An LR parser with pre-determined reduction goals

Gyung-Ok Lee *, Kwang-Moo Choe

Department of Computer Science, Korea Advanced Institute of Science and Technology, 373-1 Kusong-dong, Yosung-ku, Taejon 305-701, South Korea

Received 15 March 1999; received in revised form 12 October 1999

Communicated by K. Iwama

Keywords: Compilers; LR parsing; Error recovery

1. Introduction

A reduction goal in LR parsing is known at reduction time, but one can sometimes predict the goal in advance. In this paper, we suggest a predictive LR parser based on the goal prediction. In the LR parser, each LR state has a goal and a suffix of the stack string such that the suffix stack string and some prefix of the remaining input string will be certainly reduced into the goal.

Two relations for the goal prediction are defined by analyzing grammar symbols (Section 3). Then we construct a labeled LR automaton (Section 4) in which each LR state has a set of labels, which informs an LR parser of pre-determinable goals. The notion of pre-determined goals can be applied to many problems including the computation of right context [1], phrase-level error recovery, and predictor [4] (Section 5).

2. Basic definitions

We freely use the notation and definitions in [2]. Throughout this paper, the symbol $G$ denotes an arbitrary, but fixed context-free grammar $G = (N, \Sigma, P, S)$ (let $V = N \cup \Sigma$). $G$ is assumed to be augmented in the sense that $P$ contains a special rule $S' \rightarrow S$, where $S'$ does not occur in any other rule. For $A \in N$, $G^A$ represents the reduced subgrammar with the start symbol $A$. Define

$$\text{FIRST}(\alpha) = \{ 1 : x \mid \alpha \Rightarrow^* x \text{ in } G, \ x \in \Sigma^* \}.$$

A pair of $[A \rightarrow \alpha, \beta, u]$ is an LR item if $A \rightarrow \alpha \beta \in P$ and $u \in \text{FIRST}(\Sigma^*S)$ where the symbol $\$ not in $\Sigma$ is the end marker of an input string. Let $I^G$ be the set of the LR items over $G$. A nondeterministic LR finite automaton of $G$ is $M^G = (I^G, V, \rightarrow, [S' \rightarrow .S, \$], \emptyset)$ with the set of states $I^G$, the set of transition symbols $V$, the initial state $[S' \rightarrow .S, \$]$, the set of final states $\emptyset$, and a transition function $\rightarrow$ which is defined from $I^G \times (V \cup \{ \$ \})$ to $2^{I^G}$:

(i) $[A \rightarrow \alpha, X\beta, u] \rightarrow X [A \rightarrow \alpha X\beta, u]$ for some $X \in V$;

(ii) $[A \rightarrow \alpha, B\beta, u] \rightarrow v [B \rightarrow \gamma, v]$ where $v \in \text{FIRST}(\beta u)$.

If $I_0 \rightarrow X_1 I_1 \rightarrow X_2 \ldots \rightarrow X_n I_n$ ($n \geq 0$) holds, then we write $I_0 \Rightarrow_{X_1;X_2;\ldots;X_n} I_n$. An LR automaton of $G$ is defined by $M(G) = (Q, V, \text{GOTO}, q_0, \emptyset)$ where $Q$ is the set of LR states, which is the smallest set of $\{q_0\} \cup \{\text{GOTO}(q, X) \mid q \in Q, X \in V\}$; a transition function $\text{GOTO}$ from $Q \times V$ to $Q$ is defined by

$$\text{GOTO}(q, X) = \{ J \mid I \rightarrow^* X J, \ I \in q \}.$$
q₀ is the initial state defined by [I | | S’ → .S, $] →*$ I); the set of final states is Ø. We abbreviate GOTO(q₀, θ) by VALID(θ) and [θ]. For q ∈ Q.

FIRST(q) = \{x ∈ FIRST(βu) | [A → αβ, u] ∈ q\}
and

KERNEL(q) = \{[A → αβ, u] ∈ q | α ≠ ε or A = S’\}.

A configuration of an LR parser is of the form $[e]…[θ]\alpha x\$ where $[e]…[θ]$ is the stack string, and x is the remaining input string. The initial configuration for input string x is $[e] x\$, and the accepting configuration is $[e] [\]$ x$. A configuration $[e]…[θ]\alpha x\$ is a valid LR parsing configuration if there exists a sequence of moves in M(G) $[e] x\$ $⇒* [e] [\] x\$. The LR parser has the following moves.

