given for the second type of situation.

Proposition 6.2 The following three statements are all equivalent:

—There is a rule $Y\to\alpha X\xi$, $\alpha\in V^*$, $\xi\in N'^*$.

—There are two nonterminal-transitions (I_k, X) and (I_l, Y) and there is an occurrence of embedding $\mathscr{A}_Y$ of which the initial state is embedded in I_l and of which some state is embedded in I_k in such a way that at least one of the final states of $\mathscr{A}_Y$ is embedded in state $\text{Goto}(\cdots\text{Goto}(\text{Goto}(I_k, X), s_1),\cdots), s_r)$, $s_1 s_2\cdots s_r=\xi$, $r\geqq 1$, or state $\text{Goto}(I_k, X)$, which is reducible the rule $Y\to\alpha X\xi$.

—There are two nonterminal-transitions (I_k, X) and (I_l, Y) such that $\mu_{I_k,(I_l,Y)}=\lambda$ and $I_{kY}\partial_X A_Y C'^*_Y c_Y=\lambda$.

Figure 10 shows the above situation and relations to the definition of Follow sets. That is, if the above condition holds, we can expect to have the following relation:

$$\text{Follow}(I_k, X)\supseteq\text{Follow}(I_l, Y) \tag{6.11}$$

Here, we define a function $u:\Omega\times\Omega\to P\lambda$, as follows:

$\theta_{(I_k,X),(I_l,Y)}=\lambda$ if the condition of Proposition 6.2 holds, i.e. if it holds that $e_{I_k,(I_l,Y)}=\lambda$ and $I_{kY}\partial_X A_Y C'^*_Y c_Y=\lambda$,

$=\phi$ otherwise.

Then the above relation (6.11) is given as shown below:

$$\text{Follow}(I_k, X)\supseteq\sum_{(I_l,Y)\in\Omega}\theta_{(I_k,X),(I_l,Y)}\,\text{Follow}(I_l, Y) \tag{6.12}$$

Here, we define a $\#(\Omega)\times\#(\Omega)$ λ-matrix as

$$\Theta=(\gamma_{(I_k,X),(I_l,Y)}),\quad (I_k, X), (I_l, Y)\in\Omega,$$

and call it a Direct Inclusion matrix. This matrix Θ for $\mathscr{A}_{LR}$ in Example 4 is shown in Fig. 11.

i	(I_k, X) \ (I_ℓ, Y)	1	2	3	4	5	6
		(I_0, G)	(I_0, E)	(I_0, T)	(I_6, E)	(I_6, T)	(I_7, T)
1	(I_0, G)						
2	(I_0, E)						
3	(I_0, T)		λ				
4	(I_6, E)	λ					
5	(I_6, T)				λ		
6	(I_7, T)		λ		λ		

Fig. 11 Direct Inclusion Matrix: Θ.

For Ω of $\mathscr{A}_{LR}$, the Direct Inclusion matrix Θ is constructed as follows: For each nonterminal-transition $(I_l, Y)\in\Omega$, a new row (I_l, Y) and a new column (I_l, Y) are made. Note that embedding of $\mathscr{A}_Y$ starts newly from state I_l. Each element in this matrix is initialized as ϕ. For each nonterminal-transition (I_k, X), if state $\text{Goto}(I_k, X)$ contains a final state of $\mathscr{A}_Y$ embedded from I_l, then λ is assigned to $\theta_{(I_k,X),(I_l,Y)}$.

From Eqs. (6.9) and (6.12), we have the following equations:

$$\text{Follow}(I_k, X)=\sum_{(I_l,Y)\in\Omega}\theta_{(I_k,X),(I_l,Y)}\,\text{Follow}(I_l, Y)+d_{(I_k,X)} \tag{6.13}$$

To solve Eq. (6.13), put

$$u=(u_{(I_k,X)}),\quad u_{(I_k,X)}=\text{Follow}(I_k, X)$$
$$d=(d_{(I_k,X)}),$$

Then, the desired Follow sets are obtained as follows:

$$u=\Theta u+d=\Theta^* d \tag{6.14}$$

Example 5. Computation of Follow sets for $\mathscr{A}_{LR}$ in Example 4.

