ELSEVIER

Information Processing Letters 72 (1999) 189–196

Information Processing Letters

www.elsevier.com/locate/ipl

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 $S' \to S$, 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

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

A pair of $[A \to \alpha.\beta, u]$ is an *LR item* if $A \to \alpha\beta \in P$ and $u \in \mathrm{FIRST}(\Sigma^*\$)$ where the symbol \$ not in Σ 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 $\widetilde{M}^G = (I^G, V, \to, [S' \to .S, \$], \emptyset)$ with the set of states I^G, the set of transition symbols V, the initial state $[S' \to .S, \$]$, the set of final states $\emptyset$, and a transition function $\to$ which is defined from $I^G \times (V \cup \{\varepsilon\})$ to 2^{I^G}:

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

(ii) $[A \to \alpha.B\beta, u] \to^\varepsilon [B \to .\gamma, v]$ where $v \in \mathrm{FIRST}(\beta u)$.

If $I_0 \to^{X_1} I_1 \to^{X_2} \cdots \to^{X_n} I_n$ $(n \geqslant 0)$ holds, then we write $I_0 \to^{*X_1X_2\ldots X_n} I_n$. An *LR automaton* of G is defined by $M(G) = (Q, V, \mathrm{GOTO}, q_0, \emptyset)$ where Q is the set of LR states, which is the smallest set of $\{q_0\} \cup \{\mathrm{GOTO}(q, X) \mid q \in Q,\ X \in V\}$; a transition function GOTO from $Q \times V$ to Q is defined by

$$\mathrm{GOTO}(q, X) = \{J \mid I \to^{*X} J,\ I \in q\};$$

* Corresponding author. Email: golee@ruby.kaist.ac.kr.

0020-0190/99/\$ – see front matter
PII: S0020-0190(99)00146-5

q_0 is the initial state defined by $\{I \mid [S' \to .S, \$] \to^* I\}$; the set of final states is $\emptyset$. We abbreviate GOTO(q_0, θ) by VALID(θ) and $[\theta]$. For $q \in Q$,

$$\mathrm{FIRST}(q) = \{x \in \mathrm{FIRST}(\beta u) \mid [A \to \alpha.\beta, u] \in q\}$$

and

$$\mathrm{KERNEL}(q) = \{[A \to \alpha.\beta, u] \in q \mid \alpha \neq \varepsilon \text{ or } A = S'\}.$$

A *configuration* of an LR parser is of the form $[\varepsilon]\ldots[\theta] \| x\$$ where $[\varepsilon]\ldots[\theta]$ is the stack string, and x is the remaining input string. The initial configuration for input string x is $[\varepsilon] \| x\$$, and the accepting configuration is $[\varepsilon][S] \| \$$. A configuration $[\varepsilon]\ldots[\theta] \| y\$$ is a *valid LR parsing configuration* if there exists a sequence of moves in $M(G)$ $[\varepsilon] \| x\$ \Rightarrow^* [\varepsilon]\ldots[\theta] \| y\$ \Rightarrow^* [\varepsilon][S] \| \$$. The LR parser has the following moves.

(i) shift action: $[\varepsilon]\ldots[\theta] \| ay\$ \Rightarrow [\varepsilon]\ldots[\theta][\theta a] \| y\$$ if $[A \to \alpha.a\beta, u] \in \mathrm{VALID}(\theta)$;

(ii) reduce action: $[\varepsilon]\ldots[\theta] \| x\$ \Rightarrow [\varepsilon]\ldots[\eta][\eta A] \| x\$$ if $[A \to \alpha., u] \in \mathrm{VALID}(\theta)$ where $\theta = \eta\alpha$, $A \neq S'$, 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 $A \in N$, $X \in V$, and $\alpha \in V^*$. The d relation is defined by $A\, d^{\alpha}\, X$ iff $A \to \alpha X\beta \in P$ for some $\beta \in V^*$. Let $A\, d^{0^{\varepsilon}}\, A$ be $A\, d^{\varepsilon}\, A$ and $A\, d^{n^{\alpha\beta}}\, X$, $n > 0$, be the composition of $A\, d^{n-1^{\alpha}}\, B$ and $B\, d^{\beta}\, X$. The reflexive transitive closure of d relation, denoted by d^*, is defined by $\bigcup_{n \geqslant 0} d^n$. The directed graph associated with d relation is called the *d-graph*. A path in d-graph is a finite sequence $A_0\, d^{\alpha_1}\, A_1\, d^{\alpha_2}\, A_2 \ldots A_{n-1}\, d^{\alpha_n}\, A_n$. The notation $\langle A, \alpha, X\rangle$ represents a set of paths $\{h \mid h = A_0\, d^{\alpha_1}\, A_1\, d^{\alpha_2}\, A_2 \ldots A_{n-1}\, d^{\alpha_n}\, A_n, A_0 = A, \alpha = \alpha_1\alpha_2\ldots\alpha_n, A_n = X\}$.