(i) shift action: $[e]…[θ]\alpha y\$ $⇒* [e]…[θ][\theta] α\$ y\$ if $[A → α. α β, u] ∈ VALID(θ)$;

(ii) reduce action: $[e]…[θ]\alpha x\$ $⇒* [e]…[θ][\eta] α\$ x\$ if $[A → α, u] ∈ VALID(θ)$ where $θ = [η] α$, $A ≠ S’$, and u = 1 : x$.

We consider only LR(1) grammars, and hence the LR parser is deterministic.

3. Two relations

We define a d relation which captures reduction goal sequences in the LR parsing and a Π relation which finds a pre-determinable goal by examining the graph associated with the d relation.

3.1. d relation

Let $A ∈ N$, $X ∈ V$, and $α ∈ V^*$. The d relation is defined by $A d^* X$ iff $A → α X β ∈ P$ for some $β ∈ V^*$. Let $A d^n A$ be $A d^m A$ and $A d^n X$, $n > 0$, be the composition of $A d^n α$ and $B d^n X$. The reflexive transitive closure of d relation, denoted by $d^*$, is defined by $\bigcup_{n≥0} d^n$. The directed graph associated with d relation is called the d-graph. A path in d-graph is a finite sequence $A_0 d^{n_1} A_1 d^{n_2} A_2 … A_{n-1} d^{n_n} A_n$. The notation $(A, α, X)$ represents a set of paths $[h | h = A_0 d^{n_1} A_1 d^{n_2} A_2 … A_{n-1} d^{n_n} A_n, A_0 = A, α = α_1 α_2 … α_n, A_n = X]$.

A path in d-graph enables us to infer a derivation form in G as shown in the following property.

Property 1. Let $A_i ∈ V$, $i = 0, 1, … n$. Then there exists a path in d-graph $A_0 d^{n_1} A_1 d^{n_2} A_2 … d^{n_n} A_n$ if there exists a derivation in G

$A_0 \Rightarrow_{r_m} α_1 A_1 β_1 \Rightarrow_{r_m} α_1 A_1 A_2 z_1 \Rightarrow_{r_m} α_1 α_2 A_2 β_2 z_1 \Rightarrow_{r_m} α_1 α_2 α_3 α_{n-1} A_n β_n z_n-1 z_{n-2}… z_1 \Rightarrow_{r_m} α_1 α_2 α_3 α_{n-1} A_n β_n z_n-1 z_{n-2}… z_1 \Rightarrow\text{for some } z_i \in Σ^*$ and $β_i ∈ V^*$, $i = 1, … n$.

Proof. The only if part can be proved by simple induction on n, and the if part can be true by directly applying the definition of d relation.

Note that any viable prefix of G is a viable stack string in M(G), and any viable stack string in M(G) is a viable prefix of G. Hence we get the following property from Property 1.

Property 2. Let $A_0 = S’$, $A_i ∈ N (i = 1, …, n - 1)$, and $A_n ∈ Σ$. Then there exists a path in d-graph $A_0 d^{n_1} A_1 d^{n_2} A_2 … d^{n_n} A_n$ if there exists a sequence of moves in M(G)

$[e]…[α_1]…[α_i]…[α_n] A_n y_n \$ $⇒* [e]…[α_1]…[α_i]…[α_n] A_n y_n \$ $⇒* [e]…[α_1]…[α_i] A_n y_n \$ $⇒* [e]…[α_1] A_n A_0 y_0 \$ for some $y_n, …, y_1, y_0 ∈ Σ^*$.

From Property 2, we know that all the possible reduction goal sequence from a valid LR parsing configuration $[e]…[α] α x\$ $x ∈ Σ^*$ can be found in the paths in $(S’, α, α)$. Consequently, a goal into which a reduction must be made at the configuration can be found by analyzing the related path set.

