<table>
<thead>
<tr>
<th>Title</th>
<th>Deterministic Parse for Recursive Descent Syntax-Directed Translators</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author(s)</td>
<td>Anzai, Hiroyuki</td>
</tr>
<tr>
<td>Citation</td>
<td>数理解析研究所講究録 (1991), 754: 176-185</td>
</tr>
<tr>
<td>Issue Date</td>
<td>1991-06</td>
</tr>
<tr>
<td>URL</td>
<td><a href="http://hdl.handle.net/2433/82104">http://hdl.handle.net/2433/82104</a></td>
</tr>
<tr>
<td>Type</td>
<td>Departmental Bulletin Paper</td>
</tr>
<tr>
<td>Textversion</td>
<td>publisher</td>
</tr>
</tbody>
</table>
Deterministic Parse for Recursive Descent Syntax-Directed Translators

Hiroyuki Anzai
Kyushu Institute of Technology

Abstract For a given simple syntax-directed recursive-descent translator, a linear algebralike method of computing extended LL(1) director sets, which is necessary for the translator make its parse deterministic, is presented. The fact that the language space generated from the empty alphabet is isomorphic to Boolean algebra is pointed out and used in this method. A given translation scheme is transformed into a right linear equation system, from which simultaneous equations defining the First sets as the unknowns are given and solved. Similarly, the Follow sets are equationally defined and solved. Finally, the desired Director sets are obtained from the above sets. The form of the solutions is a formula of which almost parts are computation of Boolean matrices and vectors.

1. INTRODUCTION

As the number spaces are treated as the algebra "field", the language spaces can be treated as an algebra "semiring with idempotency in addition". We call this algebra a "language semiring". All axioms which both algebras commonly have make as a whole an algebra "semiring". For the algebra, the field has only three extra axioms: commutativity in multiplication, existence of the inverse element in addition and in multiplication, respectively, and the language semiring has one extra axiom: idempotency in addition.

The language space generated from the empty alphabet is a set whose elements are only the empty set $\varnothing$ and the empty string set $\lambda$, and is necessarily contained in the language space generated from arbitrary alphabet. This language space is, as a language semiring, isomorphic to Boolean algebra.
The above nontrivially minimum language space is useful for applying the so-called "divide and conquer" strategy to the problems concerned with language processing. Namely, the problems are solved by means of partially reducing to those of (the language space equivalent to) Boolean algebra. We have already shown the applications to the problem of obtaining LR(0) automaton and LALR(1) look-ahead sets[1] and that of obtaining LL(1) director sets[2] from a given grammar, respectively.

These problems are generally so formalized that for several finite sets which we want to obtain, simultaneous equations where they are the unknowns are given as follows and are obtained by solving them.

\[ s_i = s_{i_1} \cup s_{i_2} \cup \ldots \cup s_{i_{n-1}} \cup d_i, \text{ for } i = 1, 2, \ldots, n. \]

Traditionally, the above equations are solved as shown below: All the unknown sets \( s_i \), \( 1 \leq i \leq n \), are initialized as empty. Then the values of the right parts of the equations are computed and then they are assigned to the left unknown variables. Using these obtained values, the right parts are again computed and assigned to the left. This recursive computation is continued until the values obtained become unchanged.

Our method derives the similar kind of equations, which is, however, different from them as shown bellow on having coefficients \( \gamma_{ij} \), \( 1 \leq i, j \leq n \), of which each value is either \( \varphi \) or \( \lambda \).

\[ s_i = \gamma_{i1}s_1 \cup \gamma_{i2}s_2 \cup \ldots \cup \gamma_{in}s_n \cup d_i, \text{ for } i = 1, 2, \ldots, n. \]

They are as a whole written as a equation of matrix form \( s = \Gamma s \cup d \) and have a solution (the minimum solution, i.e., the minimum fixed point) \( s = \Gamma^* d \). For computing \( \Gamma^* \), the closure of \( \Gamma \), we can use directly the efficient methods, such as the Warshall's theorem, of computing the positive closure of a Boolean matrix. There occurs only one computation of sets of symbols, i.e., a multiplication of \( \Gamma^* \) and \( d \).

