数理解析研究所講究録
第 754 巻 1991 年 176-185

Deterministic Parse for Recursive Descent Syntax-Directed Translators

安在弘幸(Hiroyuki Anzai)
九州工業大学(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 algbra 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, existance 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 φ and the empty string set λ, and is necessarily contained in the language space generated from arbitary alphabet. This language space is, as a language semiring, isomrphic 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 sevearl finite sets which we want to obtain, simultaneous equations where they are the unknowns are given as follows and are obtained by solving them.

$$s_i = s_{i_1} \cup s_{i_2} \cup \cdots \cup s_{i_{r(i)}} \cup d_i, \text{ for } i = 1,2,\cdots,n.$$

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

Our method derives the similar kind of equations, which is, however, diffrent from them as shown bellow on having coffeficients γ_{ij}, ($1 \leq i$, $j \leq n$), of which each value is either φ or λ.

$$s_i = \gamma_{i1}s_1 \cup \gamma_{i2}s_2 \cup \cdots \cup \gamma_{in}s_n \cup d_i, \quad \text{for } i = 1,2,\cdots,n.$$

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

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 symmbol σ in A_X, a Boolean matrix called the partial adgacent matrix denoted by $\partial_\sigma A_X$, which shows the contribution of σ to the structue 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 nontrminal symbol set are denoted by T, Δ, 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 ε 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 φ and the empty string set $\{\varepsilon\}$ as λ. For sets x and y, $x + y$ and $x\cdot y$ (usually written as xy) are intepreted as the set union and the set concatination, 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 λ-matrix of which each diagonal element is λ and the others are all φ 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 writen as Gothic letters.

For a given alphabet Σ, a function $\partial : \Sigma\times\mathcal{P}\Sigma \rightarrow \mathcal{P}\lambda$ is defined as $\partial_\sigma S = \lambda$ if $\sigma\in S$; $\partial_\sigma S = \varphi$ if otherwise. The definition is extended for a V-

matrix $A = (a_{ij})$ as $\partial_\sigma A = (\partial_\sigma a_{ij})$, 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 Δ. 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 ε, 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 tanslation form* or *RTF* for short.

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

$$E = \; T \; (\; \text{'+'} \; T \; [+] \;)^* \quad ;$$
$$T = \; \text{'}i\text{'}[i] + \text{'('} \; \mathrm{E} \; \text{')'} \quad ;$$

Fig.1 An example of *RTF*.

Where $N = \{ E, T \}$, $T = \{ \text{'+'}, \text{'}i\text{'}, \text{'('}, \text{')'} \}$ and $\Delta = \{ [+], [i] \}$. The meta symbol ";" is used to seperate equations each other, and furthermore is treated as if it is an output symbol [;] which outputs ε. Thus the above equations are treated as follows:

$$E = (\; T \; (\; \text{'+'} \; T \; [+] \;)^* \;) [;]$$
$$T = (\text{'}i\text{'}[i] + \text{'('} \; \mathrm{E} \; \text{')'} \;) [;]$$

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.

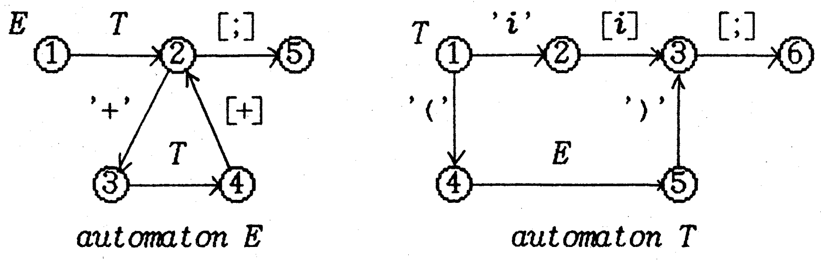

Fig.3 A system of automata

Generally, R_X is able to be unfolded by means of introducing appropriate parameters x_{X1}, x_{X2}, $\cdots$, x_{Xn_X}, resulting in obtaining an n_X-dimensional right linear equations, as follows:

$$X = x_{X1}, \quad x_{Xi} = \sum_{j=1}^{n_X} a_{Xij}\, x_{Xj} + c_{Xi}, \quad i = 1,\ 2,\ \cdots,\ n_X \qquad (2.1)$$

or as a matrix representation $\mathbf{x}_X = A_X \mathbf{x}_X + c_X$.