3.2. Π relation

Definition 1 (Π relation). Let $A, B ∈ N$, $α$ be a viable prefix of $G^A$, $α = βγ$, and $A ∈ Σ$ where $Ad^m β$. Then $(A, α) ∈ Π_B (B, γ)$ iff for each path $A_0 d^{n_1} A_1 d^{n_2} A_2 … A_{l+1} d^{n_{l+1}} A_{l+1} … A_{n-1} d^{n_n} A_n$ in $(A, α, α)$ where $A_0 = A, α = α_1 α_2 … α_n$, and $A_n = α$, there exists $l (1 ≤ l ≤ n)$ such that $A_l = B$ and $α_{l+1}…α_n = γ$. 
A;

Then we can observe that the paths in \( h \sim P \) are such that there exists a sequence of moves in \( M(G) \). Let \( \Pi_1 \) be the \( d \)-graph.

**Example 1.** Let \( G_1 = (\{S, A, C, B, X, Y\}, \{a, b, c\}, P_1, S) \) where \( P_1 = \{S \rightarrow A, S \rightarrow C, A \rightarrow BX, A \rightarrow BY, C \rightarrow Ba, B \rightarrow b, X \rightarrow BA, Y \rightarrow BC, X \rightarrow c\} \). The paths in \((A, BBB, b)\) and \((A, BBB, c)\) are shown in Fig. 2. Then we can observe that \((A, BBB) \Pi_{q_0} (A, B)\) and \((A, BBB) \Pi_{q_0} (A, B)\) hold.

The following property is a simple consequence of the definition of \( \Pi \) relation based on Property 2.

**Property 3.** Let \((A, \alpha) \Pi_{q_0} (B, \gamma)\) and \( \alpha = \beta \gamma \). A sequence of moves in \( M(G) \) such that
\[
[\epsilon]_x \gamma \Rightarrow^* [\epsilon]_x [\eta \alpha] \gamma \gamma \\
\Rightarrow^* [\epsilon]_x [\eta \alpha] \gamma \gamma \Rightarrow^* [\epsilon]_x [S] \gamma \gamma
\]
for some \( \eta \in V^* \) where \( 1 : y \gamma = a \) is always of the form
\[
[\epsilon]_x \gamma \Rightarrow^* [\epsilon]_x [\eta \alpha] \gamma \gamma \Rightarrow^* [\epsilon]_x [\eta \beta B] \gamma \gamma \\
\Rightarrow^* [\epsilon]_x [\eta \alpha] \gamma \gamma \Rightarrow^* [\epsilon]_x [S] \gamma \gamma.
\]

**4. A labeled LR automaton**

Two algorithms are presented: one is to label the LR states, and the other is to generate a pre-determined reduction goal during the LR parsing.

Each LR state \( q \) is labeled with \( (\hat{q}, A, \alpha) \) where \( \hat{q} \in Q, A \in N, \) and \( \alpha \in V^* \). The labeling means that for all valid LR parsing configuration \( [\epsilon]_x \gamma \Rightarrow^* [\eta \alpha] \gamma \gamma \gamma \) where \( \text{VALID}(\eta) = \hat{q} \) and \( \text{VALID}(\eta \alpha) = q \), there exists a sequence of moves in \( M(G) \).

\[
[\epsilon]_x \gamma \Rightarrow^* [\eta \alpha] \gamma \gamma \Rightarrow^* [\eta \alpha] \gamma \gamma \Rightarrow^* [\eta \alpha] \gamma \gamma.
\]

For a label \((\hat{q}, A, \alpha)\), \( \alpha \) is potentially infinite. The \( \Pi \) relation contributes the potentially infinite labeling process to be bounded in Algorithm 1, but the possibility of the infiniteness is still remained. A concept of a cyclic path set is introduced to limit the labeling process. Suppose that \((A, \alpha, \alpha)\) is given. Let \( M(G^\alpha) = (Q^\alpha, V^\alpha, Q_0^\alpha, B, \emptyset) \) be the LR automaton of \( G^\alpha \). If \( \alpha = X_1 ... X_n \), \( p_{i+1} \Rightarrow^{} GOTO (p_i, X_{i+1}), i = 0, 1, ..., n \) where \( p_0 = q_0^\alpha \) and \( X_{n+1} = \alpha \). For \( p_0, p_1, ..., p_n, p_{n+1}, \) if \( p_i = p_j (0 \leq i < j \leq n+1) \) and no other pair of \( p_0, p_1, ..., p_n, p_{n+1} \) is identical, then \( p_i, ..., p_j \) is a loop. We say that \((A, \alpha, \alpha)\) is cyclic if there exist more than two different values for \( i, 1 \leq i \leq n \) for some loop such that \( p_i, ..., p_j \) is a loop, and \((A, \alpha, \alpha)\) is divisible if \((A, \alpha) \Pi_{q_0} (B, \gamma)\) holds for some \( B \) and \( \gamma \). It would be noted that if \((A, \alpha, \alpha)\) is a non-divisible cyclic path set, then there exist an infinite number of non-divisible cyclic path sets that have the loop in common with \((A, \alpha, \alpha)\). It is thus reasonable to exclude the label which yields a non-divisible cyclic path set. In accordance with this notion, a label \((q, A, \alpha)\) is excluded from the label set in Algorithm 1 if \((A, \alpha, \alpha)\) where \( A = \alpha \alpha \) is a non-divisible cyclic path set. In Algorithm 1, LABEL is a table from \( Q \) to \( 2^{Q \times N \times V^*} \) that has the set of labels for each LR state.

