<table>
<thead>
<tr>
<th>項目</th>
<th>内容</th>
</tr>
</thead>
<tbody>
<tr>
<td>タイトル</td>
<td>インタプリターのためのLALR(1)ルックアヘッドセットの計算方法</td>
</tr>
<tr>
<td>著者</td>
<td>Anzai, Hiroyuki</td>
</tr>
<tr>
<td>引用</td>
<td>数理解析研究所講究録 (1987), 618: 150-169</td>
</tr>
<tr>
<td>発行日</td>
<td>1987-04</td>
</tr>
<tr>
<td>URL</td>
<td><a href="http://hdl.handle.net/2433/99857">http://hdl.handle.net/2433/99857</a></td>
</tr>
<tr>
<td>タイプ</td>
<td>Departmental Bulletin Paper</td>
</tr>
<tr>
<td>テキストバージョン</td>
<td>publisher</td>
</tr>
</tbody>
</table>

京都大学学術情報リポジトリ
Kyoto University Research Information Repository
Equational Computation of LALR(1) Look-Ahead Sets

安在 弘幸

Hiroyuki Anzai

Kyushu Institute of Technology
Tobata-ku, Kitakyushu-shi, 804 Japan

Summary

An efficient method computing LALR(1) Look-Ahead sets from a given grammar was given by DeRemer and Pennello in 1982. Instead of their method based on the traditional graph theoretical approach, this paper presents a formalized and more efficient method based on a new methodology, that is, linear algebra-like approach called semi-linear algebra.

For a given BNF or extended BNF, Tixier's Standard Right Linear equation which is transformed from it and made empty-free is used in this method. For the equation, First sets are equationally defined and solved. Then, a formalized method obtaining an LR(0) automaton from the equation is given, and Follow sets concerned with nonterminal state transitions of the automaton are equationally defined and solved. Finally LALR(1) Look-Ahead sets are obtained as unions of some Follow sets.

1. INTRODUCTION

In order to construct 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 from the automaton. Many
researchers gave methods doing so. An efficient method was shown by DeRemer and Pennello\textsuperscript{1}) in 1982.

The author has been studied linear algebra-like approach to language and automata theory called semi-linear algebra\textsuperscript{4}). This approach applying to the above problem is shown in this paper. In other words, this paper presents a formalized efficient method based on the linear algebra-like approach instead of the graph theoretical one used traditionally until now. This method is efficient because of depending on mainly operation of Boolean matrices instead of set operation used in usual methods.

For a given BNF or extended BNF, Tixier gave a parametric representation equivalent to it, that is, a system of simultaneous right linear equations, and called it Standard Right Linear equation (system). In the equations, each coefficient is a set of terminal and nonterminal symbols and each constant term is either an empty set or an empty string set.

Chapter 2 prepares some definitions necessary in this paper and introduces Tixier's SRL equation. For the equation, if some of the coefficients contain empty strings, Chapter 3 offers a method modifying the equation so as to preserve the effect of the empty strings even if they are removed or neglected. So in the following chapters, it is assumed that none of the coefficients of the equation contains the empty string.

In Chapter 4, for each nonterminal $X$, the first symbol set $\text{First}(X)$, a set of symbols appeared at the first position in sentential forms derived from $X$, is equationally defined and solved.
Chapter 5 presents a method constructing an LR(0) automaton from the SRL equation. In Chapter 6, for each nonterminal state transition \((I_k, X)\) of the LR(0) automaton given in the previous chapter, the follow set \(\text{Follow}(I_k, X)\) is equationally defined and solved. Finally in Chapter 7, each LALR(1) Look-Ahead set is obtained as a union of some Follow sets.

2. STANDARD RIGHT LINEAR EQUATION

Some preparatory definitions and Tixier's SRL equation\(^3\) as a description of context-free grammars are introduced.

A set of terminals and a set of nonterminals are denoted by \(T\) and \(N\), respectively. A union of \(T\) and \(N\) is written by \(V\). The empty set is represented by \(\phi\). The empty string and the empty string set are both denoted by \(\lambda\). A union of \(V\) and \(\lambda\) is represented by \(V^\lambda\). For notational conventions, we use \(t\) as an element of \(T\); \(X, Y\) and \(Z\) as elements of \(N\); and \(s\) as an element of \(V\). And furthermore, for the summing operator \(\sum\), \(\sum_{t \in T} \) is abbreviated as \(\sum_t\); \(\sum_{X \in N} \) as \(\sum_X\); and \(\sum_{s \in V} \) as \(\sum_s\).

For a given set \(S\), a matrix \(A = (a_{ij})\), \(a_{ij} \subseteq S\) and a vector \(v = (v_i)\), \(v_i \subseteq S\), are called an \(S\)-matrix and an \(S\)-vector, respectively.