For six nonterminal-transitions (I_0, G), (I_0, E), (I_0, T), (I_6, E), (I_6, T) and (I_7, T) in $\mathscr{A}_{LR}$, a six-dimensional T-vector d and a 6×6 λ-matrix Θ (Fig. 11) are necessary. The vector d needs a T-vector b, which is

$$b=\begin{bmatrix} b_S\\ b_G\\ b_E\\ b_T\end{bmatrix},\quad \text{where } b_S=\begin{bmatrix}\{f\}\\ \{*\}\\ \phi\end{bmatrix},\quad b_G=\begin{bmatrix}\{f\}\\ \{=\}\\ \{f\}\\ \phi\\ \phi\end{bmatrix},\quad b_E=\begin{bmatrix}\{f\}\\ \phi\\ \{+\}\\ \{f\}\\ \phi\end{bmatrix},\quad b_T=\begin{bmatrix}\{f\}\\ \phi\\ \{*\}\\ \{f\}\\ \phi\end{bmatrix}.$$

$$d=\begin{bmatrix} d(I_0, G)\\ d(I_0, E)\\ d(I_0, T)\\ d(I_6, E)\\ d(I_6, T)\\ d(I_7, T)\end{bmatrix}=\begin{bmatrix}\text{Trans}(I_0, G)\\ \text{Trans}(I_0, E)\\ \text{Trans}(I_0, T)\\ \text{Trans}(I_6, E)\\ \text{Trans}(I_6, T)\\ \text{Trans}(I_7, T)\end{bmatrix}\quad b=\begin{bmatrix}\{\#\}\\ \{=, +\}\\ \{*\}\\ \{+\}\\ \{*\}\\ \{*\}\end{bmatrix},\quad \Theta=\begin{bmatrix}\phi&\phi&\phi&\phi&\phi&\phi\\ \phi&\phi&\phi&\phi&\phi&\phi\\ \phi&\lambda&\phi&\phi&\phi&\phi\\ \lambda&\phi&\phi&\phi&\phi&\phi\\ \phi&\phi&\phi&\lambda&\phi&\phi\\ \phi&\lambda&\phi&\lambda&\phi&\phi\end{bmatrix}$$

Thus we have the desired Follow sets as follows:

$$u=\begin{bmatrix}\text{Folooow } (I_0, G)\\ \text{Folooow } (I_0, E)\\ \text{Folooow } (I_0, T)\\ \text{Folooow } (I_6, E)\\ \text{Folooow } (I_6, T)\\ \text{Folooow } (I_7, T)\end{bmatrix}=\Theta^* d=\begin{bmatrix}\lambda & \phi & \phi & \phi & \phi & \phi\\ \phi & \lambda & \phi & \phi & \phi & \phi\\ \phi & \lambda & \lambda & \phi & \phi & \phi\\ \lambda & \phi & \phi & \lambda & \phi & \phi\\ \lambda & \phi & \phi & \lambda & \lambda & \phi\\ \lambda & \lambda & \phi & \lambda & \phi & \lambda\end{bmatrix}\begin{bmatrix}\{\#\}\\ \{=,+\}\\ \{*\}\\ \{+\}\\ \{*\}\\ \{*\}\end{bmatrix}=\begin{bmatrix}\{\# \quad\}\\ \{\ =,+\ \}\\ \{\ =,+,*\}\\ \{\#,\quad +\ \}\\ \{\#,\quad +,*\}\\ \{\#,=,+,*\}\end{bmatrix}$$

composed of b_S, b_G, b_E, and b_T obtained from Eq. (6.7), and the value Trans(I_k, X) for (I_k, X)$\in \Omega$ (Fig. 8).

7. Look-Ahead Sets

LALR(1) parsers use sets of symbols called Look-Ahead sets to make parsing deterministic. The sets are defined for each pair of a reduce state I_k and the reduced rule $X\to\alpha$, as follows:

$$\mathrm{LA}(I_k, X\to\alpha)=\{t\in T \mid X_0 \overset{*}{\underset{R}{\Rightarrow}} \delta X t w,\ \delta\in V^*,\ w\in T^*,\ \delta\alpha \text{ accesses } I_k\} \tag{7.1}$$

$$=A \text{ union of all Follow}(I_l, X) \text{ such that } I_l \text{ transits to } I_k \text{ by } \alpha. \tag{7.2}$$

Figure 12 shows the situation expressed by the definition. We thus have the following proposition:

Proposition 7.1 The following three statements are all equivalent:

—I_l transits to I_k by α for a rule $X\to\alpha$.