The spending time in Algorithm 1 is proportional to \( n_Q \) and \( n_L \) where \( n_Q \) is the size of \( Q \) and \( n_L \) is the size of LABEL table. On the other hand, LABEL has \( O(n_Q) \) entries, and so the time complexity of Algorithm 1 is \( O(n_Q^2) \).

In Algorithm 1, the input argument \( \Pi_{\text{non-cyclic}} \) can be replaced by a subset \( \widehat{\Pi} \) of \( \Pi_{\text{non-cyclic}} \), which can be chosen depending on an application. The labeled LR automaton with \( \widehat{\Pi} \) is denoted by \( M(G, \widehat{\Pi}) \).
Input: $M(G)$ and $\Pi^\text{non-cyclic}$ where $\Pi^\text{non-cyclic} = \{(A, \alpha) \mid \{A, \alpha, a\} \text{ is non-cyclic}\}$. 
Output: LABEL table 
Method: 
1. LABEL($q_0$) = \{($q_0$, $S$, $\varepsilon$)} 
2. for all LR states $q$ except the initial state do LABEL($q$) = $\emptyset$ endfor 
3. repeat 
   for each $q \in Q$ do 
   (a) for each $(\hat{q}, A, \alpha) \in$ LABEL($q$) and $a \in$ FIRST($q$) do 
      for each $B$ and $\gamma$ such that $(A, \alpha) \Pi^\text{non-cyclic}_a (B, \gamma)$ do let $\hat{p}$ be GOTO($\hat{q}$, $\beta$) where $\alpha = \beta \gamma$ 
      (i) $\text{LABEL}(p) = \text{LABEL}(p) \cup \{(\hat{q}, A, \beta B)\}$ where $p =$ GOTO($\hat{p}$, $B$) 
      (ii) if there exists $[C \rightarrow \delta, a, \xi, w] \in q$ for some $C, \delta, \xi, w$ then 
          $\text{LABEL}(p) = \text{LABEL}(p) \cup \{(\hat{p}, B, \gamma a)\}$ where $p =$ GOTO($\hat{p}$, $a$) endif 
      (iii) if there exists $[C \rightarrow \delta, a] \in q$ for some $C, \delta$ then 
          $\text{LABEL}(p) = \text{LABEL}(p) \cup \{(\hat{p}, B, \rho C)\}$ where $p =$ GOTO($\hat{p}$, $\rho C$) and $\gamma = \rho \delta$ endif 
   endfor 
   (b) if $(\hat{q}, A, \alpha)$ such that $(A, \alpha) \Pi^\text{non-cyclic}_a (B, \gamma)$ then 
      remove $(\hat{q}, A, \alpha)$ from LABEL($q$) endif 
   endfor 
   until LABEL table does not change 

Algorithm 1 (Labeling of the LR states).

Example 2. For $G_1$ in Example 1, assume that $\hat{\Pi}$ is composed of $(S, B) \Pi_b (A, B), (S, B) \Pi_b (A, B), (A, BBB) \Pi_b (A, B)$, and $(A, BBB) \Pi_c (A, B)$. Then Fig. 3 shows the labeled LR automaton $M(G_1, \hat{\Pi})$.