For a set \(S \subseteq V^\lambda\) and a symbol \(\sigma \subseteq V^\lambda\), a function \(\partial_\sigma\) is defined as

\[
\partial_\sigma S = \lambda \quad \text{if } \sigma \subseteq S, \\
= \phi \quad \text{otherwise.}
\]

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

\[
\partial_\sigma A = (\partial_\sigma a_{ij}).
\]

For a nonterminal symbol \(X \subseteq N\), a function \(\rho\) is defined as
\[ \rho X = \lambda \quad \text{if } X \text{ derives } \lambda \text{, i.e. } X \Rightarrow^* \lambda , \]
\[ = \phi \quad \text{otherwise.} \]

And furthermore, the definition is extended for a set \( S \) and a matrix \( A = (a_{ij}) \), as follows:
\[ \rho S = \lambda \quad \text{if } \lambda \in S \text{ or } \exists X \in S: \rho X = \lambda , \]
\[ = \phi \quad \text{otherwise.} \]
\[ \rho A = (\rho a_{ij}). \]

Example 1. For a set \( S = \{X, t\} \), where \( \rho X = \lambda \) and \( Y \notin X \),
it holds that \[ \rho S = \lambda, \quad \exists_\lambda S = \phi, \quad \exists X S = \lambda, \quad \exists t S = \lambda \] and \[ \exists Y S = \phi. \]

For a given BNF or extended BNF, Tixier gave its parametric representation called Standard Right Linear or SRL equation. This is an extension of representing a given regular expression as a right linear equation of which the 1st component of the minimal solution is equivalent to the regular expression. Namely, for each nonterminal \( X \) in the BNF or extended BNF, the \( X \)-defining equation:
\[ X ::= f(Y, \cdots, X, \cdots, Z) \quad (2.1) \]
is unfolded by means of introducing appropriate intermediate parameters \( x_{X1}, x_{X2}, \ldots, x_{Xn_X} \) as follows:
\[ X = x_{X1} \]
\[ x_{X_i} = \sum_{j=1}^{n_X} a_{Xij} x_{Xj} + c_{X_i} \]
where \( a_{Xij} \in \mathcal{L}^\lambda, \ c_{X_i} \in \lambda \)
\[ i = 1, 2, \ldots, n_X \]
or
\[ x_X = A_x x_X + c_X \quad (2.2) \]
where \( x_X = (x_{X1}), \ c_X = (c_{X_i}), \ a_X = (a_{Xij}). \)

In the BNF or extended BNF Eq.(2.1), the right part
f(Y,\cdots,X,\cdots,Z) is able to be regarded as a regular expression defined on a space $V^*$, which is a set of strings generated from an alphabet $V^\lambda$. On the other hand, the SRL equation Eq.(2.2) is able to be regarded as defined on an $n_X$-dimensional $V^*$-vector space. In the space, it has the minimal solution $A_X^*c_X$, of which the 1st component is equivalent to the right part $f(Y,\cdots,X,\cdots,Z)$ of Eq.(2.1). We call the solution on $V^*$ the semi-solution.

In the following discussions, we use the SRL equation Eq.(2.2) instead of the BNF or extended BNF Eq.(2.1), and there, the right three $n_X$-dimensional $\lambda$-vectors associated with Eq.(2.2) will be often used.

Example 2.

Fig.1 is an example grammar written by BNF. Fig.2 is a state transition diagram representing the multiple finite automata system derived from Fig.1.

\[
S \rightarrow G \# \\
G \rightarrow E = E \mid f \\
E \rightarrow T \mid E + T \\
T \rightarrow f \mid T \ast f
\]

Fig.1 An example grammar quoted from the book written by Tremblay and Sorensen²).
The coefficient matrices and the constant term vectors of the SRL equations describing the automata in Fig.2 are shown below.

\[ A_S = \begin{bmatrix} \phi & G & \phi \\ \phi & \phi & \# \\ \phi & \phi & \phi \end{bmatrix}, \quad c_S = \begin{bmatrix} \phi \\ \phi \\ \lambda \end{bmatrix}, \quad A_E = \begin{bmatrix} \phi & T & E & \phi & \phi \\ \phi & \phi & \phi & \phi & \phi \\ \phi & \phi & \phi & \phi & T \\ \phi & \phi & \phi & \phi & \phi \end{bmatrix}, \quad c_E = \begin{bmatrix} \phi \\ \phi \\ \phi \end{bmatrix} \]

\[ A_G = \begin{bmatrix} \phi & E & \phi & f \\ \phi & \phi & \phi & \phi \\ \phi & \phi & \phi & \phi \\ \phi & \phi & \phi & \phi \end{bmatrix}, \quad c_G = \begin{bmatrix} \phi \end{bmatrix}, \quad A_T = \begin{bmatrix} \phi & f & T & \phi & \phi \\ \phi & \phi & \phi & \phi & \phi \\ \phi & \phi & \phi & \phi & \phi \\ \phi & \phi & \phi & \phi & \phi \end{bmatrix}, \quad c_T = \begin{bmatrix} \phi \end{bmatrix} \]

Fig.3

3. EMPTY-FREE EQUATION

An SRL equation \( x_X = A_X x_X + c_X \) (2.2) is said to be empty-free if its minimal semi-solution is preserved even if each empty string contained in \( A_X \) are removed or neglected. This chapter presents a method making a given SRL equation empty-free without changing its minimal semi-solution. For this, we must obtain the value of \( \rho X \), for each \( X \in N \).