In the above our equations, the coefficients and the constant terms were each given a formula to compute the value by Boolean computation. For any nonterminal \( X \), the structure of connections of symbols in the right part of BNF which defines \( X \) is represented by both a transition matrix \( A_X \) and a final vector \( f_X \) (Boolean vector). Furthermore, for each symbol \( \sigma \) in \( A_X \), a Boolean matrix called the partial adjacent matrix
denoted by $\partial_\sigma A_k$, which shows the contribution of $\sigma$ to the structure of the connections, is abstracted. The above coefficients and the constant terms are defined using these Boolean matrices and vectors. Therefore, they are computed from a given grammar as Boolean computation.

As one more application of the above methods to the problems of language processing, this paper presents a method of computing the $LL(1)$ look-ahead sets for a simple recursive-descent syntax-directed translator in order to make its parse deterministic. We have already given a theory to generate such kind of translators and have proven the correctness of the generation[4]. Furthermore, a practical generator called MYLANG has been developed as a generator of attributed recursive-descent syntax-directed translators[5].

From a given translation scheme, a machine model of recursive-descent syntax-directed translator is constructed. For the machine, this paper gave a method to compute the $LL(1)$ director sets necessary to make its parse deterministic.

2. RECURSIVE DESCENT SEQUENTIAL TRANSDUCER SYSTEM

An input symbol set, an output symbol set and a nonterminal symbol set are denoted by $T$, $\Delta$, and $N$, respectively. $T \cup \Delta \cup N$ is shown by $V$. For the nonterminal set $N$, a set of all nonterminals which may generate the empty string $\varepsilon$ is denoted by $N$. For a given set $S$, the cardinal number of $S$ is written as $\#(S)$ and the power set of $S$ as $\mathcal{P}S$. The empty set is written as $\emptyset$ and the empty string set ($\varepsilon$) as $\lambda$. For sets $x$ and $y$, $x + y$ and $x \cdot y$ (usually written as $xy$) are interpreted as the set union and the set concatenation, respectively.

For a given set $S$, a matrix $A = (a_{ij}) = ([A]_{ij})$, $a_{ij} \in \mathcal{P}S$, is called an $S$-matrix. The $\lambda$-matrix of which each diagonal element is $\lambda$ and the others are all $\emptyset$ is denoted by $E$. The sum and the product of matrices are defined as the same as the case in usual algebra. Vectors are all written as Gothic letters.

For a given alphabet $\Sigma$, a function $\partial : \Sigma \times \mathcal{P}\Sigma \to \mathcal{P}\lambda$ is defined as $\partial_\sigma S = \lambda$ if $\sigma \in S$; $\partial_\sigma S = \emptyset$ if otherwise. The definition is extended for a $V$-
matrix $A = (a_{ij})$ as $\delta_\alpha A = (\delta_\alpha a_{ij})$, and it is called the partial adjacent matrix.

The Extended Bacus Naur Form (EBNF) is inherently used for description of a grammar. However in this paper, EBNF is used for describing a kind of translation scheme by regarding the terminal symbol occurred in the EBNF either as an input symbol or as an output symbol. That is, EBNF used here is such an extension of Backus Naur Form that its right part is allowed to be written as the form of a regular expression generated from the given alphabet $V$. This kind of EBNF is interpreted as one of the translation scheme. And the translation defined by this scheme is called simple syntax-directed translation.

In the EBNF, a terminal symbol set is a union of $T$ and $\Delta$. Each string $w$ defined by this EBNF is a string generated from $T \cup \Delta$. For the string $w$, when every output symbol in $w$ is replaced by $\varepsilon$, an input string $w_i$ is obtained, and similarly an output string $w_o$ associated with $w_i$ is obtained. Thus, it can be said that $w$ shows the translation from $w_i$ to $w_o$. We call this translation scheme regular translation form or RTF for short.

The following simple example defines the translation from a simple arithmetic expression to the postfix notation.

$$E = T ( ' + ' T [+] )^* ;$$
$$T = 'i' [i] + ('E')' ;$$

Fig.1 An example of RTF.