—There exists an occurrence of embedding $\mathscr{A}_X$ of which the initial state is embedded in I_l and of which a final state is embedded in I_k and the initial state transits to the final state by α in $\mathscr{A}_X$.

$$-\mu_{I_k,(I_l,X)}=\lambda \quad \text{and} \quad I_{kX}c_X=\lambda \tag{7.3}$$

From the embedding matrix $M=(\mu_{I_k,(I_l,X)})$ given from the embedding table in Section 5, we obtain here a matrix in such a manner that if I_k is not a reduced state, the I_k-th row is removed. Note that only reduced states, that satisfy Eq. (7.3) are left. Then the matrix is altered as shown below: The row index I_k is replaced by a pair of I_k and the rule reducible there. However, there may be a state reducible concerned with two or more rules. For such states, we expand the matrix to row-direction in such a manner that if I_k is reducible for rules $X\to\alpha$, $X'\to\alpha'$, $\cdots$, $X''\to\alpha''$, we generate state-rule pairs (I_k,

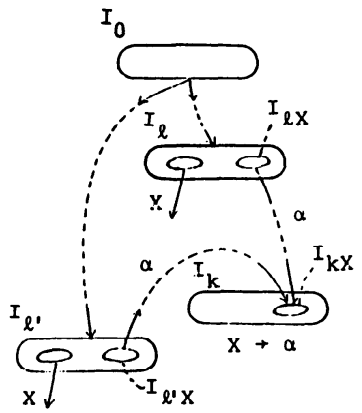

Fig. 12 Relation of a rule $X\to\alpha$ and states in an LR(0) automaton.

	(I_ℓ, Y)	1	2	3	4	5	6
	(I_k , X→α)	(I_0,G)	(I_0,E)	(I_0,T)	(I_6,E)	(I_6,T)	(I_7,T)
1	(I_3 , E→T)		λ		λ		
2	(I_4 , G→f)	λ					
3	(I_4 , T→f)			λ			
4	(I_9 , G→E=E)	λ					
5	(I_{10}, T→f)					λ	λ
6	(I_{11}, E→E+T)		λ		λ		
7	(I_{12}, T→T*f)			λ		λ	λ

Fig. 13 Look-Back Matrix: Δ.

$X\to\alpha$), $(I_k, X'\to\alpha'), \cdots, (I_k, X''\to\alpha'')$ and replace the I_k-th row by $(I_k, X\to\alpha)$-th, $(I_k, X'\to\alpha')$-th, $\cdots$, $(I_k, X''\to\alpha'')$-th rows so as to satisfy the following condition. The obtained λ-matrix is shown as

$$\Delta=(\delta_{(I_k,X\to\alpha),(I_l,Y)}) \tag{7.4}$$

where $\delta_{(I_k,X\to\alpha),(I_l,Y)}=\mu_{I_k,(I_l,Y)}$ if $X=Y$,

$$=\phi \qquad \text{otherwise.} \tag{7.5}$$

and is called the Look-Back matrix.

From the Embedding matrix M obtained from Fig. 6, we obtained Look-Back matrix Δ shown in Fig. 13.

Look-Back matrix Δ is also directly constructed as follows: For each nonterminal-transition (I_l, Y) in $\mathscr{A}_{\mathrm{LR}}$, a new column (I_l, Y) is made. Each column indexed by (I_l, Y) corresponds to an occurrence of embedding of $\mathscr{A}_Y$ started from I_l. For each pair of a reduced state I_k and the reduced rule $X\to\alpha$, a new row $(I_k, X\to\alpha)$ is also made. If I_k is the state where a rule $X\to\alpha$ must be reduced (i.e. for the embedded automaton $\mathscr{A}_X$ associated to a column (I_l, X), if a final state I_{kX} is accessed from the initial state I_{lX} by α), λ is assigned to the associated element $\delta_{(I_k,X\to\alpha),(I_l,X)}$.