For a given BNF or extended BNF, it is not so difficult to evaluate \( \rho X, X \in N \), by inspection. A method doing so by calculation is given as follows:

<table>
<thead>
<tr>
<th>\cdot</th>
<th>\lambda u \phi</th>
<th>\lambda u \phi</th>
</tr>
</thead>
<tbody>
<tr>
<td>\lambda</td>
<td>\lambda u \phi</td>
<td>\lambda</td>
</tr>
<tr>
<td>u</td>
<td>u</td>
<td>u</td>
</tr>
<tr>
<td>\phi</td>
<td>\phi</td>
<td>\phi</td>
</tr>
</tbody>
</table>

As the preparation, interpret the concatenation operator "\cdot" and the selection operator "\mid" (or addition operator "\oplus") as shown in Fig.4. Then, for each \( X \in N \), initialize \( \rho X \) as \( \rho X = u \), where \( u \) means "undefined". Note
that the function $\rho$ is homomorphism. Namely, it holds that
\[ \rho(s + s') = \rho s + \rho s', \quad \rho(ss') = \rho s \rho s' \quad \text{and} \quad \rho(s^*) = (\rho s)^*. \]

Then for each $X$-defining equation:
\[ X ::= f(Y, \cdots, X, \cdots, Z), \]
such that $\rho X = u$, evaluate
\[ \rho X = f(\rho Y, \cdots, \rho X, \cdots, \rho Z) \]
recursively until values of every $\rho X$, $X \in N$, become stable.

**Example 3.** Evaluate $\rho X$ and $\rho Y$ in Fig.5.
\[
\begin{align*}
\rho X &= \rho(aX + Y) \\
&= \rho a \rho X + \rho Y \\
&= \rho Y \\
&\quad (\because \rho a = \phi)
\end{align*}
\]
Fig.5 An example of BNF.
\[
\begin{align*}
Y &= \rho(aY + \lambda) = \rho a \rho Y + \rho \lambda = \lambda \\
X &= \lambda
\end{align*}
\]

On the other hand, in order to evaluate $\rho X$ by using the $X$-defining SRL equation Eq.(2.2), the following equation can be used in the above manner.
\[ \rho X = i_X(\delta_\lambda A_X + \sum Y \rho Y \delta_\lambda A_X)^* c_X \]  
(3.2)

If the value of each $\rho X$, $X \in N$, is thus determined as whether $\lambda$ or $\phi$, we are able to make each the SRL equation empty-free by means of the following transformation.
\[ x_X = (\rho A_X)^* (A_X - \delta_\lambda A_X) x_X + c_X \]  
(3.3)

Eq.(3.3) preserves the semi-solution of Eq.(2.2). Furthermore, Eq.(3.3) is equivalent to Eq.(2.2) concerned with the semi-solution even if each nonterminal symbol in the coefficients of Eq.(3.3) would not contain $\lambda$. In other words, the transformation Eq.(3.3) completely realizes in Eq.(3.3) itself the effect of all $\lambda$'s contained in $A_X$ of Eq.(2.2).
This fact means that, for Eq.(3.3), if there exists a non-terminal containing $\lambda$, we may neglect it or remove it from the non-terminal except for the starting nonterminal $X_0$ without loss of its effect. Namely, we may assign $\phi$ to $c_{X_1}$, the 1st component of $c_X$ in Eq.(3.3) defining $X$ except for $X_0$.

For a given grammar, parsing techniques generally require to add a padding rule defining a new starting nonterminal $X'_0$ instead of the original $X_0$, as follows:

$$X'_0 ::= X_0 \#$$

where $\#$ is a newly introduced terminal which is not contained in the original terminal set $T$ and is used as the end-marker of input sequences.

We are thus able to make a given SRL equation system empty-free. Therefore in the following chapters, we assume without loss of generality that a given SRL equation system is empty-free.

**Example 4.** A system of SRL equations transformed from the BNF in Fig.5 is shown below. Make each constituent equation empty-free.

\[
S = s_1, \quad \begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix} = \begin{bmatrix} \phi & X & \phi \\ \phi & \phi & \# \\ \phi & \phi & \phi \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix} + \begin{bmatrix} \phi \end{bmatrix} \tag{3.4}
\]

\[
X = x_1, \quad \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} \phi & a & Y \\ \phi & \phi & X \\ \phi & \phi & \phi \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} \phi \end{bmatrix} \tag{3.5}
\]

\[
Y = y_1, \quad \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} \phi & a & \lambda \\ \phi & \phi & Y \\ \phi & \phi & \phi \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} + \begin{bmatrix} \phi \end{bmatrix} \tag{3.6}
\]