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

$$E = x_{E1}, \quad \begin{pmatrix} x_{E1} \\ x_{E2} \\ x_{E3} \\ x_{E4} \\ x_{E5} \end{pmatrix} = \begin{pmatrix} \varphi & T & \varphi & \varphi & \varphi \\ \varphi & \varphi & '+' & \varphi & [;] \\ \varphi & \varphi & \varphi & T & \varphi \\ \varphi & [+] & \varphi & \varphi & \varphi \\ \varphi & \varphi & \varphi & \varphi & \varphi \end{pmatrix} \begin{pmatrix} x_{E1} \\ x_{E2} \\ x_{E3} \\ x_{E4} \\ x_{E5} \end{pmatrix} + \begin{pmatrix} \varphi \\ \varphi \\ \varphi \\ \varphi \\ \lambda \end{pmatrix} \qquad (2.2)$$

$$T = x_{T1}, \quad \begin{pmatrix} x_{T1} \\ x_{T2} \\ x_{T3} \\ x_{T4} \\ x_{T5} \\ x_{T6} \end{pmatrix} = \begin{pmatrix} \varphi & 'i' & \varphi & '(' & \varphi & \varphi \\ \varphi & \varphi & [i] & \varphi & \varphi & \varphi \\ \varphi & \varphi & \varphi & \varphi & \varphi & [;] \\ \varphi & \varphi & \varphi & \varphi & E & \varphi \\ \varphi & \varphi & ')' & \varphi & \varphi & \varphi \\ \varphi & \varphi & \varphi & \varphi & \varphi & \varphi \end{pmatrix} \begin{pmatrix} x_{T1} \\ x_{T2} \\ x_{T3} \\ x_{T4} \\ x_{T5} \\ x_{T6} \end{pmatrix} + \begin{pmatrix} \varphi \\ \varphi \\ \varphi \\ \varphi \\ \varphi \\ \lambda \end{pmatrix}. \qquad (2.3)$$

For a given *RTF* equation, n_X-dimensional λ-vectors $(\lambda\ \varphi\ \cdots\ \varphi)$, $(\varphi\ \varphi\ \cdots\ \varphi)$ and ${}^t(\lambda\ \lambda\ \cdots\ \lambda)$ are denoted by i_X, φ_X and λ_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 λ-vector such that its last constituent is λ and the others are all φ. 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 $\mathcal{A}_X$, which is described formally as follows:

$$\mathcal{A}_X(S_X,\ V,\ x_{X1},\ x_{Xn_X},\ \tau_X\ ),\quad X \in N, \qquad (2.4)$$

where S_X : a set of states $\{ x_{X1}, x_{X2}, \cdots , x_{Xn_X} \}$,
V : a set of transition(input) symbols : $T \cup N \cup \Delta$,
x_{X1}: the initial state,
x_{Xn_X} : the final state,
τ_X : the transition function : $S_X \times V \rightarrow S_X$.
: $\tau_X(x_{Xi}, \sigma) = x_{Xj}$ iff $\sigma \in [A_X]_{ij}$ iff $[\partial_\sigma A_X]_{ij} = \lambda$.

Here, the language over V^* accepted by $\mathcal{A}_X$ is written as $\mathcal{T}_V(\mathcal{A}_X)$.

All automata $\mathcal{A}_X$, $X \in N$ 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 $N = \{ X_0, X_1, \cdots, X_\nu \}$ of a given *RTF*, a system of multiple sequential transducers $\mathcal{A} = (\mathcal{A}_{X_0}, \mathcal{A}_{X_1}, \cdots, \mathcal{A}_{X_\nu})$ is defined and is called *recursive descent sequential transducer system* or *RDSTS* for short.

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

(1) When $\sigma = t' \in T$ and $t' = t$, $\mathcal{A}_X$ transits to x_{Xj} and inputs a new look-ahead symbol *as* t.