Using the Look-Back matrix Δ, Look-Ahead sets are defined as

$$\mathrm{LA}(I_k, X\to\alpha)=\sum_{(I_l,Y)\in\Omega}\delta_{(I_k,X\to\alpha),(I_l,Y)}\,\mathrm{Follow}(I_l, Y) \tag{7.6}$$

In order to solve them, put

$$v=(\mathrm{LA}(I_k, X\to\alpha)),$$

$$u=(u_{(I_l,Y)}),\ u_{(I_l,Y)}=\mathrm{Follow}(I_l, Y),$$

The desired Look-Ahead sets are then obtained from Eqs. (7.6) and (6.14), as follows:

$$v=\Delta u=\Delta\Theta^*\delta \tag{7.7}$$

Example 6. Computation of Look-Ahead sets for $\mathscr{A}_{\mathrm{LR}}$ in Example 4.

From Eq. (7.7), Fig. 13 and Example 5, we have

$$v=(\mathrm{LA}(I_k, x\to\alpha))=\Delta\Theta^*d$$

$$=\begin{bmatrix}\mathrm{LA}(I_3, E\to T)\\ \mathrm{LA}(I_4, G\to f)\\ \mathrm{LA}(I_4, T\to f)\\ \mathrm{LA}(I_9, G\to E=E)\\ \mathrm{LA}(I_{10}, T\to f)\\ \mathrm{LA}(I_{11}, E\to E+T)\\ \mathrm{LA}(I_{12}, T\to T*f)\end{bmatrix}=\begin{bmatrix}\phi&\lambda&\phi&\lambda&\phi&\phi\\ \lambda&\phi&\phi&\phi&\phi&\phi\\ \phi&\phi&\lambda&\phi&\phi&\phi\\ \lambda&\phi&\phi&\phi&\phi&\phi\\ \phi&\phi&\phi&\phi&\lambda&\lambda\\ \phi&\lambda&\phi&\lambda&\phi&\phi\\ \phi&\phi&\lambda&\phi&\lambda&\lambda\end{bmatrix}\begin{bmatrix}\lambda&\phi&\phi&\phi&\phi&\phi\\ \phi&\lambda&\phi&\phi&\phi&\phi\\ \phi&\lambda&\lambda&\phi&\phi&\phi\\ \lambda&\phi&\phi&\lambda&\phi&\phi\\ \lambda&\phi&\phi&\lambda&\lambda&\phi\\ \lambda&\lambda&\phi&\lambda&\phi&\lambda\end{bmatrix}\begin{bmatrix}\{\#\}\\ \{=,+\}\\ \{*\}\\ \{+\}\\ \{*\}\\ \{*\}\end{bmatrix}=\begin{bmatrix}\{\#,=,+,\}\\ \{\#\}\\ \{+,*\}\\ \{\#\}\\ \{\#,=,+,*\}\\ \{\#,=,+,\}\\ \{\#,=,+,*\}\end{bmatrix},$$

8. Conclusion

This paper showed a new method for computing, from a given grammar, an LR(0) automaton and the LALR(1) Look-Ahead sets as an application of a new methodology, a linear algebra-like approach to the language semi-ring. The method is based on the recognition that the language space generated from the empty alphabet is isomorphic to Boolean algebra and on the methodology for solving problems by partially reducing them to problems of Boolean algebra.

Traditionally, these problems are generally formalized so that simultaneous equations concerned with unknown sets $s_1, s_2, \cdots, s_n$ are derived as follows:

$$s_i=s_{i_1}\cup s_{i_2}\cup\cdots\cup s_{i_n}\cup d_i \text{ for } i=1, 2, \cdots, n,$$

where $1\leqq i_1, i_2, \cdots, i_n\leqq n$,

and are solved in such a manner that all the unknown sets $s_1, s_2, \cdots, s_n$ are initialized as ϕ and then computed by using the equations iteratively until the sets obtained become unchanged.

Our method introduced into the equations the coefficients γ_{ij}, $1\leqq i, j\leqq n$, which take the value λ or ϕ, as follows:

$$s_i=\gamma_{i1}s_1+\gamma_{i2}s_2+\cdots+\gamma_{in}s_n+d_i$$

for $i=1, 2, \cdots, n$.

or

$$s=\Gamma s+d$$

where

$$s={}^t(s_1, s_2, \cdots, s_n),$$

$$d={}^t(d_1, d_2, \cdots, d_n),$$

$$\Gamma=(\gamma_{ij})$$

For the equation, the traditional method is represented as the following computation of symbol sets:

(1) Set $s\leftarrow d$.

(2) Compute $s\leftarrow\Gamma s+d$ until the value s becomes unchanged.