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

Property 1. *Let $A_i \in V$, $i = 0, 1, \ldots, n$. Then there exists a path in d-graph $A_0\, d^{\alpha_1}\, A_1\, d^{\alpha_2}\, A_2 \ldots d^{\alpha_{n-1}}\, A_{n-1}\, d^{\alpha_n}\, A_n$ iff there exists a derivation in G*

$$\begin{aligned}
A_0 &\Rightarrow_{rm} \alpha_1 A_1 \beta_1 \Rightarrow^*_{rm} \alpha_1 A_1 z_1 \Rightarrow_{rm} \alpha_1\alpha_2 A_2 \beta_2 z_1 \\
&\Rightarrow^*_{rm} \alpha_1\alpha_2\ldots\alpha_{n-1} A_{n-1} z_{n-1} z_{n-2} \ldots z_1 \\
&\Rightarrow_{rm} \alpha_1\alpha_2\ldots\alpha_{n-1}\alpha_n A_n \beta_n z_{n-1} z_{n-2} \ldots z_1 \\
&\Rightarrow^*_{rm} \alpha_1\alpha_2\ldots\alpha_{n-1}\alpha_n A_n z_n z_{n-1} z_{n-2} \ldots z_1 \\
&\qquad \textit{for some } z_i \in \Sigma^* \textit{ and } \beta_i \in V^*,\ i = 1, \ldots, n.
\end{aligned}$$

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 $A_0 = S'$, $A_i \in N$ $(i = 1, \ldots, n-1)$, and $A_n \in \Sigma$. Then there exists a path in d-graph $A_0\, d^{\alpha_1}\, A_1 \ldots A_{l-1}\, d^{\alpha_l}\, A_l \ldots A_{n-1}\, d^{\alpha_n}\, A_n$ iff there exists a sequence of moves in $M(G)$*

$$\begin{aligned}
&[\varepsilon]\ldots[\alpha_1]\ldots[\alpha_1\ldots\alpha_l\ldots\alpha_n] \| A_n y_n\$ \\
&\quad \Rightarrow^+ [\varepsilon]\ldots[\alpha_1]\ldots[\alpha_1\ldots\alpha_l A_l] \| y_l\$ \\
&\quad \Rightarrow^* [\varepsilon]\ldots[\alpha_1][\alpha_1 A_1] \| y_1\$ \\
&\quad \Rightarrow^* [\varepsilon][A_0] \| y_0\$ \quad \textit{for some } y_n, \ldots, y_1,\ y_0 \in \Sigma^*.
\end{aligned}$$

From Property 2, we know that all the possible reduction goal sequence from a valid LR parsing configuration $[\varepsilon]\ldots[\alpha] \| ax\$$, $x \in \Sigma^*$ can be found in the paths in $\langle S', \alpha, a\rangle$. 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 $A, B \in N$, α be a viable prefix of G^A, $\alpha = \beta\gamma$, and $a \in \Sigma$ where $A\, d^{*^{\alpha}}\, a$. Then $(A, \alpha)\, \Pi_a\, (B, \gamma)$ iff for each path $A_0\, d^{\alpha_1}\, A_1\, d^{\alpha_2}\, A_2 \ldots A_l\, d^{\alpha_{l+1}}\, A_{l+1} \ldots A_{n-1}\, d^{\alpha_n}\, A_n$ in $\langle A, \alpha, a\rangle$ where $A_0 = A$, $\alpha = \alpha_1\alpha_2\ldots\alpha_n$, and $A_n = a$, there exists l $(1 \leqslant l \leqslant n)$ such that $A_l = B$ and $\alpha_{l+1}\ldots\alpha_n = \gamma$.