(2) When $\sigma = \delta$ ($\neq$ [;]) $\in \Delta$, $\mathcal{A}_X$ outputs δ and x_{Xi} transits to x_{Xj}.

(3) When $\sigma = Y \in N$, $\mathcal{A}_X$ calls $\mathcal{A}_Y$, *i.e.*, the cuurent state becomes x_{Y1}, the initial state of $\mathcal{A}_Y$, and $\mathcal{A}_X$ pauses.

(4) When σ = [;], $\mathcal{A}_X$ returns control to the machine (say $\mathcal{A}_Z$) which has called $\mathcal{A}_X$. Let the state transition where $\mathcal{A}_Z$ has called $\mathcal{A}_X$ be $\tau_Z(x_{Zk}, X) = x_{Zl}$, then the next state becomes x_{Zl}.

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 $\mathcal{A}_X$, 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 by only the empty string ε. A nonterminal X which derives the empty string ε ($X \overset{*}{\Rightarrow} \varepsilon$) is called the *$\varepsilon$-generating* nonterminal. A set of all ε-generating nonterminals in N is denoted by N'. Each output symbol $\delta \in \Delta$ has the same effect as ε for state-tansition as shown in the above (2) and (4). For A_X, a λ-matrix $C'_X = \sum_{Y \in N'} \partial_Y A_X + \sum_{\delta \in \Delta} \partial_\delta A_X$ is defined and called the *$(N' \cup \Delta)$-adjacent* matrix of A_X. For the C'_X, *it holds that* x_{X_i} *is ε-accessible to* x_{X_j} 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

$$First(X) = \{ t \in T \mid X \overset{*}{\Rightarrow} tw,\ w \in V^* \} \qquad (3.1)$$

and is called the *First (symbol) set*.

In order to compute the First set for the X-defining *RTF* equation $X = R_X$, we prepare a function $\gamma : N \times V \to P\lambda$, *as follows*:

$$\begin{aligned} \gamma_{X\sigma} &= \lambda \quad \text{if there exists a string } \xi\sigma w \in R_X \\ &\qquad \text{such that } \xi \in (N' \cup \Delta)^*,\ \sigma \in V \text{ and } w \in V^*; \\ &= \varphi \quad \text{otherwise.} \end{aligned} \qquad (3.2)$$

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

$$\gamma_{X\sigma} = i_X C'^*_X \partial_\sigma A_X \lambda_X = \partial_\sigma \omega_X, \quad \omega_X = i_X C'^*_X A_X \lambda_X \qquad (3.3)$$

Thus we have

$$First(X) = \sum_{Y \in N} \gamma_{XY} First(Y) + d_X , \qquad (3.4)$$

$$\text{where } d_X = \sum_{t \in T} \gamma_{Xt} \{t\} = \omega_X \cap T.$$

Now, in order to solve eq.(3.4), we define two $\#(N)$-dimensional column vectors $\boldsymbol{u}$ and $\boldsymbol{d}$, and a $\#(N) \times \#(N)$ λ-matrix Γ, as follows:

$$\boldsymbol{u} = \begin{pmatrix} First(X_0) \\ First(X_1) \\ \vdots \\ First(X_\nu) \end{pmatrix}, \quad \boldsymbol{d} = \begin{pmatrix} d_{X_0} \\ d_{X_1} \\ \vdots \\ d_{X_\nu} \end{pmatrix}, \quad \Gamma = \begin{pmatrix} \gamma_{X_0X_0} & \gamma_{X_0X_1} & \cdots & \gamma_{X_0X_\nu} \\ \gamma_{X_1X_0} & \gamma_{X_1X_1} & \cdots & \gamma_{X_1X_\nu} \\ & & \cdots\cdots & \\ \gamma_{X_\nu X_0} & \gamma_{X_\nu X_1} & \cdots & \gamma_{X_\nu X_\nu} \end{pmatrix}$$

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

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 σ in a given *RTF*, a set of input symbols which follow σ directly or through a sequence composed of ε-generating nonterminals and output symbols in sentential forms derived from the *RTF* is called *Follow set* of σ and denoted by $Follow(\sigma)$. 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* $\alpha\sigma\xi\rho\beta$, *where* α, $\beta \in V^*$, $\xi \in (N' \cup \Delta)^*$ *and* σ, $\rho \in V$, *then* $Follow(\sigma)$ *contains* $First(\rho)$,

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

For a given Y-defining *RTF* $Y = R_Y$, let the Y-defining equation derived from it be $\boldsymbol{x}_Y = A_Y \boldsymbol{x}_Y + \boldsymbol{f}_Y$ and the associated automaton Y be $\mathcal{A}_Y$.