From Eq.(3.3), Eq.(3.4) becomes
\[ S = s_1, \quad \begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix} = \phi \lambda \phi^* \begin{bmatrix} \phi & \lambda & \phi \\ \phi & \phi & \# \\ \phi & \phi & \phi \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix} + \phi \]

\[ = \begin{bmatrix} \lambda & \lambda & \phi \\ \phi & \lambda & \phi \\ \phi & \phi & \phi \end{bmatrix} \begin{bmatrix} \phi & \lambda & \phi \\ \phi & \phi & \# \\ \phi & \phi & \phi \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix} + \phi \]

\[ = \begin{bmatrix} \phi & \lambda & \phi \\ \phi & \phi & \# \\ \phi & \phi & \phi \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix} + \phi \]

(3.7)

Similarly, Eq. (3.5) and Eq. (3.6) becomes

\[ X = x_1, \quad \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \phi \alpha \gamma \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \phi \]

(3.8)

\[ Y = y_1, \quad \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \phi \alpha \phi \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} + \phi \]

(3.9)

In each Eq. (3.8) and (3.9), the 1st constituent of the constant term was \( \lambda \) after computation. However they were changed to \( \phi \) without loss of the effect.

4. FIRST SYMBOL SETS

For a nonterminal \( X \) in a production grammar, a set of symbols (terminals and nonterminals) appeared at the 1st position of sentential forms directly derived from \( X \) is defined as

\[ \text{DFirst}(X) = \{ s \in V \mid X \Rightarrow s w, w \in V^* \} \]

and is called a Directly First (symbol) set.

Similarly, a set of symbols appeared at the first position of sentential forms derived from \( X \) is defined as

\[ \text{First}(X) = \{ s \in V \mid X \Rightarrow s w, w \in V^* \} \]

and is called a First (symbol) set.
They are, in our equation Eq.(2.2), defined as

\[
D\text{First}(X) = i_X a_X^X = \sum_{j=1}^{n_X} a_X^1 j
\]  
(4.1)

\[
\text{First}(X) = \sum_{Y \in X} \delta_Y a_X^X \text{First}(Y) + D\text{First}(X)
\]  
(4.2)

Now, in order to solve them, put

\[
u = (u_X), \ u_X = \text{First}(X),
\]
\[
d = (d_X), \ d_X = D\text{First}(X),
\]
\[
\Gamma = (\gamma_{XY}), \ \gamma_{XY} = i_X \delta_Y a_X^X \text{First}(Y) = \delta_Y (i_X a_X^X) = \delta_Y d_X
\]
then, Eq.(4.2) becomes

\[
u = \Gamma \ u + d = _s \Gamma^* d
\]  
(4.3)

Where, _s shows that its right-hand side is the minimal solution of the equation in its left-hand side.

Example 5 Obtain all First (symbol) sets in Example 2.

From Eq.(4.1) and Fig.3, we have

\[
d_S = \{ G \}, \ \ d_G = \{ E, f \}, \ d_E = \{ E, T \}, \ d_T = \{ T, f \}.
\]

From Eq.(4.3), the desired First sets are obtained as follows:

\[
\Gamma = \begin{bmatrix}
\delta_S d_S & \delta_G d_S & \delta_E d_S & \delta_T d_S \\
\delta_S d_G & \delta_G d_G & \delta_E d_G & \delta_T d_G \\
\delta_S d_E & \delta_G d_E & \delta_E d_E & \delta_T d_E \\
\delta_S d_T & \delta_G d_T & \delta_E d_T & \delta_T d_T \\
\end{bmatrix}
\]
\[
d = \begin{bmatrix}
\phi & \lambda & \phi & \phi \\
\phi & \phi & \lambda & \phi \\
\phi & \phi & \lambda & \lambda \\
\phi & \phi & \phi & \lambda \\
\end{bmatrix}
\]
\[
\Gamma = \begin{bmatrix}
\delta_S d_S & \delta_G d_S & \delta_E d_S & \delta_T d_S \\
\delta_S d_G & \delta_G d_G & \delta_E d_G & \delta_T d_G \\
\delta_S d_E & \delta_G d_E & \delta_E d_E & \delta_T d_E \\
\delta_S d_T & \delta_G d_T & \delta_E d_T & \delta_T d_T \\
\end{bmatrix}
\]
\[
u = \text{First}(S), = \Gamma^* d = \begin{bmatrix}
\lambda & \lambda & \lambda & \lambda \\
\phi & \lambda & \lambda & (G) \\
\phi & \phi & \lambda & (E,f) \\
\phi & \phi & \phi & (T) \\
\end{bmatrix}
\]
\[
\text{First}(G) = (G, E, T, f) \quad \text{First}(E) = (E, T, f) \quad \text{First}(T) = (T, f)
\]

5. LR(0) AUTOMATON

We have studied the traditional method to derive an LR(0) automaton from a given grammar and found that it is an extended variation of the subset construction method used in the case of
transformation from a nondeterministic finite automaton into a
deterministic one. This chapter shows a formalized method con-
structing an LR(0) automaton from a given SRL equation (2.2).