Where $N = \{ E, T \}$, $T = \{ ' + ', 'i', '(', ')', ' ' \}$ and $\Delta = \{ [+]$,[i]$ \}$. The meta symbol ";" is used to separate equations each other, and furthermore is treated as if it is an output symbol $[;]$ which outputs $\varepsilon$.

Thus the above equations are treated as follows:

$$E = ( T ( ' + ' T [+] )^* )[;]$$
$$T = ('i' [i] + ('E')' )[;]$$

Fig.2 An example of RTF.

For each nonterminal $X$, there exists in RTF one and only one equation $X = R_X$, which is called the $X$-defining RTF equation. The right part $R_X$ is a regular expression generated from $V = N \cup T \cup \Delta$. Therefore,
for each nonterminal $X$, we can have an automaton $X$ accepting $R_X$ over $V^*$. From Fig.2, we have the following two automata $E$ and $T$.

Fig.3 A system of automata

Generally, $R_X$ is able to be unfolded by means of introducing appropriate parameters $x_{X_1}, x_{X_2}, \ldots, x_{X_n}$, resulting in obtaining an $n_X$-dimensional right linear equations, as follows:

$$X = x_{X_1}, x_{X_i} = \sum_{j=1}^{n_{X_i}} a_{X_{ij}} x_{X_j} + c_{X_i}, \ i = 1, 2, \ldots, n_X$$  \hspace{1cm} (2.1)

or as a matrix representation $x_X = A_X x_X + c_X$.

From Fig.1 or Fig.2, the $E$-defining and the $T$-defining RTF equations are obtained respectively, as follows:

$$E = x_{T_1}, \begin{bmatrix} x_{T_1} \\ x_{T_2} \\ x_{T_3} \\ x_{T_4} \\ x_{T_5} \\ x_{T_6} \end{bmatrix} = \begin{bmatrix} \varphi & T & \varphi & \varphi & \varphi \\ \varphi & \varphi & 'i' & \varphi & [;] \\ \varphi & \varphi & \varphi & T & \varphi \\ \varphi & \varphi & \varphi & \varphi & \varphi \\ \varphi & \varphi & \varphi & \varphi & \varphi \\ \varphi & \varphi & \varphi & \varphi & \varphi \end{bmatrix} \begin{bmatrix} x_{T_1} \\ x_{T_2} \\ x_{T_3} \\ x_{T_4} \\ x_{T_5} \\ x_{T_6} \end{bmatrix} \hspace{1cm} (2.2)$$

$$T = x_{T_1}, \begin{bmatrix} x_{T_1} \\ x_{T_2} \\ x_{T_3} \\ x_{T_4} \\ x_{T_5} \\ x_{T_6} \end{bmatrix} = \begin{bmatrix} \varphi & 'i' & \varphi & 'i' & \varphi & \varphi \\ \varphi & \varphi & [i] & \varphi & \varphi & \varphi \\ \varphi & \varphi & \varphi & \varphi & E & \varphi \\ \varphi & \varphi & \varphi & \varphi & \varphi & \varphi \\ \varphi & \varphi & \varphi & \varphi & \varphi & \varphi \\ \varphi & \varphi & \varphi & \varphi & \varphi & \varphi \end{bmatrix} \begin{bmatrix} x_{T_1} \\ x_{T_2} \\ x_{T_3} \\ x_{T_4} \\ x_{T_5} \\ x_{T_6} \end{bmatrix} \hspace{1cm} (2.3)$$

For a given RTF equation, $n_X$-dimensional $\lambda$-vectors $(\lambda \varphi \cdots \varphi), (\varphi \varphi \cdots \varphi)$ and $^t(\lambda \lambda \cdots \lambda)$ are denoted by $i_X$, $\varphi_X$ and $\lambda_X$, respectively.

As shown in eqs. (2.2) and (2.3), because of introducing $[;]$ into RTF, for any $X \in N$, $c_X$ becomes always a special column $\lambda$-vector such that its last constituent is $\lambda$ and the others are all $\varphi$. We write the vector as $f_X$. Namely, $f_X = ^t(\varphi \varphi \cdots \varphi \lambda)$.