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

$$d_{\sigma\rho Y} = i_Y C_Y^* \, \partial_\sigma A_Y C_Y^* \, \partial_\rho A_Y C_Y^* \, f_Y, \qquad (4.1)$$

which means that if there exists a string $w\sigma\xi\rho w'$ in $R_Y = [A_Y^*]_{1n_Y} = \mathcal{T}_V(\mathcal{A}_Y)$, where $\xi \in (N' \cup \Delta)^*$ and w, $w' \in V^*$, then $d_{\sigma\rho Y} = \lambda$; otherwise φ. That is, $d_{\sigma\rho Y} = \lambda$ iff $\mathcal{A}_Y$ accepts $w\sigma\xi\rho w'$ iff $\mathcal{A}_Y$ has states x_{Yi_1}, x_{Yi_2}, x_{Yi_3} and x_{Yi_4} such that x_{Yi_1} is accessible from the initial state x_{Y1}, $x_{Yi_2} = \tau(x_{Yi_1}, \sigma)$, x_{Yi_3} is ε-accessible from x_{Yi_2}, $x_{Yi_4} = \tau(x_{Yi_3}, \rho)$, and x_{Yi_4} is accessible to the final state x_{Yn_Y}.

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

$$\theta_{\sigma Y} = i_Y C_Y^* \, \partial_\sigma A_Y C_Y^* \, f_Y, \qquad (4.2)$$

which means that if there exists a string $w\sigma\xi$ in $R_Y = [A_Y^*]_{1n_Y} = \mathcal{T}_V(\mathcal{A}_Y)$, where $w \in V^*$ and $\xi \in N'^*$, then $\theta_{\sigma Y} = \lambda$; otherwise φ. Namely, $\theta_{\sigma Y} = \lambda$ iff $\mathcal{A}_Y$ accepts $w\sigma\xi$ iff $\mathcal{A}_Y$ has states x_{Yi_1} and x_{Yi_2} such that x_{Xi_1} is accessible from the initial state x_{Y1}, $x_{Yi_2} = \tau(x_{Yi_1}, \sigma)$, and x_{Yi_2} is ε-accessible to the final state x_{Yn_Y}.

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

$$\text{For } \forall Y \in N,\ \forall \rho \in V : \quad Follow(\sigma) \geqq d_{\sigma\rho Y} First(\rho), \qquad (4.3)$$

$$\text{For } \forall Y \in N : \quad Follow(\sigma) \geqq \theta_{\sigma Y} Follow(Y). \qquad (4.4)$$

From eqs.(4.3) and (4.4), the Follow set of σ is defined as

$$Follow(\sigma) = \sum_{Y\in N} \theta_{\sigma Y} Follow(Y) + d_\sigma, \tag{4.5}$$

$$\text{where} \quad d_\sigma = \sum_{\rho\in V} d_{\sigma\rho} First(\rho), \tag{4.6}$$

$$d_{\sigma\rho} = \sum_{Y\in N} d_{\sigma\rho Y}. \tag{4.7}$$

For all nonterminals, put

$$u_N = \begin{pmatrix} Follow(X_0) \\ Follow(X_1) \\ \vdots \\ Follow(X_\nu) \end{pmatrix}, \quad d_N = \begin{pmatrix} d_{X_0} \\ d_{X_1} \\ \vdots \\ d_{X_\nu} \end{pmatrix}, \quad \Theta_{NN} = \begin{pmatrix} \theta_{X_0X_0} & \theta_{X_0X_1} & \cdots & \theta_{X_0X_\nu} \\ \theta_{X_1X_0} & \theta_{X_1X_1} & \cdots & \theta_{X_1X_\nu} \\ & & \cdots\cdots & \\ \theta_{X_\nu X_0} & \theta_{X_\nu X_1} & \cdots & \theta_{X_\nu X_\nu} \end{pmatrix} \tag{4.8}$$

