공학 연구논문집 제24권 제2호 pp. 109~123, 울산대학교
Journal of Engineering Research Vol. 24, No. 2, pp. 109~123, 1993.
University of Ulsan.

On the Efficient Resolution of LALR(1) Parsing Conflicts†

Myung-Joon Lee* · Yang-Su Park* · Young-Phil Cheung**
*Department of Computer Science · **Department of Computer Engineering

Abstract

For a given LR(1) grammar which is not LALR(1), we propose a new method for resloving parsing conflicts which may occur while the LALR(1) parser for the grammar processes an input string. When a parsing conflict occurs at an LR(0) state, the method identifes the LR(1) state, which would be the top of the parsing stake if LR(1) parsing was performed,by the use of the current content of the parsing stack and a *conflict-resolving automaton* for the LR(0) state; and then the method selects the correct the correct parsing action with the previously computed LR(1) lookahead information. For this,we establish the formalism for identifying LR(1) states by utilizing the properties of the *LR(1)-colored grammar* for the given grammar. On the basis of the formalism,we present a method for constructing the conflict-resolvin gautomata for LR(0) states which exhibit LALR(1) parsing conflicts.

Keywords: formal languages; context-free grammar, LR(*k*)/LALR(*k*) grammar, LR(*k*)/LALR(*k*) parsing, Optimization of LR(*k*) parsers.

LALR(1) 파싱충돌의 효율적인 해결에 관한 연구

이명준* · 박양수* · 정영필**
*전자계산학과 · **컴퓨터공학과

〈요 약〉

† Research supported in part by NON DIRECTED RESEARCH FUND, Korea Research Foundation 1992.

LR(1) 문법이지만 LALR(1) 문법이 아닌 문법을 파싱하기 위해서는 그 문법을 LALR(1) 문법으로 변형시키거나 LR(1) 파싱 테이블을 형성하여야 한다. 본 연구에서는 LALR(1) 문법이 아닌 LR(1) 문법의 LALR(1) 파싱에서 발생하는 충돌을 효율적으로 해결하는 새로운 방법을 개발하였다. 주어진 문장을 RALR(1) 파싱하는 과정에서 파싱충돌을 만나면 먼저 현재의 파싱 스택의 내용과 현재의 LR(0) 상태 (파싱 스택의 top)에 대해 미리 구성된 충돌해결자(conflict-resoving automaton)를 이용하여, 만일 LR(1) 파싱이 행해졌다면 도달했을 LR(1) 상태를 식별하고, 또한 미리 계산된 파싱충돌을 내포하고 있는 LR(0) 상태들의 LR(1) 상태에서의 Lookhead 정보를 이용하여 파서의 올바른 행동을 결정하므로써 파싱충돌을 해결한다. 이를 위하여, LR 파싱의 여러 특성을 잘 기술할 수 있도록 개발된 LR(k)-colored grammar의 성질을 이용하여 먼저 위와 같이 LR(1) 상태를 식별할 수 있는 방법을 성형적으로 기술한다. 이러한 성형론을 바탕으로 파싱충돌을 내포하고 있는 LR(0) 상태들에 대하여 현재의 파싱 스택을 입력으로 하는 유한자동기계(finite automaton)인 충돌해결자를 구성하는 방법을 개발하였다.

I. Introduction

Since the announcement of LR(k) grammars and their parsing [Knu65], they have filled a large role in handling the syntax of programming languages because the grammars generate all of the deterministitc context-free languages. However, since the size of the canonical LR(k) parsers is often unacceptably large, much work has been done to construct the parsers for the grammars, trying to reduce the size of the parsers as small as possible.

As a fruit of such work,LALR(k) parsing method has been widely accepted [Der69, Joh78, D&P82, PCC85] because the method uses LR(0) states and appropriate lookahead infomation. But,for those grammars which fail to be LALR(k), equivalent LALR(k)grammars should be constructed by some appropriate transformation from those LR(k) grammars [L&C91],leading often to grammars of unacceptably large size. In addition, various techniques for compressing LR(k) parsing tables have been developed. they merge the rows and columns of the parsing table [A&U72, AEH73, Tok81, S&82, LaL84] by using default reduction, eilminating reductions by single productions, making use of inessenial error entries, or the like. Also Pareger developed a method [Pag77a] which employs an LR(0) algorithm initially and then if the grammar is non-LALR(k) splits states so as to remove parsing conflicts;in addition, he devekoped a method[Pag77b] which merges *compatible states* with common cores during the con-strution of the proposed parsers.

In this paper, we persent another method for resoving parsing conflicts in the LALR(1) parser for an LR(1) grammar which is non-LALR(1).During parsing a given input string, the method uses LR(0) states and LALR(1) lookahead,and then if a parsing conflict occurs at a state, a *conflict-resolving automaton* for

that state,which is previously constructed by the LR(1) parser for the underlying garmmar, resolves the conflict by the use imformation in parsing stack and the perviously computed LR (1) lookahead imformation for the reducible items which exhibits LALR(1) parsing conflicts.

The organization of this paper is as follows. In Section 2, some fundamental definitions and properties are given. In Section 3, the formalism for identifying LR(1) states over an LR(0) automation is exploited with the LR(k)-colored grammar[Lee91, L&C91]. In Section 4, The method for constructing conflict-resolving automata is presented along with an example. A parsing example using the proposed method is also given. Final observations are stated in Section 5.

II. Preliminaries

We assume that the reader is familiar with the notations and conventions of the references [A&U72, Lee91, L&C91] concerning context-free grammars, LR(k) parsing, and LALR(k) parsing. In this section, we introduce some fundamental definitions and properties.