Lemma 1. Let $(\hat{q}, A, \alpha) \in$ LABEL($q$). Then for all valid LR parsing configuration $[\varepsilon]\ldots[\eta]\ldots[\eta a]\ldots x$ $S$ where $\text{VALID}([\eta]) = \hat{q}$ and $\text{VALID}([\eta a]) = q$, there exists a sequence of moves in $M(G)$ such that $[\varepsilon]\ldots[\eta]\ldots[\eta a]\ldots x \Rightarrow^* [\varepsilon]\ldots[\eta]\ldots[\eta A]\ldots y$.

Proof. This lemma can be proved by induction on the iteration number of the (a) block in Algorithm 1. Assume that this lemma holds for the label $(\hat{q}, A, \alpha) \in$ LABEL($q$) in the (a) block. Then we can verify using Property 3 that this lemma holds for each newly added labels to LABEL table according to the steps (i)–(iii); we can get the same thing for the steps (iv), (v) by the principle of labeling. The detail process is omitted. □

The converse of Lemma 1 is not true; although we can predict the move $q_0q_5q_12|bcS \Rightarrow^* q_0q_1|S$ in $M(G_1)$, LABEL($q_12$) does not contain the label $(0, S, Bb)$.

An LR parser with pre-determined goals can be generated by the following algorithm, which is built on the labeled LR automaton. In the algorithm, CURRENT_LABEL has the set of pre-determined goals for the given stack string.
Input: a stack string $[\varepsilon] \ldots [	heta]$ of a configuration

Output: CURRENT_LABEL for the given stack string $[\varepsilon] \ldots [	heta]$

Method:

1. CURRENT_LABEL = \{(A, \alpha) \mid (\text{VALID}(\eta), A, \alpha) \in \text{LABEL}(\text{VALID}(\theta)) \text{ where } \theta = \eta \alpha\}

2. if CURRENT_LABEL = \emptyset then
   (a) find the longest prefix $[\varepsilon] \ldots [\eta]$ of $[\varepsilon] \ldots [\theta]$ for which CURRENT_LABEL is a non-empty set
   (b) CURRENT_LABEL = \{(B, \beta \gamma) \mid (B, \beta) \in \text{CURRENT_LABEL for } [\varepsilon] \ldots [\eta] \text{ where } \theta = \eta \gamma\} \text{ endif}

Algorithm 2 (Generation of pre-determined goals).

To analyze the running time of Algorithm 2, suppose that $q$ is the current state and $n$ is the length of the stack string. Step 1 needs the time $c_L \times c_M$ where $c_L$ is the number of the entries of $\text{LABEL}(q)$ and $c_M$ is the matching time between the stack string and a label in $\text{LABEL}(q)$. Step 2 requires the time $n \times (c_L \times c_M)$ at the worst case. Hence the time $O(n)$ is necessary to obtain CURRENT_LABEL. On the other hand, the matching time $c_M$ in Step 1 is unnecessary when $\text{LABEL}(q)$ contains the label $(\tilde{q}, A, \alpha)$ for all $\alpha$-predecessor $\tilde{q}$ of $q$ where $A$ and $\alpha$ are fixed. At this point, only $(A, \alpha)$ without $\tilde{q}$ can be stored in LABEL, and CURRENT_LABEL is obtained directly from LABEL table. We believe that in practical grammars, the label of most LR states does not require predecessor information, and it is very seldom that Steps 2(a) and 2(b) are demanded. Hence we
can say that the time $O(1)$ is needed to obtain \textsc{current}_{-label} in practice.

**Example 3.** Assume that input string “bbc” is given in $M(G_1, \widehat{T})$ of Example 2. Then Table 1 shows the change of \textsc{current}_{-label} from the initial configuration to the accepting configuration.

The following theorem follows from Lemma 1.

**Theorem 1.** Let $(A, \alpha)$ be in \textsc{current}_{-label} for a stack string $[\varepsilon] \ldots [\theta]$. For all valid LR parsing configuration $[\varepsilon] \ldots [\theta]$ $\langle \varepsilon \rangle S$, there exists a sequence of moves in $M(G)$ such that $[\varepsilon] \ldots [\theta] \langle \varepsilon \rangle S \Rightarrow^* [\varepsilon] \ldots [\theta][\eta] \langle \eta \rangle S$ where $\theta = \eta \alpha$.

**5. Applications**

This section shows several applications of pre-determined reduction goals.