It is known that the obtained $s=\Gamma(\Gamma\cdots(\Gamma d+d)+\cdots)+d)+d$ is the minimal solution, i.e. the minimal fixed point, of the equation, which is simply written as $s=\Gamma^*d$. Therefore if we compute the closure of the Boolean matrix Γ^* first, computation of sets of sym-

bols is done only for $(\Gamma^*)d$. Now, let us give a method for computing Γ^*:

(1) Set $M \leftarrow E$,

(2) Compute $M \leftarrow M\Gamma + M$ until the value of M becomes unchanged.

This method may be said to be equivalent to the traditional one if we ignore the difference between the computation of symbol sets and that of Boolean numbers.

It is, however, well-known that $\Gamma^* = E + \Gamma^+$ and in order to compute Γ^+, we have effective methods such as Warshall's theorem. This is the basic reason why our method is more effective than traditional ones.

The above method was commonly used for obtaining First sets, Follow sets, and LR(0) automaton. (Each Look-Ahead set is given as a union of properly selected Follow sets). The next problem was how to compute the coefficients γ_{ij}, $1 \leqq i, j \leqq n$, and the constant terms d_i, $1 \leqq i \leqq n$, of the equations. From the above discussion, it will be seen that they can be obtained by traditional methods.

In this paper, however, we showed again the method for computing them by Booloean algebra. That is, for each nonterminal X, the structure of the right part of the X-defining BNF was represented as a V-matrix A_X and a λ-vector (i.e. Boolean vector) c_X, and for each $s \in V$, the structure depending on s was abstracted from A_X by means of the operator ∂_s as a λ-matrix (i.e. Boolean matrix) $\partial_s A_X$. The coefficients γ_{ij}, $1 \leqq i, j \leqq n$, and the constant terms d_i, $1 \leqq i \leqq n$, were computed by using these Boolean matrices and vectors.

A_X, whose size is nearly equal to the length of the X-defining BNF, is generally sparse, and $\delta_s A_X$ is more so. How to implement matrices of the sparse kind is well known in numerical analysis. It is easy to give, if neccesary, a procedure for obtaining a compact form of $\partial_s A_X$ from the X-defining BNF directly without using A_X. Let B be an array containing the right part (whose length is n) of the X-defining BNF, and let C be an array containing a pair of indices i and j such that the (i, j)-th element of $\partial_s A_X$ is λ. The desired procedure is then given as follows:

```
f:=0; k:=1; m:=0; t:=1;
while t<=n do begin
  if B[t]='|' then f:=0
  else begin
    k:=k+1;
    if B[t]='s' then begin
      m:=m+1;
      if f=0 then C[m, 1]:=1 else C[m, 1]:=k-1;
      f:=1; C[m, 2]:=k
    end
  end;
  t:=t+1
end;
```

This means that the method of representing a given problem as algebraic expressions is different from the method of implementing those expressions, that is, from the method of transforming them into a system composed of procedures and data structures so as to adapt it to the given circumstances. The aim of this paper is, of course, the former.

Generally speaking, LR(0) automata as well as recursive descent parsers can be regarded as using the mechanism of finite automata supported by the mechanism of the pushdown stack. In other words, the properties of regular languages are used to analyze context-free languages. We can find this kind of methodology in mathematics. (Remember that in order to analyze real numbers, the properties of rational numbers are used.)

Thus it may be said that this paper has shown that just like the above, the properties of Boolean algebra, that is of the language space generated from empty sets, are available for analyzing regular and context-free languages, and that the algebra has sufficient capability in language processing for the linear algebra-like computation of the form of matrices.

For the electrical circuit theory, there are various kinds of underlying mathematical theory such as the theory of differential equations, the theory of complex functions, graph theory, and linear algebra. The author considers that the algebra shown in this paper is available as one more underlying candidate theory in language machine theory. We call the algebra, that is, the linear algebra-like theory on the semi-ring with idempotency in addition, "semi-linear algebra."