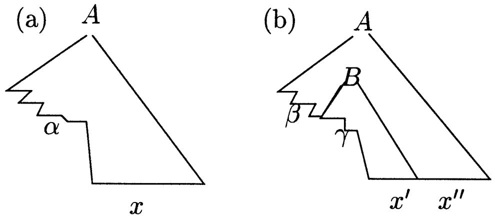


Fig. 1. Derivation trees where $1: x = a$, $\alpha = \beta\gamma$, and $x = x'x''$.

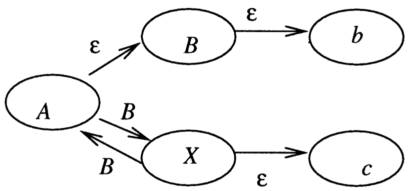


Fig. 2. The d-graph.

The $(A, \alpha)\, \Pi_a\, (B, \gamma)$ implies that in the tree (a) in Fig. 1, the suffix part γ of the string α and a prefix x' of the terminal string x must be derived from the goal B as shown in the tree (b).

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

The following property is a simple consequence of the definition of Π relation based on Property 2.

Property 3. *Let $(A, \alpha)\, \Pi_a\, (B, \gamma)$ and $\alpha = \beta\gamma$. A sequence of moves in $M(G)$ such that*

$$\begin{aligned}[\varepsilon]\|\, x\$ &\Rightarrow^* [\varepsilon]\ldots[\eta\alpha]\|\, y\$ \\ &\Rightarrow^* [\varepsilon]\ldots[\eta A]\|\, z\$ \Rightarrow^* [\varepsilon][S]\|\, \$\end{aligned}$$

for some $\eta \in V^*$ where $1: y\$ = a$ is always of the form

$$\begin{aligned}[\varepsilon]\|\, x\$ &\Rightarrow^* [\varepsilon]\ldots[\eta\alpha]\|\, y\$ \Rightarrow^* [\varepsilon]\ldots[\eta\beta B]\|\, w\$ \\ &\Rightarrow^* [\varepsilon]\ldots[\eta A]\|\, z\$ \Rightarrow^* [\varepsilon][S]\|\, \$.\end{aligned}$$

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 $[\varepsilon]\ldots[\eta]\ldots[\eta\alpha]\|\, x\$$ where $\mathrm{VALID}(\eta) = \hat{q}$ and $\mathrm{VALID}(\eta\alpha) = q$, there exists a sequence of moves in $M(G)$

$$[\varepsilon]\ldots[\eta]\ldots[\eta\alpha]\|\, x\$ \Rightarrow^* [\varepsilon]\ldots[\eta][\eta A]\|\, y\$.$$

For a label $(\hat{q}, A, \alpha)$, α is potentially infinite. The Π 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 $\langle A, \alpha, a\rangle$ is given. Let $M(G^A) = (Q^A, V^A, \mathrm{GOTO}^A, q_0^A, \emptyset)$ be the LR automaton of G^A; $\alpha = X_1 \ldots X_n$; $p_{i+1} = \mathrm{GOTO}^A(p_i, X_{i+1})$, $i = 0, 1, \ldots, n$ where $p_0 = q_0^A$ and $X_{n+1} = a$. For $p_0, p_1, \ldots, p_n, p_{n+1}$, if $p_i = p_j$ $(0 \leqslant i < j \leqslant n+1)$ and no other pair of $p_0, p_1, \ldots, p_n, p_{n+1}$ is identical, then $p_i, \ldots, p_j$ is a *loop*. We say that $\langle A, \alpha, a\rangle$ is *cyclic* if there exist more than two different values for i, $1 \leqslant i \leqslant n$ for same loop such that $p_i, \ldots, p_j$ is a loop, and $\langle A, \alpha, a\rangle$ is *divisible* if $(A, \alpha)\, \Pi_a\, (B, \gamma)$ holds for some B and γ. It would be noted that if $\langle A, \alpha, a\rangle$ 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 $\langle A, \alpha, a\rangle$. It is thus reasonable to exclude the label which yields a non-divisible cyclic path set. In accordance with this notion, a label $(\hat{q}, A, \alpha)$ is excluded from the label set in Algorithm 1 if $\langle A, \alpha', a\rangle$ where $\alpha = \alpha' a$ 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 $\mathrm{O}(n_Q)$ entries, and so the time complexity of Algorithm 1 is $\mathrm{O}(n_Q^2)$.