**5.1. Computation of right context**

The right context [1] is useful in error repair of an LR based parser, and its computation will be shown to be efficiently factored using pre-determined reduction goals.

For $\alpha \in V^+$ and $R \in 2^{V^+}$, $\alpha \cdot R$ means $[\alpha \beta] | \beta \in R$; for $Q, R \in 2^{V^+}$, $Q \cdot R$ means $[\alpha \beta] | \alpha \in Q \cdot \beta \in R$. We present the definition of right context for the convenience of readers.

**Definition 2 (Right context) [1].** For a stack string $q_0q_1 \ldots q_n$, let $X_i$ be an entry symbol of $q_i$ for all $i, 1 \leq i \leq n$ and $q_i A \Rightarrow \text{GOTO}(q_i, A)$. Then the right context of $q_0q_1 \ldots q_n$ is defined as follows:

\[
\text{RC}(q_0q_1 \ldots q_n) = \bigcup_{e \in \text{Kernel}(q_n)} [A \rightarrow X_{i+1} \ldots X_n. \beta, u] \quad [A \rightarrow X_{i+1} \ldots X_n. \beta, u] \]

where $1 \leq i \leq n$.

\[
\text{RCI}(q_0q_1 \ldots q_n, [A \rightarrow X_{i+1} \ldots X_n. \beta, u]) = \begin{cases} 
\beta \cdot \text{RC}(q_0q_1 \ldots q_{i}, \beta), & \text{if } [A \rightarrow X_{i+1} \ldots X_n. \beta, u] \neq [S' \rightarrow S., .]; \\
\varepsilon, & \text{otherwise.}
\end{cases}
\]

We will express the computation of RC in terms of path sets in $d$-graph. For it, the $d$ relation is refined by adding the subscript as $A d^n_\alpha X$ if $A \rightarrow \alpha X \beta \in P$, $X \in V$; the relation is extended by adding $\varepsilon$ as $A d^n_\alpha \varepsilon$ if $A \rightarrow \alpha \in P$. Let $A d^n_\alpha A$ be $A d^n_\alpha$. The $n$th power of this relation, $n > 0$ is defined as follows.

Let $A_0 \rightarrow \alpha_1 A_1 \beta_1, A_1 \rightarrow \alpha_2 A_2 \beta_2, \ldots, A_{n-1} \rightarrow \alpha_n A_n \beta_n$ in $P$.

Then $A_0 d^{\alpha_1}_1 A_1, A_1 d^{\alpha_2}_2 A_2, \ldots, A_{n-1} d^{\alpha_n}_n A_n$.

We write $A_0 d^{\alpha_1 \beta_1}_1 \cdots d^{\alpha_n \beta_n}_n A_n$.

The reflexive transitive closure of the relation, denoted by $A d^n_\alpha X$, is defined by $\bigcup_{n \geq 0} A d^n_\beta X$. For $A \in N, \alpha \in V^*, X \in V \cup \{\varepsilon\}$, the right context of a path set $(A, \alpha, X)$ is defined as $\text{RCP}(A, \alpha, X) = \{ \beta | A d^n_\beta X \}$. Note that

\[
\text{RCP}(S', X_1 \ldots X_{n-1}, X_n) = \{ \beta \theta | S' d^{X_1} \ldots X_{n-1} A d^{X_{n+1}} \ldots X_n \}
\]
For $G$, let $X_i$ be an entry symbol of $q_i$ for all $i$, $1 \leq i \leq n$. Then

$$ RC(q_0q_1 \ldots q_n) = RCP(S', X_1 \ldots X_n-1, X_n). $$

(Here if $n = 1$, then $X_1 \ldots X_n-1 = \varepsilon$.)

The computation of RCP can be divided into several substeps depending on the associated paths.

**Property 5.** Assume that each path $A d^{\alpha} X$ is a composition of a path $A d^{\alpha} B$ and a path $B d^{\alpha} X$ where $\alpha = \beta \gamma$. Then

$$ RCP(A, \alpha, X) = RCP(B, \gamma, X) \cdot RCP(A, \beta, B). $$

Using Properties 4, 5, and Theorem 1, we can prove the following theorem.