For each nonterminal $X$, the $X$-defining RTF equation $X = R_X$ is transformed into an automaton as shown in Fig.3. It is called the $X$-
defining automaton and is denoted by $A_X$, which is described formally as follows:

$$A_X(S_X, V, x_{x_1}, x_{x_2}, \ldots, x_{x_n}, \tau_X), \quad X \in N,$$

(2.4)

where $S_X$ : a set of states ($x_{x_1}, x_{x_2}, \ldots, x_{x_n}$),
$V$ : a set of transition(input) symbols : $T \cup N \cup \Delta$,
$x_{x_1}$ : the initial state,
$x_{x_2}$ : the final state,
$\tau_X$ : the transition function : $S_X \times V \rightarrow S_X$,
$\tau_X(x_{x_1}, \sigma) = x_{x_2}$ iff $\sigma \in [A_X]_{x_2}$ iff $[\theta_\sigma A_X]_{x_2} = \lambda$.

Here, the language over $V^*$ accepted by $A_X$ is written as $T_V(A_X)$.

All automata $A_X$, $X \in N$ are linked together by means of a recursive-call mechanism as shown below and works as a syntax-directed translator. Namely, for the nonteminal set $N = \{ X_0, X_1, \ldots, X_n \}$ of a given RTF, a system of multiple sequential transducers $A = (A_{X_0}, A_{X_1}, \ldots, A_{X_n})$ is defined and is called recursive descent sequential transducer system or RDSTS for short.

This system $A$ starts from $A_{X_0}$ and moves as follows: Now, let the currently active machine be $A_X$, the current state be $x_{x_1}$, and the current input look-ahead symbol be $t$. Then, for state-transition $\tau_X(x_{x_1}, \sigma) = x_{x_2}$, $A_X$ performs one of the following operations:

1. When $\sigma = t' \in T$ and $t' = t$, $A_X$ transits to $x_{x_1}$ and inputs a new look-ahead symbol as $t$.

2. When $\sigma = \delta (\neq [;]) \in \Delta$, $A_X$ outputs $\delta$ and $x_{x_1}$ transits to $x_{x_2}$.

3. When $\sigma = Y \in N$, $A_X$ calls $A_Y$, i.e., the current state becomes $x_{x_1}$, the initial state of $A_Y$, and $A_X$ pauses.

4. When $\sigma = [;]$, $A_X$ returns control to the machine (say $A_Z$) which has called $A_X$. Let the state transition where $A_Z$ has called $A_X$ be $\tau_Z(x_{x_2}, X) = x_{x_1}$, then the next state becomes $x_{x_1}$.

The behavior of the above system is generally nondeterministic. In order to make it deterministic if possible, sets of symbols called director sets are used, which are shown in the following sections.

For $A_X$, state $x_{x_1}$ is said to be $\varepsilon$-accessible to state $x_{x_2}$ if $x_{x_1}$ is accessible to $x_{x_2}$ via a sequence of transitions caused by only the empty
string \( \varepsilon \). A nonterminal \( X \) which derives the empty string \( \varepsilon \) \( (X \Rightarrow^* \varepsilon) \) is called the \( \varepsilon \)-generating nonterminal. A set of all \( \varepsilon \)-generating nonterminals in \( N \) is denoted by \( N' \). Each output symbol \( \delta \in \Delta \) has the same effect as \( \varepsilon \) for state-transition as shown in the above (2) and (4). For \( A_X \), a \( \lambda \)-matrix \( C'_X = \sum_{Y \in V'} \delta_Y A_X + \sum_{\delta \in \Delta} \delta A_X \) is defined and called the \( (N' \cup \Delta) \)-adjacent matrix of \( A_X \). For the \( C'_X \), it holds that \( x_{i1} \) is \( \varepsilon \)-accessible to \( x_{ij} \) if and only if \( [C'_X]_{ij} = \lambda \).

3. FIRST SETS

For a nonterminal \( X \in N \), a set of input symbols appearing at the first position of sentential forms derived from \( X \) is defined as

\[
First(X) = \{ t \in T \mid X \Rightarrow^* tv, \; w \in V^* \} \tag{3.1}
\]

and is called the First (symbol) set.