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 $\Pi^{\text{non-cyclic}} = \{(A,\alpha)\,\Pi_a\,(B,\gamma) \mid \langle A,\alpha,a\rangle$ is non-cyclic$\}$.
Output: LABEL table
Method:

1. $\mathrm{LABEL}(q_0) = \{(q_0, S, \varepsilon)\}$
2. **for** all LR states q except the initial state **do** $\mathrm{LABEL}(q) = \emptyset$ **endfor**
3. **repeat**
   for each $q \in Q$ **do**
   (a) **for** each $(\hat{q}, A, \alpha) \in \mathrm{LABEL}(q)$ and $a \in \mathrm{FIRST}(q)$ **do**
   - **for** each B and γ such that $(A,\alpha)\,\Pi_a^{\text{non-cyclic}}\,(B,\gamma)$ **do** let $\hat{p}$ be $\mathrm{GOTO}(\hat{q},\beta)$ where $\alpha = \beta\gamma$
     (i) $\mathrm{LABEL}(p) = \mathrm{LABEL}(p) \cup \{(\hat{q}, A, \beta B)\}$ where $p = \mathrm{GOTO}(\hat{p}, B)$
     (ii) **if** there exists $[C \to \delta.a\xi, w] \in q$ for some C, δ, ξ, w **then**
     $\mathrm{LABEL}(p) = \mathrm{LABEL}(p) \cup \{(\hat{p}, B, \gamma a)\}$ where $p = \mathrm{GOTO}(q, a)$ **endif**
     (iii) **if** there exists $[C \to \delta., a] \in q$ for some C, δ **then**
     $\mathrm{LABEL}(p) = \mathrm{LABEL}(p) \cup \{(\hat{p}, B, \rho C)\}$ where $p = \mathrm{GOTO}(\hat{p}, \rho C)$ and $\gamma = \rho\delta$ **endif**
     endfor
   - **if** there are no B and γ such that $(A,\alpha)\,\Pi_a^{\text{non-cyclic}}\,(B,\gamma)$ **then**
     (iv) **if** there exists $[C \to \delta.a\xi, w] \in q$ for some C, δ, ξ, w **then**
     $\mathrm{LABEL}(p) = \mathrm{LABEL}(p) \cup \{(\hat{q}, A, \alpha a)\}$ where $p = \mathrm{GOTO}(q, a)$ **endif**
     (v) **if** there exists $[C \to \delta., a] \in q$ for some C, δ **then**
     $\mathrm{LABEL}(p) = \mathrm{LABEL}(p) \cup \{(\hat{q}, A, \beta C) \mid \beta C \neq A$ or A is left recursive nonterminal$\}$
     where $p = \mathrm{GOTO}(\hat{q}, \beta C)$ and $\alpha = \beta\delta$ **endif**
     endif

   endfor
   (b) **if** $(\hat{q}, A, \alpha)$ such that $\langle A, \alpha', a\rangle$ is cyclic where $\alpha = \alpha' a$, is newly added to $\mathrm{LABEL}(q)$ **then**
   remove $(\hat{q}, A, \alpha)$ from $\mathrm{LABEL}(q)$ **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 $\widehat{\Pi}$ is composed of $(S,B)\,\Pi_b\,(A,B)$, $(S,B)\,\Pi_c\,(A,B)$, $(A,BBB)\,\Pi_b\,(A,B)$, and $(A,BBB)\,\Pi_c\,(A,B)$. Then Fig. 3 shows the labeled LR automaton $M(G_1, \widehat{\Pi})$.

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

$$[\varepsilon]\ldots[\eta]\ldots[\eta\alpha] \| x\$ \Rightarrow^* [\varepsilon]\ldots[\eta][\eta A] \| y\$.$$

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 $(\hat{q}, A, \alpha) \in \mathrm{LABEL}(q)$ 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 $q_0q_5q_{12} | bc\$ \Rightarrow^* q_0q_1 | \$$ in $M(G_1)$, $\mathrm{LABEL}(q_{12})$ 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.