References

1. DEREMER, F. and PENNELLO, T. Efficient computation of LALR(1) Look-Ahead sets. *ACM. Trans. Prog. Lang. and Syst.* **4** (1982), 615-649.
2. TREMBLAY, J. and SORENSION, P. G. The Theory and Practice of Compiler Writing, *McGrow-Hill*, New York (1985).
3. AHO, A. V., SETHI, R. and ULLMAN, J. D. Compilers, Principles, and Techniques, and Tools, Addison-Wesley (1986).
4. CARRÉ, B. Graphs and Networks, Clarendon Press, Oxford (1979).
5. GONDRAN, M. and MINOUX, M. Graphs and Algorithms, John Wiley and Sons, New York (1984), 84-128.
6. UTAGAWA, K., INAGAKI, Y. and TANGE, H. The state-characteristic equations of finite automata and their regular expressions, *Proc. IECE, Japan*, **48**, 9 (1965), 1524-1533 (in Japanese).
7. NOZAKI, A. Algebraic theory for transition matrix of finite automata, *Proc. IECE, Japan*, **50**, 2 (1967), 204-206 (in Japanese).
8. ANZAI, H. Algorithms equationally characterizing a group of regular expressions, *Trans. IECE, Japan*, **57-D**, 12 (1974), 653-660; available in English in Systems. Computers. Controls (Scripta Pub. Co.), **5**, 6 (1974), 47-55.
9. ANZAI, H. A theory of recursive descent syntax-directed translator generator, *Proc. of the Int'l Comp. Symp.* '80. 1171-1182.
10. ANZAI, H. and YAMANOUE, T. Fundamental concepts of language processor generator MYLANG, *Trans. IECE, Japan*, **J69-D**, 2 (1986), 117-127; available in English in Systems and Computers in Japan, **17**, 12 (1986), 23-35.
11. TIXIER, V. Recursive Functions of Regular Expressions in Language Analysis, Tech. Rpt. CS58, Comp. Sci. Dept., Stanford U. (1968).
12. WARSHALL, S. A theorem on Boolean matrices, *J. ACM* **9** (1962), 11-12.

(Received June 6, 1988; revised July 3, 1989)

Appendix

Proof of Lemma 2.2 The left statement is valid

iff there exists a sequence of states $x_i=x_{i_0}, x_{i_1}, \cdots, x_{i_r}=x_j$ such that $\tau(x_{i_{k-1}}, s_k)=x_{i_k}$, for $k=1, 2, \cdots, r$

iff ${}^3i=i_0, {}^3i_1, {}^3i_2, \cdots, {}^3i_r=j$ $(1\leqq i_0, i_1, i_2, \cdots, i_r\leqq n)$:

$$[\partial_{s_1}A]_{ii_1}[\partial_{s_2}A]_{i_1i_2}\cdots[\partial_{s_r}A]_{i_{r-1}j}=\lambda$$

iff $\sum_{i_1=1}^{n}\sum_{i_2=1}^{n}\cdots\sum_{i_{r-1}=1}^{n}[\partial_{s_1}A]_{ii_1}[\partial_{s_2}A]_{i_1i_2}\cdots[\partial_{s_r}A]_{i_{r-1}j}=\lambda$

iff the right statement is valid. □

Proof of Lemma 2.3 The left statement is valid

iff $\tau(\cdots\tau(\tau(x_1, s_1), s_2), \cdots, s_r)\in F$

iff ${}^3j(1\leqq j\leqq n)$: $[\partial_{s_1}A\partial_{s_2}A\cdots\partial_{s_r}A]_{1j}=\lambda$ and $[\boldsymbol{c}]_j=\lambda$

iff $\sum_{j=1}^{n}[\partial_{s_1}A\partial_{s_2}A\cdots\partial_{s_r}A]_{1j}[\boldsymbol{c}]_j=\lambda$

iff the right statement is valid. □

Proof of Theorem 2.1 $T(\mathscr{A})=$

$$\sum_{r=0}^{n}\sum_{s_1,s_2,\cdots s_r\in\Sigma}\boldsymbol{e}_1\partial_{s_1}A\partial_{s_2}A\cdots\partial_{s_r}A\boldsymbol{c}\cdot s_1s_2\cdots s_r$$

$$=\boldsymbol{e}_1\sum_{r=0}^{\infty}(\sum_{s_1\in\Sigma}s_1\partial_{s_1}A)(\sum_{s_2\in\Sigma}s_2\partial_{s_2}A)\cdots(\sum_{s_r\in\Sigma}s_r\partial_{s_r}A)\boldsymbol{c}$$

$$=\boldsymbol{e}_1\left(\sum_{r=0}^{\infty}A^r\right)\boldsymbol{c}=\boldsymbol{e}_1A^*\boldsymbol{c}\ \square$$