For any nonterminals Z and X, a function \( f_{ZX} \) is defined as

\[
f_{ZX} = \delta_X(\{ Z \} \cup First(Z)).
\]  

Each state of the LR(0) automaton, the so-called LR(0) term, is
specified by a bit vector of which the length (i.e. dimension)
is \( \sum X n_X \), as follows:

\[
I_k = ( I_{kX}, \ldots, I_{kX}, \ldots, I_{kZ} ), \; k = 0, 1, \ldots, K.
\]

where each \( I_{kX}, X \in N \), is an \( n_X \)-dimensional bit vector associated
with the \( X \)-defining SRL equation: \( x_X = \lambda_X x_X + c_X \).

The initial state of the automaton is defined as

\[
I_0 = ( I_{0X_0}, \ldots, I_{0X}, \ldots, I_{0Z} ),
\]

where \( I_{0X} = f_{X_0X} x_{X_0} \), \( X \in N \),

\( X_0 \) is the starting nonterminal.

The state transition function of the automaton is defined as

\[
Goto(I_k, s) = \text{Closure}(\text{Trans}(I_k, s))
\]  

where

\[
\text{Trans}(I_k, s) = \text{Trans}((I_{kX}, \ldots, I_{kX}, \ldots, I_{kZ}), s)
\]

\[
= (\text{Trans}(I_{kX_0}, s), \ldots, \text{Trans}(I_{kX}, s), \ldots, \text{Trans}(I_{kZ}, s))
\]  

where for any \( X \in N \), \( \text{Trans}(I_{kX}, s) = J_{kX} = I_{kX} \delta_s A_X \)  

\[
\text{Closure}(J_k) = \text{Closure}( J_{kX_0}, \ldots, J_{kX}, \ldots, J_{kZ} )
\]

\[
= (\text{Closure}(J_{kX_0}), \ldots, \text{Closure}(J_{kX}), \ldots, \text{Closure}(J_{kZ}))
\]  

where for any \( X \in N \),

\[
\text{Closure}(J_{kX}) = J_{kX} + \sum Y \sum Z J_{kY} \delta_Z A_Y \lambda_Y f_{ZX} x_X
\]  

If a state \( I_k \) contains at least one \( I_{kX} \) such that \( I_{kX} c_X = \lambda \),
then it is a reduce state of the automaton.
Example 6. From Example 2, obtain the LR(0) automaton.

\[ I_k = (I_{kS}, I_{kG}, I_{kE}, I_{kT}) \]

\[ I_0 = (\lambda \lambda \lambda \lambda \lambda \lambda \lambda \lambda \lambda \lambda \lambda \lambda \lambda \lambda \lambda \lambda \lambda \lambda \lambda) = I_0. \]

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

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

Trans(I_0,T) = (\phi \lambda \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi) = \text{Goto}(I_0,T) = I_3.

Trans(I_0,f) = (\phi \lambda \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi) = \text{Goto}(I_0,f) = I_4.

Trans(I_1,#) = (\phi \lambda \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi) = \text{Goto}(I_1,#) = I_5.

Trans(I_2,\#) = (\phi \lambda \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi) = \text{Goto}(I_2,\#) = I_6.

= (\phi \lambda \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi) = \text{Goto}(I_3,\#) = I_7.

Trans(I_3,\#) = (\phi \lambda \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi) = \text{Goto}(I_4,\#) = I_8.

Trans(I_8,E) = (\phi \lambda \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi) = \text{Goto}(I_5,E) = I_9.

Trans(I_8,f) = (\phi \lambda \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi \phi) = \text{Goto}(I_6,f) = I_{10}.

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

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

Trans(I_8,T) = \text{Goto}(I_8,T) = I_{13}.

Trans(I_7,f) = \text{Goto}(I_7,f) = I_{14}.

Trans(I_9,#) = \text{Goto}(I_9,#) = I_{15}.

Trans(I_{11},\#) = \text{Goto}(I_{11},\#) = I_{16}.

The embedding matrix

\[
\begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\
S & G & E & T & E & T & T & T & T & T & T & T \\
I_0 & I_0 & I_0 & I_0 & I_0 & I_0 & I_0 & I_0 & I_0 & I_0 & I_0 & I_0 \\
1 & 1 & 1 & 1 & . & . & . & . & . & . & . & . \\
\end{array}
\]

Fig. 6
The above method constructs an LR(0) automaton from a given SRL equation (2.2) so as to embed once or more times each one of finite automata described by the SRL equation into the LR(0) automaton. In the above example, this is shown by a matrix in the right-hand side. In the matrix, each column corresponds to one of occurrences of the embedded automata and, for each state $I_k$ of the LR(0) automaton, each element of the $I_k$-th row contains $\lambda$ if there is, in the embedded automata, a state which is embedded in $I_k$, otherwise $\phi$. (In Fig.6, the state number is written instead of $\lambda$.) In this case, automata $S$ and $G$ were each embedded from state $I_0$, automaton $E$ was embedded from states $I_0$ and $I_6$, and automaton $T$ was embedded from states $I_0$, $I_6$ and $I_7$.