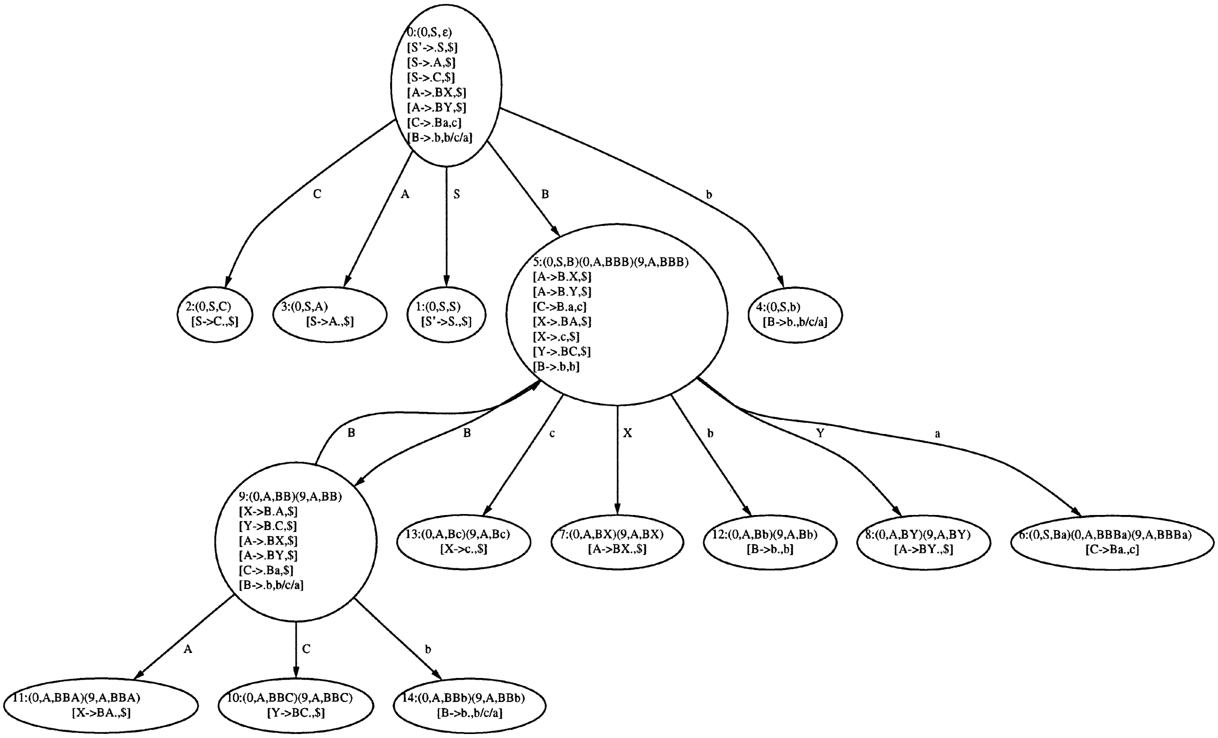


Fig. 3. The labeled $M(G_1, \widehat{\Pi})$.

Input: a stack string $[\varepsilon]\ldots[\theta]$ of a configuration
Output: CURRENT_LABEL for the given stack string $[\varepsilon]\ldots[\theta]$
Method:

1. CURRENT_LABEL $= \{(A, \alpha) \mid (\text{VALID}(\eta), A, \alpha) \in \text{LABEL}(\text{VALID}(\theta))$ 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 $= \{(B, \beta\gamma) \mid (B, \beta) \in$ CURRENT_LABEL for $[\varepsilon]\ldots[\eta]\}$ where $\theta = \eta\gamma$
   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 LABEL(q) and c_M is the matching time between the stack string and a label in 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 LABEL(q) contains the label $(\hat{q}, A, \alpha)$ for all α-predecessor $\hat{q}$ of q where A and α are fixed. At this point, only (A, α) without $\hat{q}$ 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

Table 1

Configuration CURRENT_LABEL	$q_0 \mid bbbc\$$ (S, ε)	$\vdash q_0q_4 \mid bbc\$$ (S, b)	$\vdash q_0q_5 \mid bbc\$$ (S, B)	$\vdash q_0q_5q_{12} \mid bc\$$ (A, Bb)
$\vdash q_0q_5q_9 \mid bc\$$ (A, BB)	$\vdash q_0q_5q_9q_{14} \mid c\$$ (A, BBb)	$\vdash q_0q_5q_9q_5 \mid c\$$ (A, BBB)	$\vdash q_0q_5q_9q_5q_{13} \mid \$$ (A, Bc)	$\vdash q_0q_5q_9q_5q_7 \mid \$$ (A, BX)
$\vdash q_0q_5q_9q_{11} \mid \$$ (A, BBA)	$\vdash q_0q_5q_7 \mid \$$ (A, BX)	$\vdash q_0q_3 \mid \$$ (S, A)	$\vdash q_0q_1 \mid \$$ (S, S)	

can say that the time O(1) is needed to obtain CURRENT_LABEL in practice.