**Theorem 2.** For $G$, let $X_i$ be an entry symbol of $q_i$ for all $i$, $1 \leq i \leq n$. Assume that $(A, X_{m+1} \ldots X_n)$ for some $m$, $0 \leq m \leq n$ is in CURRENT_LABEL for a stack string $q_0q_1 \ldots q_n$. Then

$$ RC(q_0q_1 \ldots q_n) = RCP(A, X_{m+1} \ldots X_n-1, X_n) \cdot RCP(q_0q_1 \ldots q_mq_m, A) $$

where $q_m, A = GOTO(q_m, A)$.

**Proof.** According to Theorem 1 and as $M(G)$ is deterministic, a sequence of moves in $M(G)$

$$ [\varepsilon] x S \Rightarrow^* [\varepsilon] \ldots [X_1 \ldots X_n] y S \Rightarrow^* [\varepsilon] [S] y S $$

is always of the form

$$ [\varepsilon] x S \Rightarrow^* [\varepsilon] \ldots [X_1 \ldots X_n] y S \Rightarrow^* [\varepsilon] [X_1 \ldots X_m A] u S \Rightarrow^* [\varepsilon] [S] [S]. $$

Then by Property 2, a path $S' d^{x_1 \ldots x_n-1} X_n$ is always of the form $S' d^{x_1 \ldots x_m} A d^{x_{m+1} \ldots x_n-1} X_n$. Therefore, we get

$$ RCP(S', X_1 \ldots X_{n-1}, X_n) = RCP(A, X_{m+1} \ldots X_{n-1}, X_n) \cdot RCP(S', X_1 \ldots X_m, A) $$

from Property 5. Finally, we conclude

$$ RC(q_0q_1 \ldots q_n) = RCP(A, X_{m+1} \ldots X_{n-1}, X_n) \cdot RC(q_0q_1 \ldots q_mq_m, A) $$

by Property 4. □

Here the computation of $RC(q_0q_1 \ldots q_n)$ is factored in respect to $RC(q_0q_1 \ldots q_mq_m, A)$.

**Example 4.** In $M(G_1, \tilde{F})$ of Example 2, let us compute the right context for stack strings $\sigma q_0q_5q_9$ according to Theorem 2. Note CURRENT_LABEL = $\{(A, BB)\}$, and $RCP(A, B, B) = [A, C]$ for $A d^{x_2} B$ and $A d^{x_2} B$. Since $GOTO(q_9, A) = q_{11}$, $RC(\sigma q_0q_5q_9) = [A, C] \cdot RC(\sigma q_0q_9_{11})$.

The factorization makes it possible to avoid redundant computation as discussed in [1]. The several efficient computations according to [1] can be considered as special cases of the factorization shown in Theorem 2. The work, unfortunately, cannot factor out the common part $RC(\sigma q_0q_9_{11})$ in Example 4. In fact, the work can be applied to the only restricted LR states such that pre-determined goals are easily obtained by the inspection of the related LR states.

A deterministic algorithm for computing right context requires a strategy to select one label among several labels in CURRENT_LABEL. The strategy will depend on the application of right context.

### 5.2. Refinement of feasible reduction goals

The feasible reduction goal is a useful concept in phrase-level error recovery [3], and the pre-determined goal is a useful refinement of the feasible reduction goal. The previous work [3] suggested the heuristic strategies that find an error phrase [3] with a unique feasible reduction goal. A unique feasible reduction goal, however, does not guarantee that some prefix of
the remaining input string will be certainly reduced to the goal. On the other hand, a pre-determined goal guarantees it, and the heuristic search is not needed.

5.3. Predictors of subgrammars

A predictor is a useful string for global error recovery or validation [4], but it is seldom in a grammar. The prediction of reduction goal enables us to use a predictor of a subgrammar. As an example, assume that $z$ is a prefix predictor [4] of $G^A$, and $\alpha$ is the unique canonical prefix [4] of $G^A$ for $z$. Then in a configuration $[\varepsilon]...[\theta]|z\alpha\$ , if $(A, \beta) \in \text{CURRENT}\_\text{LABEL}$ for $[\varepsilon]...[\theta]$ , then $\beta$ must be equal to $\alpha$.

6. Concluding remarks

We suggested the predictive LR parser and showed some applications. Although our work was performed on the LR(1) automaton, it is extensible to an LR($k$) automaton, $k > 1$ in a straightforward way.

References