In order to compute the First set for the \( X \)-defining \( \operatorname{KTR} \) equation \( X = R_X \), we prepare a function \( \gamma : N \times V \rightarrow \mathcal{P} \lambda \), as follows:

\[
\gamma_{\lambda X} = \lambda \quad \text{if there exists a string } \xi \sigma w \in R_X \quad \text{such that } \xi \in (N' \cup \Delta)^*, \; \sigma \in V \text{ and } w \in V^*; \tag{3.2}
\]

\( = \varphi \) otherwise.

For our \( x_X = A_X x_X + c_X \), using the \( (N' \cup \Delta) \)-adjacent matrix \( C'_X \), the above \( \gamma_{\lambda X} \) is given as follows:

\[
\gamma_{\lambda X} = i_X C'_X \delta_X A_X \lambda_X = \delta_X \omega_X, \; \omega_X = i_X C'_X A_X \lambda_X \tag{3.3}
\]

Thus we have

\[
First(X) = \sum_{Y \in N} \gamma_{\lambda Y} First(Y) + d_X \tag{3.4}
\]

where \( d_X = \sum_{t \in T} \gamma_{\lambda X} \{ t \} = \omega_X \cap T \).

Now, in order to solve eq.(3.4), we define two \#(\( N \))\( \times \)\#(\( N \)) \( \lambda \)-matrix \( \Gamma \), as follows:

\[
\begin{bmatrix}
First(X_0) \\
First(X_1) \\
\vdots \\
First(X_n)
\end{bmatrix}, \quad
\begin{bmatrix}
d_{X_0} \\
d_{X_1} \\
\vdots \\
d_{X_n}
\end{bmatrix}, \quad
\begin{bmatrix}
\gamma_{X_0 X_0} & \gamma_{X_0 X_1} & \cdots & \gamma_{X_0 X_n} \\
\gamma_{X_1 X_0} & \gamma_{X_1 X_1} & \cdots & \gamma_{X_1 X_n} \\
\vdots & \vdots & \ddots & \vdots \\
\gamma_{X_n X_0} & \gamma_{X_n X_1} & \cdots & \gamma_{X_n X_n}
\end{bmatrix}
\]

Then, eq.(3.4) becomes \( u = \Gamma u + d \) and has the solution \( u = \Gamma^{-1} d \).

For \( t \in T \) and \( \delta \in \Delta \), the definition of the First set is naturally extended as \( First(t) = \{ t \} \) and \( First(\delta) = \varphi \), respectively.
4. FOLLOW SETS

For a symbol $\sigma$ in a given RTF, a set of input symbols which follow $\sigma$ directly or through a sequence composed of $\varepsilon$-generating nonterminals and output symbols in sentential forms derived from the RTF is called Follow set of $\sigma$ and denoted by $\text{Follow}(\sigma)$. Among the Follow sets for a given RTF, we can find the following relations:

Case 1. If there exists a nonterminal $Y$ such that the right part of the $Y$-defining RTF equation contains a string $\alpha \varepsilon \rho \beta$, where $\alpha, \beta \in V^*$, $\varepsilon \in (N \cup \Delta)^*$ and $\rho \in \Sigma$, then $\text{Follow}(\sigma)$ contains $\text{First}(\rho)$.

Case 2. If there exists a nonterminal $Y$ such that the right part of the $Y$-defining RTF equation contains a string $\alpha \varepsilon \xi$, where $\alpha \in V^*$, $\xi \in (N \cup \Delta)^*$, then $\text{Follow}(\sigma)$ contains $\text{Follow}(Y)$.

For a given $Y$-defining RTF $Y = R_Y$, let the $Y$-defining equation derived from it be $x_Y = A_Y x_Y + f_Y$ and the associated automaton $Y$ be $A_Y$.