Example 3. Assume that input string "*bbbc*" is given in $M(G_1, \widehat{\Pi})$ of Example 2. Then Table 1 shows the change of CURRENT_LABEL from the initial configuration to the accepting configuration.

The following theorem follows from Lemma 1.

Theorem 1. *Let* (A, α) *be in* CURRENT_LABEL *for a stack string* $[\varepsilon]\ldots[\theta]$. *For all valid LR parsing configuration* $[\varepsilon]\ldots[\theta] \mid x\$$, *there exists a sequence of moves in* $M(G)$ *such that*

$$[\varepsilon]\ldots[\theta] \mid x\$ \Rightarrow^* [\varepsilon]\ldots[\eta][\eta A] \mid y\$ \quad \textit{where } \theta = \eta\alpha.$$

5. Applications

This section shows several applications of predetermined 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 $\{\alpha\beta \mid \beta \in R\}$; for $Q, R \in 2^{V^*}$, $Q \cdot R$ means $\{\alpha\beta \mid \alpha \in Q,\ \beta \in R\}$. 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, $1 \leqslant i \leqslant n$ and $q_{t,A} = \mathrm{GOTO}(q_t, A)$. Then the right context of $q_0q_1\ldots q_n$ is defined as follows:

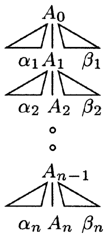
$$\mathrm{RC}(q_0q_1\ldots q_n) = \bigcup_{\substack{[A\to X_{t+1}\ldots X_n.\beta,u]\\ \in\mathrm{KERNEL}(q_n)}} \mathrm{RCI}\big(q_0q_1\ldots q_n,\ [A \to X_{t+1}\ldots X_n.\beta, u]\big)$$

where $1 \leqslant t \leqslant n$.

$$\mathrm{RCI}\big(q_0q_1\ldots q_n, [A \to X_{t+1}\ldots X_n.\beta, u]\big) = \begin{cases} \beta \cdot \mathrm{RC}(q_0q_1\ldots q_tq_{t,A}), \\ \qquad \text{if } [A \to X_{t+1}\ldots X_n.\beta, u] \neq [S' \to S., \$]; \\ \varepsilon, \quad \text{otherwise.} \end{cases}$$

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 $A\,d^{\alpha}_{\beta}\,X$ if $A \to \alpha X\beta \in P$, $X \in V$; the relation is extended by adding ε as $A\,d^{\alpha}_{\varepsilon}\,\varepsilon$ if $A \to \alpha \in P$. Let $A\,d^{0^\varepsilon}_{\varepsilon}\,A$ be $A\,d^{\varepsilon}_{\varepsilon}\,A$. The nth power of this relation, $n > 0$ is defined as follows.

Let $A_0 \to \alpha_1A_1\beta_1, A_1 \to \alpha_2A_2\beta_2, \ldots,$
$A_{n-1} \to \alpha_nA_n\beta_n$ in P.
Then $A_0\,d^{\alpha_1}_{\beta_1}A_1, A_1\,d^{\alpha_2}_{\beta_2}A_2, \ldots,$
$A_{n-1}\,d^{\alpha_n}_{\beta_n}A_n$.
We write $A_0d^{n^{\alpha_1\alpha_2\ldots\alpha_n}}_{\beta_n\ldots\beta_2\beta_1}A_n$.



The reflexive transitive closure of the relation, denoted by $A\,d^{*^\alpha}_{\beta}\,X$, is defined by $\bigcup_{n\geqslant 0} A\,d^{n^\alpha}_{\beta}\,X$. For $A \in N$, $\alpha \in V^*$, $X \in V \cup \{\varepsilon\}$, the *right context of a path set* $\langle A, \alpha, X\rangle$ is defined as $\mathrm{RCP}\langle A, \alpha, X\rangle = \{\beta \mid A\,d^{*^\alpha}_{\beta}\,X\}$. Note that

$$\mathrm{RCP}\langle S', X_1\ldots X_{n-1}, X_n\rangle = \big\{\beta\theta \mid S'\,d^{*^{X_1\ldots X_t}}_{\theta}\ A\,d^{X_{t+1}\ldots X_{n-1}}_{\beta}\ X_n\big\}$$