This matrix is called an embedding matrix and each element is denoted by $E[I_k,(I_k,Y)]$, where $I_k$ is the row number associated with the state $I_k$ of the derived LR(0) automaton and $(I_k,Y)$ indicates an occurrence of embedding the automaton $Y$ from state $I_k$ in the LR(0) automaton. For each nonterminal transition $(I_k,Y)$, there exists one and only one occurrence of embedding the automaton $Y$ from state $I_k$, and it suffices that $I_k Y^t i_Y = \lambda$.

6. FOLLOW SETS

For the LR(0) automaton, Follow set concerned with each nonterminal (state) transition $(I_k,X)$ is defined as

$$\text{Follow}(I_k,X) = \{ t \in T \mid X_0 \xrightarrow{\hat{t}} \theta X w \text{ and } \theta \text{ access state } I_k \}$$

where $X_0$ is the starting nonterminal and $w \in T^*$.

DeRemer and Penello derived from the definition two types of inclusive relations related to Follow sets and gave a method to obtain the sets concretely on the basis of the inclusive
relations in the manner of recursive computation, which needs repetitive computation of set union and set comparison taking much time.

Our method shown below takes not so much time because of depending on mainly closure operation of an \( n \times n \) Boolean matrix instead of the recursive computation on sets, where \( n \) is the number of nonterminal transitions in the LR(0) automaton.

Generally, a production grammar has two types of rules such as \( Y \rightarrow \alpha X \beta, \alpha, \beta \in V^* \) and \( Y \rightarrow \alpha X, \alpha \in V^* \), which introduce two types of situations in the derivation sequences, as follows:

For \( Y \rightarrow \alpha X \beta \), there is \( X_0 \xrightarrow{R} \delta Y w \xrightarrow{R} \delta \alpha X \beta w \) and for \( Y \rightarrow \alpha X \), there is \( X_0 \xrightarrow{R} \delta Y w \xrightarrow{R} \delta \alpha X w \).

Similarly, in the related LR(0) automaton, two types are derived as shown in Fig.7 and Fig.8.

![Fig.7](image)

\( Y \rightarrow \alpha X \beta \)

![Fig.8](image)

\( Y \rightarrow \alpha X \)
According to the 1st type, we have the following.

**Proposition 6.1** The following 2 conditions are all equivalent.
- There exists a rule \( Y \rightarrow \alpha X \beta, \alpha, \beta \in V^* \).
- There exist an occurrence of embedding an automaton \( Y \) and a state \( I_k \) of the LR(0) automaton such that
  \[
  I_{kY} \beta_X A_Y \delta_s A_Y e_Y = \lambda
  \]  
  \( (6.1) \)

Fig. 7 shows this situation and its relation to the definition of Follow sets. Namely, if Eq. (6.1) holds, we can expect to have the following relation.

\[
\text{Follow}(I_k, X) \supseteq \text{First}_T(s)
\]

where \( \text{First}_T(s) = s \), for \( s \in T \),

\[
= T \cap \text{First}(s), \text{ for } s \in N.
\]

The above conditional relation is simply shown as follows:

\[
\text{Follow}(I_k, X) \supseteq I_{kY} \beta_X A_Y \delta_s A_Y e_Y \text{First}_T(s)
\]

(6.3)

This relation for any \( Y \in N \) and for any \( s \in V \) is given as

\[
\text{Follow}(I_k, X) \supseteq d[(I_k, X)]
\]

(6.4)

where \( d[(I_k, X)] = \sum_y I_{kY} \beta_X A_y b_y \)

(6.5)

\[
b_y = \sum_s \delta_s A_Y e_Y \text{First}_T(s)
\]

(6.6)

Note that \( I_{kY} \beta_X A_Y \) is a \( \lambda \)-vector \( \text{Trans}(I_{kY}, X) \) given in the previous chapter.

Associated with the 2nd type, the following is given.

**Proposition 6.2** The following 3 conditions are all equivalent.
- There is a rule \( Y \rightarrow \alpha X, \alpha \in V^* \).
- There are two nonterminal transitions \( (I_k, X) \) and \( (I_\ell, Y) \) and there is an occurrence of embedding an automaton \( Y \) of which the initial state is embedded in \( I_\ell \) and of which a final state is embedded in state \( \text{Goto}(I_k, X) \).
There are two nonterminal transition \((I_k, X)\) and \((I_\ell, Y)\) such that \(E[I_k, (I_\ell, Y)] = \lambda\) and there is a state \(\text{Goto}(I_k, X)\) reducible concerned with \(Y\), that is, it holds that \(I_k Y A X Y C Y = \lambda\).