Proof of Lemma 2.4 From Lemma 2.2, it holds that

${}^3r\geqq 0$, ${}^3s_1, {}^3s_2, \cdots, {}^3s_r\in\Sigma'$: $[\partial_{s_1}A\partial_{s_2}A\cdots\partial_{s_r}A]_{ij}=\lambda$

iff $\sum_{r=0}^{\infty}\sum_{s_1,s_2,\cdots s_n\in\Sigma'}[\partial_{s_1}A\partial_{s_2}A\cdots\partial_{s_r}A]_{ij}=\lambda$

iff $\sum_{r=0}^{\infty}[(\sum_{s_1\in\Sigma'}\partial_{s_1}A)(\sum_{s_2\in\Sigma'}\partial_{s_2}A)\cdots\sum_{s_r\in\Sigma'}\partial_{s_r}A]_{ij}=\lambda$

iff $\sum_{r=0}^{\infty}[C']_{ij}=\left[\sum_{r=0}^{\infty}C''\right]_{ij}=[C'^*]_{ij}=\lambda\ \square$

Proof of Lemma 2.5 ${}^3w'\in\Sigma'^*$, ${}^3w''\in\Sigma''^*$, ${}^3w'''\in\Sigma'''^*$: $w'sw''s'w'''\in T(\mathscr{A})$ means that there exist states x_i, x_j, x_k and x_l such that $[C'^*]_{1i}=\lambda$, $[\partial_sA]_{ij}=\lambda$, $[C''^*]_{jk}=\lambda$, $[\partial_{s'}A]_{kl}=\lambda$ and $[C'''^*\boldsymbol{c}]_l=\lambda$ are valid. Accordingly,

${}^3i, {}^3j, {}^3k, {}^3l$ $(1\leqq i, j, k, l\leqq n)$:

$$[C'^*]_{1i}[\partial_sA]_{ij}[C''^*]_{jk}[\partial_{s'}A]_{kl}[C'''^*\boldsymbol{c}]_l=\lambda$$

iff

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n}[C'^*]_{1i}[\partial_sA]_{ij}[C''^*]_{jk}[\partial_{s'}A]_{kl}[C'''^*\boldsymbol{c}]_l=\lambda$$

iff $\boldsymbol{e}_1C'^*\partial_sAC''^*\partial_{s'}AC'''^*\boldsymbol{c}=\lambda$. □

Proof of Lemma 2.6 In Lemma 2.5, let Σ', Σ'' and Σ''' be Σ, Σ' and Σ, respectively. Then the following relation holds.

$$\boldsymbol{e}_1C^*\partial_sAC'^*\partial_{s'}A\lambda\supseteq\boldsymbol{e}_1C^*\partial_sAC''^*\partial_{s'}AC^*\boldsymbol{c}=\lambda\ \square$$

Proof of Lemma 2.7 Put $w'=s_1s_2\cdots s_r$, $r\geqq 0$, then it holds that ${}^3i, {}^3j$ $(1\leqq i, j\leqq n)$, ${}^3r\geqq 0$, ${}^3s_1, {}^3s_2, \cdots, {}^3s_r\in\Sigma'$:

$$[\partial_sA\partial_{s_1}A\partial_{s_2}A\cdots\partial_{s_r}A]_{ij}=\lambda \text{ and } [\boldsymbol{c}]_j=\lambda$$

iff ${}^3i(1\leqq i\leqq n)$: $\left[\partial_sA\left(\sum_{r=0}C''\right)\boldsymbol{c}\right]_i=\lambda$

iff $'\lambda\partial_sAC'^*\boldsymbol{c}=\lambda\ \square$

Algorithm for Computing Positive Closure of λ-Matrix

Input: $n\times n$ λ-matrix C.

Output. C^+.

Procedure. (1). $M^{(0)}\leftarrow C$.

(2). For $k=1, 2, \cdots, n$, do the following computation:

$$M^{(k)}\longleftarrow M^{(k-1)}+\begin{bmatrix}m_{1k}^{(k-1)}\\ \cdot\\ \cdot\\ \cdot\\ m_{nk}^{(k-1)}\end{bmatrix}[m_{k1}^{(k-1)}\cdots m_{kn}^{(k-1)}]$$

(3). $C^+=M^{(n)}$. □