$$= \bigcup_{\substack{[A\to X_{t+1}\ldots X_n.\beta,u]\\ \in \mathrm{KERNEL}(q_n)}} \beta \cdot \{\theta \mid S' \, d_\theta^{*^{X_1\ldots X_t}} \, A\}$$

$$= \bigcup_{\substack{[A\to X_{t+1}\ldots X_n.\beta,u]\\ \in \mathrm{KERNEL}(q_n)}} \beta \cdot \mathrm{RCP}\langle S', X_1\ldots X_t, A\rangle$$

where $q_n = \mathrm{GOTO}(q_0, X_1\ldots X_n)$.

Then we can observe that the computation of RCP has the same structure with that of RC, and RC can be expressed by RCP as shown in the following property.

Property 4. *For G, let X_i be an entry symbol of q_i for all i, $1 \leqslant i \leqslant n$. Then*

$$\mathrm{RC}(q_0q_1\ldots q_n) = \mathrm{RCP}\langle S', X_1\ldots X_{n-1}, X_n\rangle.$$

(*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 $A\, d^{*^\beta} B$ and a path $B\, d^{*^\gamma} X$ where $\alpha = \beta\gamma$. Then*

$$\mathrm{RCP}\langle A, \alpha, X\rangle = \mathrm{RCP}\langle B, \gamma, X\rangle \cdot \mathrm{RCP}\langle A, \beta, B\rangle.$$

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, $1 \leqslant i \leqslant n$. Assume that $(A, X_{m+1}\ldots X_n)$ for some m, $0 \leqslant m \leqslant n$ is in* CURRENT_LABEL *for a stack string $q_0q_1\ldots q_n$. Then*

$$\mathrm{RC}(q_0q_1\ldots q_n) = \mathrm{RCP}\langle A, X_{m+1}\ldots X_{n-1}, X_n\rangle \cdot \mathrm{RC}(q_0q_1\ldots q_m q_{m,A})$$

where $q_{m,A} = \mathrm{GOTO}(q_m, A)$.

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

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

is always of the form

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

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

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

from Property 5. Finally, we conclude

$$\mathrm{RC}(q_0q_1\ldots q_n) = \mathrm{RCP}\langle A, X_{m+1}\ldots X_{n-1}, X_n\rangle \cdot \mathrm{RC}(q_0q_1\ldots q_m q_{m,A})$$

by Property 4. □

Here the computation of $\mathrm{RC}(q_0q_1\ldots q_n)$ is factored in respect to $\mathrm{RC}(q_0q_1\ldots q_m q_{m,A})$.

Example 4. In $M(G_1, \widehat{\Pi})$ of Example 2, let us compute the right context for stack strings $\sigma q_9 q_5 q_9$ according to Theorem 2. Note CURRENT_LABEL $= \{(A, BB)\}$, and $\mathrm{RCP}\langle A, B, B\rangle = \{A, C\}$ for $A\, d_A^{*^B} B$ and $A\, d_C^{*^B} B$. Since $\mathrm{GOTO}(q_9, A) = q_{11}$,

$$\mathrm{RC}(\sigma q_9 q_5 q_9) = \{A, C\} \cdot \mathrm{RC}(\sigma q_9 q_{11}).$$

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 $\mathrm{RC}(\sigma q_9 q_{11})$ 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 α is the unique *canonical prefix* [4] of G^A for z. Then in a configuration $[\varepsilon]\ldots[\theta]|\,zx\$$, if $(A,\beta) \in$ CURRENT_LABEL for $[\varepsilon]\ldots[\theta]$, then β must be equal to α.

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

[1] M.-S. Jung, K.-M. Choe, T. Han, An efficient computation of right context for LR-based error repair, Inform. Process. Lett. 49 (2) (1994) 63–72.

[2] S. Sippu, E. Soisalon-Soininen, Parsing Theory, Vols. I and II, Springer, Berlin, 1990.

[3] S. Sippu, E. Soisalon-Soininen, A syntax-error-handing technique and its experimental analysis, ACM Trans. Program. Languages Systems 5 (4) (1983) 656–679.

[4] K.C. Tai, Predictor of context-free grammars, SIAM. J. Comput. 9 (3) (1980) 653–664.