Fig. 8 shows the above situation and relations to the definition of Follow sets. Namely, if the above condition holds, we can expect to have the following relation

\[
\text{Follow}(I_k, X) \equiv \text{Follow}(I_\ell, Y)
\]  

(6.7)

The above conditional relation is simply shown as

\[
\text{Follow}(I_k, X) \equiv \sum_{(I_\ell, Y) \in \Omega} \gamma[(I_k, X), (I_\ell, Y)] \cdot \text{Follow}(I_\ell, Y)
\]  

(6.8)

where

\(\Omega\) is an ordered set of all nonterminal transitions.

\(\gamma[(I_k, X), (I_\ell, Y)] = \lambda\) if the condition of PROPOSITION 6.2 holds,

\(= \phi\) otherwise.

From Eqs. (6.4) and (6.8), Follow sets are, for a given LR(0) automaton, defined as follows:

\[
\text{Follow}(I_k, X) = \sum_{(I_\ell, Y) \in \Omega} \gamma[(I_k, X), (I_\ell, Y)] \cdot \text{Follow}(I_\ell, X) + d[(I_k, X)]
\]  

(6.9)

In order to compute Eq. (6.9), put

\[
\begin{align*}
\mathbf{u} &= ( u[(I_k, X)] ), \quad u[(I_k, X)] = \text{Follow}(I_k, X) \\
\Gamma &= ( \gamma[(I_k, X), (I_\ell, Y)] ), \\
d &= ( d[(I_k, X)] ).
\end{align*}
\]

Then, the desired solution is obtained as follows:

\[
\mathbf{u} = \Gamma \mathbf{u} + d = S \Gamma^* d
\]  

(6.10)

**Example 7** Obtain Follow sets in the LR(0) automaton in Example 6.

Because there are six allowable nonterminal transitions \((I_0, G), (I_0, E), (I_0, T), (I_0, E), (I_0, T)\) and \((I_7, T)\) in the LR(0) automaton, it is necessary to obtain a six-dimensional vector \(d\) and an \(n \times n\) matrix \(\Gamma\). \(d\) needs \(b\) which is composed of \(b_S, b_G, b_E\) and \(b_T\) obtained from Eq. (6.5) and Example 5.
b = \begin{bmatrix}
\{ f \} \\
\phi \\
\phi \\
\phi \\
\phi \\
\phi \\
\{ f \} \\
\{ f \} \\
\{ f \} \\
\{ f \}
\end{bmatrix}

\Gamma = \begin{bmatrix}
\phi & \phi & \phi & \phi & \phi \\
\phi & \phi & \phi & \phi & \phi \\
\phi & \phi & \phi & \phi & \phi \\
\phi & \phi & \phi & \phi & \phi \\
\phi & \phi & \phi & \phi & \phi \\
\phi & \phi & \phi & \phi & \phi \\
\phi & \phi & \phi & \phi & \phi \\
\phi & \phi & \phi & \phi & \phi \\
\phi & \phi & \phi & \phi & \phi \\
\phi & \phi & \phi & \phi & \phi 
\end{bmatrix}

Thus, we have the desired Follow sets as follows:

u = \begin{bmatrix}
\text{Follow}(I_0, \emptyset) \\
\text{Follow}(I_0, E) \\
\text{Follow}(I_0, T) \\
\text{Follow}(I_6, E) \\
\text{Follow}(I_6, T) \\
\text{Follow}(I_7, T)
\end{bmatrix} = \Gamma^d = \begin{bmatrix}
\lambda & \phi & \phi & \phi & \phi \\
\phi & \lambda & \phi & \phi & \phi \\
\phi & \lambda & \lambda & \phi & \phi \\
\lambda & \phi & \phi & \lambda & \phi \\
\lambda & \phi & \phi & \lambda & \phi \\
\lambda & \phi & \phi & \lambda & \lambda
\end{bmatrix} = \begin{bmatrix}
\{ \# \} \\
\{ \# \} \\
\{ \# \} \\
\{ \# \} \\
\{ \# \} \\
\{ \# \}
\end{bmatrix} = \begin{bmatrix}
\{ \# \} \\
\{ \# \} \\
\{ \# \} \\
\{ \# \} \\
\{ \# \} \\
\{ \# \}
\end{bmatrix}

7. LOOK-AHEAD SETS

LALR(1) parsers use sets called Look-Ahead sets to make parse deterministic. The sets are defined for each reduce state \( I_k \) concerned with a nonterminal \( X \), i.e. a rule \( X \rightarrow \alpha \), as follows:

\[ \text{LA}(I_k, X \rightarrow \alpha) = \{ t \in T \mid X_0 \xrightarrow{R} \delta Xtw \} \]

and \( \delta \alpha \) access \( I_k \)
= \{ \text{Follow}(I, X) \mid I \text{ transits to } I_k \text{ by } \alpha \}

Fig. 9 shows the situation in the LR(0) automaton concerned with the definition. So we can expect to have the following.