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

$$u_N = \Theta_{NN} u_N + d_N = \Theta_{NN}^* d_N. \tag{4.9}$$

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 σ:

$$Director(\sigma) = First(\sigma) + \Lambda(\sigma)\, Follow(\sigma) \tag{5.1}$$

$$\text{where} \quad \Lambda(\sigma) = \lambda \quad \text{if } \sigma \in N' \text{ (i.e. } \sigma \overset{*}{\Rightarrow} \varepsilon) \text{ or } \sigma \in \Delta$$
$$= \varphi \quad \text{otherwise.} \tag{5.2}$$

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

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 σ in $\mathcal{A}_X$ is defined as

$$Follow(\sigma, X) = \theta_{\sigma X} Follow(X) + \sum_{\rho\in V} d_{\sigma\rho X} First(\rho), \tag{5.3}$$

and the *Director set* of σ in $\mathcal{A}_X$ is thus defined as

$$Director(\sigma, X) = First(\sigma) + \Lambda(\sigma) Follow(\sigma, X). \tag{5.4}$$

Furthermore, we can have, if necessary, the Director set of a transition symbol σ from state x_{Xi} in automaton $\mathcal{A}_X$.

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

$$d_{i\sigma\rho X} = [\partial_\sigma A_X C_X^* \, \partial_\rho A_X C_X^*]_{in_X}, \tag{5.5}$$

where $\tilde{n}_X = \{1, 2, \cdots, n_X\}$,

which means that if there exists a string $\sigma\xi\rho w$ in $[A_X^*]_{in_X}$ such that σ, $\rho \in V$, $\xi \in N^*$ and $w \in V^*$, then $d_{i\sigma\rho w} = \lambda$; otherwise φ.

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

$$\theta_{i\sigma X} = [\partial_\sigma A_X C_X^*]_{in_X}, \tag{5.6}$$

which means that if there exists a string $\sigma\xi$ in $[A_X^*]_{in_X}$ such that $\sigma \in V$ and $\xi \in N^*$, then $\theta_{i\sigma X} = \lambda$; otherwise φ.

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

$$Follow(x_{Xi}, \sigma) = \theta_{i\sigma X} Follow(X) + \sum_{\rho \in V} d_{i\sigma\rho X} First(\rho), \tag{5.7}$$

$$Director(x_{Xi}, \sigma) = First(\sigma) + \Lambda(\sigma)\, Follow(x_{Xi}, \sigma). \tag{5.8}$$

REFERENCES

[1]H.Anzai:*Almost Boolean algebraic computation of LALR(1) look-ahead sets*, Jour. of INFORMATION PROCESSING, IPSJ, Vol.14, No.1(1991).

[2]H.Anzai, S.Jiang and G.Shao: *Near-Boolean algebraic computation of LL (1) director sets*, Proc. of Second Makuhari Int'l Conf. on High Technology, Jan. 1991.

[3]W.Kuich and A.Salomaa: *Semiring, Automata, Languages*, Springer-Verlag, Berlin(1986).

[4]H.Anzai: *A theory of recursive-descent syntax-directed translator*, Proc. of the Int'l Comp. Symp. '80, 1171-1182(1980).

[5]H.Anzai and T.Yamanoue: *Fundamental concept of language processor generator MYLANG*, Systems and Computers in Japan, Vol.17, No.12, 23-35(1986).

[6]M.Gondran and M.Minoux: *Graphs and Algorithms*, John Wiley and Sons. New York, 84-128(1984).

[7]A.V.Aho, R.Sethi and J.D.Ullman: *Compilers, Principles, Techniques, and Tools*, Addison-Wesley, Readings, MA(1986).

[8]S.Warshall: *A theorem on Booleam matrices*, J.ACM 9(1962).