Here, we define a function $d: V \times V \times N \rightarrow \rho \lambda$:

$$d_{\alpha \rho} = i_Y C_Y \sigma A_Y C_Y \rho A_Y C_Y f_Y,$$

which means that if there exists a string $\alpha \varepsilon \rho \omega$ in $R_Y = [A_Y]^* \sigma A_Y = \mathcal{T}_Y(A_Y)$, where $\varepsilon \in (N \cup \Delta)^*$ and $\sigma, \omega \in V^*$, then $d_{\alpha \rho} = \lambda$; otherwise $\phi$. That is, $d_{\alpha \rho} = \lambda$ iff $A_Y$ accepts $\alpha \varepsilon \rho \omega$ iff $A_Y$ has states $x_{Y1}$, $x_{Yt1}$, $x_{Yt2}$ and $x_{Yt3}$ such that $x_{Yt1}$ is accessible from the initial state $x_{Y1}$, $x_{Yt2}$ is $\varepsilon$-accessible from $x_{Yt1}$ and $x_{Yt3}$ is $\varepsilon$-accessible to the final state $x_{Yt3}$.

Next, a function $\theta: V \times N \rightarrow \rho \lambda$ is defined as

$$\theta_{\alpha \sigma} = i_Y C_Y \sigma A_Y C_Y f_Y,$$

which means that if there exists a string $\alpha \varepsilon \xi$ in $R_Y = [A_Y]^* \sigma A_Y = \mathcal{T}_Y(A_Y)$, where $\varepsilon \in V^*$ and $\xi \in N^*$, then $\theta_{\alpha \sigma} = \lambda$; otherwise $\phi$. Namely, $\theta_{\alpha \sigma} = \lambda$ iff $A_Y$ accepts $\alpha \varepsilon \xi$ iff $A_Y$ has states $x_{Yt1}$ and $x_{Yt2}$ such that $x_{Yt1}$ is accessible from the initial state $x_{Y1}$, $x_{Yt2}$ is $\varepsilon$-accessible from $x_{Yt1}$ and $x_{Yt3}$ is $\varepsilon$-accessible to the final state $x_{Yt3}$.

Using the above functions $d_{\alpha \rho}$ and $\theta_{\alpha \sigma}$, Case 1 and Case 2 are formally written, respectively, as follows:

For $\forall Y \in N, \forall \rho \in V : \text{Follow}(\sigma) \supseteq d_{\alpha \rho} \text{First}(\rho)$, \hspace{1cm} (4.3)

For $\forall Y \in N : \text{Follow}(\sigma) \supseteq \theta_{\alpha \sigma} \text{Follow}(Y)$, \hspace{1cm} (4.4)
From eqs. (4.3) and (4.4), the Follow set of $\sigma$ is defined as

$$\text{Follow}(\sigma) = \sum_{y \in N} \theta_{\sigma y} \text{Follow}(Y) + d_{\sigma},$$  \hspace{1cm} (4.5)

where

$$d_{\sigma} = \sum_{p \in V} d_{\sigma p} \text{First}(p),$$  \hspace{1cm} (4.6)

$$d_{qp} = \sum_{y \in N} d_{qpy}.$$  \hspace{1cm} (4.7)

For all nonterminals, put

$$u_N = \begin{bmatrix} \text{Follow}(X_0) \\ \text{Follow}(X_1) \\ \vdots \\ \text{Follow}(X_N) \end{bmatrix}, \quad d_N = \begin{bmatrix} d_{X_0} \\ d_{X_1} \\ \vdots \\ d_{X_N} \end{bmatrix}, \quad \Theta_{NN} = \begin{bmatrix} \theta_{X_0X_0} & \theta_{X_0X_1} & \cdots & \theta_{X_0X_N} \\ \theta_{X_1X_0} & \theta_{X_1X_1} & \cdots & \theta_{X_1X_N} \\ \vdots & \vdots & \ddots & \vdots \\ \theta_{X_NX_0} & \theta_{X_NX_1} & \cdots & \theta_{X_NX_N} \end{bmatrix}.$$  \hspace{1cm} (4.8)

Then we have the Follow sets of all nonterminals, as follows:

$$u_N = \Theta_{NN} u_N + d_N = \Theta_{NN}^* d_N.$$  \hspace{1cm} (4.9)

For each output symbol, we have its Follow sets from eq. (4.5).