**PROPOSITION 7.1** The following 3 conditions are all equivalent.

- \( I_k \) transits to \( I_k \) by \( \alpha \) for the rule \( X \rightarrow \alpha \).
- There exists an occurrence of embedding an automaton \( X \) of which the initial state is embedded in \( I_k \) and of which a final state is embedded in \( I_k \) and the initial state transits to the final state by \( \alpha \) in the automaton \( X \).
- \( E(I_k, (I, X)) = \lambda \) and \( I_kX^C X = \lambda \) \hspace{1cm} (7.1)

Here, we obtain a table called Look-Back table from the embedding matrix in such a manner that if \( I_k \) is not a reduce state, then the \( I_k \)-th row is omitted. Note that only states satisfying Eq.(7.1) are left. There may be a state reducible concerned with two or more nonterminals. Accordingly, we expand the LB table to the row-direction such that if \( I_k \) is reducible concerned with nonterminals \( X, X',..., X'' \), then we make reducible state-nonterminal pairs \( (I_k, X), (I_k, X'),..., (I_k, X'') \) and expand the \( I_k \)-th row to the \( (I_k, X) \)-th, the \( (I_k, X') \)-th, ..., the \( (I_k, X'') \)-th rows so as to satisfy the following condition. The obtained is a matrix called Look-Back matrix of which the \( [(I_k, X), (I_k, Y)] \) element is defined as

\[
\Delta[(I_k, X), (I_k, Y)] = \begin{cases} 
\text{LB}(I_k, (I_k, Y)) & \text{if } X = Y \\
\emptyset & \text{otherwise.}
\end{cases}
\]

Note that the Look-Back matrix is obtained directly from the embedding matrix as follows:

\[
\Delta[(I_k, X), (I_k, Y)] = E(I_k, (I_k, Y)), \text{ if } X = Y
\]
\[ = \phi \quad \text{otherwise.} \quad (7.2) \]

Then for each reducible state-nonterminal pair \((I_k, X \rightarrow \alpha)\), the Look-Ahead set \(LA(I_k, X \rightarrow \alpha)\) is defined equationally as follows:

\[
LA(I_k, X \rightarrow \alpha) = \bigoplus_{(I_k, Y) \in \Omega} \Delta[(I_k, X), (I_k, Y)] \cdot \text{Follow}(I_k, Y)
\]

\[ (7.3) \]

In order to compute them, put

\[ v = (LA(I_k, X \rightarrow \alpha)), \]
\[ \Delta = (\Delta[(I_k, X), (I_k, Y)]), \]
\[ u = (u[(I_k, Y)]), \quad u[(I_k, Y)] = \text{Follow}(I_k, Y) \]

Then, from Eqs. (7.3) and (6.9), the desired Look-Ahead sets are obtained, as follows:

\[ v = \Delta u = \Delta \Gamma^* d \]

\[ (7.4) \]

Note that it is not necessary to obtain Follow sets concretely in order to compute Look-Ahead sets.

**Example 8** Obtain Look-Ahead sets of the LR(0) automaton in Example 6.

\[ u = (LA(I_k, X \rightarrow \alpha)) = \Delta \Gamma^* d \]

\[
= \begin{bmatrix}
LA(I_3, E \rightarrow T) \\
LA(I_4, G \rightarrow f) \\
LA(I_4, T \rightarrow f) \\
LA(I_9, G \rightarrow E = E) \\
LA(I_{10}, T \rightarrow f) \\
LA(I_{11}, E \rightarrow E + T) \\
LA(I_{12}, T \rightarrow T * f)
\end{bmatrix} = \begin{bmatrix}
\phi \lambda \phi \lambda \phi \phi \phi \\
\lambda \phi \phi \phi \phi \phi \phi \phi \phi \\
\phi \phi \lambda \phi \phi \phi \phi \phi \phi \\
\lambda \phi \phi \phi \phi \phi \phi \phi \phi \\
\phi \phi \phi \lambda \lambda \phi \lambda \phi \phi \phi \\
\phi \phi \phi \phi \phi \lambda \lambda \phi \phi \phi \phi \phi \phi
\end{bmatrix} \begin{bmatrix}
\{ # \} \\
\{ #, =, + \} \\
\{ =, +, * \} \\
\{ + \} \\
\{ # \} \\
\{ #, =, +, * \}
\end{bmatrix} = \begin{bmatrix}
\{ #, =, + \}
\end{bmatrix}
\]

19
8. CONCLUSION

For a given BNF or extended BNF, Tixier's SRL equation equivalent to it was used. The SRL equation, which describes multiple finite automata system, was at first made empty-free. First sets were equationally defined on the empty-free SRL equation and solved.

A method that derives an LR(0) automaton from a given SRL equation was given. Follow sets were equationally defined on the LR(0) automaton and solved. LALR(1) parse on the LR(0) automaton uses Look-Ahead sets composed of Follow sets.

Until now, concrete computation of the above sets has been done in the recursive manner on the equations. Instead, this paper gave direct and efficient computation using the solutions.

This work was supported by the Spec. Res. Grant. Min. Educ. of Japan in 1986. This assistance is acknowledged.

REFERENCES