5. DIRECTOR SETS

For each transition symbol $\sigma \in V$, a set of terminals used to make the state transition deterministic is defined as follows and called the Director set of $\sigma$:

$$\text{Director}(\sigma) = \text{First}(\sigma) + \Lambda(\sigma) \text{Follow}(\sigma)$$  \hspace{1cm} (5.1)

where

$$\Lambda(\sigma) = \lambda \text{ if } \sigma \in N' \text{ (i.e. } \sigma \Rightarrow \varepsilon \text{) or } \sigma \in \Delta$$

$$= \varphi \text{ otherwise.}$$  \hspace{1cm} (5.2)

For each state $x_{Xt}$ in automaton $A_X$ of a RDFAS $A$, if there are no two transitions $\tau_X(x_{Xt}, \sigma)$ and $\tau_X(x_{Xt}, \sigma')$ such that $\text{Director}(\sigma) \cap \text{Director}(\sigma') = \varphi$, the grammar from which the RDFAS $A$ is constructed is called LL(1). In this case, we can make the move of $A$ deterministic in the following manner. Namely, for a look-ahead input symbol $t$ at state $x_{Xt}$ in $A_X$, we make $A_X$ perform the state transition $\tau_X(x_{Xt}, \sigma)$ if $t \in \text{Director}(\sigma)$.

Similarly, we can easily obtain the more detailed director sets associated with not only a symbol but with the automaton or the state where the set is used.

For each symbol $\sigma \in N \cup \Delta$, the Follow set of $\sigma$ in $A_X$ is defined as

$$\text{Follow}(\sigma, X) = \theta_{\sigma X} \text{Follow}(X) + \sum_{p \in V} d_{\sigma pX} \text{First}(p),$$  \hspace{1cm} (5.3)

and the Director set of $\sigma$ in $A_X$ is thus defined as

$$\text{Director}(\sigma, X) = \text{First}(\sigma) + \Lambda(\sigma) \text{Follow}(\sigma, X).$$  \hspace{1cm} (5.4)
Furthermore, we can have, if necessary, the Director set of a transition symbol $\sigma$ from state $x_{X_1}$ in automaton $A_X$.

First, a function $d : \tilde{n}_X V X V N \rightarrow P \lambda$ is given as

$$d_{\lambda \sigma \rho \xi} = [d_{\sigma A_X C_X^* \rho A_X C_X^*}]_\lambda \tilde{n}_X,$$  \hspace{1cm} (5.5)

where $\tilde{n}_X = \{1, 2, \ldots, n_X\}$, which means that if there exists a string $\sigma \xi \rho w$ in $[A_X^*]_\lambda \tilde{n}_X$ such that $\sigma, \rho \in V, \xi \in N^*$ and $w \in V^*$, then $d_{\lambda \sigma \rho \xi} = \lambda$; otherwise $\varphi$.

Second, a function $\theta : \tilde{n}_X V X V N \rightarrow P \lambda$ is defined as

$$\theta_{\lambda \sigma \rho} = [\theta_{\sigma A_X C_X^* \rho}]_\lambda \tilde{n}_X,$$  \hspace{1cm} (5.6)

which means that if there exists a string $\sigma \xi \rho$ in $[A_X^*]_\lambda \tilde{n}_X$ such that $\sigma \in V$ and $\xi \in N^*$, then $\theta_{\lambda \sigma \rho} = \lambda$; otherwise $\varphi$.

For each transition $\tau_X(x_{X_1}, \sigma)$, the Follow set and the Director set of the transition are defined as follows:

$$\text{Follow}(x_{X_1}, \sigma) = \theta_{\lambda \sigma} \text{Follow}(X) + \sum_{\rho \in V} d_{\lambda \sigma \rho} \text{First}(\rho),$$  \hspace{1cm} (5.7)

$$\text{Director}(x_{X_1}, \sigma) = \text{First}(\sigma) + \Lambda(\sigma) \text{Follow}(x_{X_1}, \sigma).$$  \hspace{1cm} (5.8)

REFERENCES