Property 2.1. *$GOTO(q_0, \gamma)$ is defined if and only if γ is a viable prefix of G.*

Property 2.2. *C_k is equivalent to Φ_k, the collection of the sets of valid LR(k) items for G,*

$$\Phi_k=\{q \mid q \text{ is the set of valid LR(k) items for some viable prefix of } G\}.$$

Moreover, a state q is the set of valid LR(k) items for a viable prfix γ iff $q=GOTO(q_0, gamma)$.

Definition 2.3. (*LR(k) lookahead set*) Let q be an LR(k) state in C_k. Then

$$LR_k(q, [A\rightarrow\alpha \,.\, \beta])=\{u \mid [A\rightarrow\alpha \,.\, \beta,u]\in q\}.$$

Property 2.4. (*LR(k) lookahead set*)

$$LR_k(q, [A\rightarrow\alpha \,.\, \beta])=\{PREF_k(y\$) \mid S \underset{rm}{\Rightarrow} \gamma\, Ay \Rightarrow \gamma\,\alpha\,\beta y,\ GOTO(q_0, \gamma\alpha)=q\}.$$

Property 2.5. (*LR(k) condition*) *G is LR(k) if and only if for all $q\in C_k$ and for all distinct cores $[A\rightarrow\alpha \,.\, B]$ and $[B\rightarrow\gamma \,.]$ of the items in q we have*

$$EFF_k(\beta) \oplus_k LR_k(q, [A\rightarrow\alpha \,.\, \beta]) \cap LR_k(q, [B\rightarrow\gamma \,.\,])=\varnothing$$

Definition 2.6. (*LALR(k) lookahead*) Let $p, q\in C_0$.

$$LALR_k(q, [A\rightarrow\alpha \,.\, \beta])=\{x \mid x\in FOLLOW_k(p,A),\ GOTO(p,\alpha)=q\},$$

where $FOLLOW_k(p,A)=\{PREF_k(y\$) \mid S \overset{*}{\underset{rm}{\Rightarrow}} \gamma Ay,\ GOTO(q_0, \gamma)=q\}$.

Definition 2.7. (*LALR(k) grammar*) G is said to be *LALR*(k) if for all $q \in C_0$ and for all distinct items $[A \rightarrow \alpha . \beta]$ and $[B \rightarrow \gamma .]$ in q we have

$$EFF_k(\beta) \oplus_k LALR_k(q, [A \rightarrow \alpha . \beta]) \; LALR_k(q, [B \rightarrow \gamma .]) = \varnothing.$$

Property 2.8. *There is a derivation in G such that*

$S' \overset{\cdot}{\Rightarrow}_{rm} \gamma_1 w_1 \overset{\cdot}{\Rightarrow}_{rm} \gamma_2 w_2 w_1 \overset{\cdot}{\Rightarrow} z$, and γ_1 and γ_2 are *viable prefixes of G*
if and only if there is a valid LR(k) parsing sequence over G such that
$(q_0, z\$, \varepsilon) \vdash^{\cdot} (\sigma_{\gamma 2}, w_2 w_1 \$, \Pi) \vdash^{\cdot} (\sigma_{\gamma 1}, w_1 \$, \Pi') \vdash^{\cdot} (\varepsilon, \$, \Pi'')$ for *some* $\Pi, \Pi', \Pi'' \in P^{\cdot}$.

III. Identifying LR(1) States over an LR(0) Automation with LR(0) /LR(1) Coloring

As a tool for identifying LR(1) states over a given LR(0) automaton, we introduce a special grammar called LR(k)-*colored grammar* ([Lee91], [L&C92]), and establish a formalism for the identification, utilizing the relationship between LR(1)-colored grammar and LR(0)-colored grammar.

LR(k)-colored Grammar

The LR(k)-colored grammar is constructed from the LR(k) machine on the ground that *GOTO* transitions on terminal symbols from each LR(k) state causes *shift* moves, and reducible items in the state causes *reduce* moves on the productions in the items, while *GOTO* transitions on nonterminal symbols causes state transitions as a part of *reduce* moves. For representing such moves of the LR (k) machine as grammar symbols, we transfigure each *GOTO* transition into a symbol of the introduced grammar, and also transfigure each reduction. As a result, we have a one-to-one correspondence between *shift* and/or *reduce* moves of the LR(k) machine, and those symbols that appear in the LR(k)-colored grammar ([Lee91], [L&C92]); furthermore, the LR(k)-colored grammar for an LR(k) grammar G is an *SLR(k)-cover* of G ([Lee91], [L&C91]).

Defining the LR(k)-colred grammar, we use new notations X^q and π^q to denote nonterminals of the grammar; they mean the transition on symbol X from state q and the reduction on π at state q, respectively.

Construction 3.1. Let a CFG $G=(N,\Sigma,P,S)$, and $LRM_k(G)=(C_k, GOTO, ACTION, q_0)$. The *LR (k)-colored grammar* for G is $\boldsymbol{G}=(\boldsymbol{N},\boldsymbol{\Sigma},\boldsymbol{P},\boldsymbol{S})$, where

(1) $\boldsymbol{N}=\{\boldsymbol{S}\} \; \{X^q \mid q \in C_k, [A \rightarrow \alpha . X\beta, u] \in q,$
$X \in V\} \cup \{\pi^q \mid q \in C_k, [A \rightarrow \alpha ., u] \in q,$
$A \neq S', \pi \text{ is } A \rightarrow \alpha \in P\}$,

The set of new vocabularies, $\boldsymbol{N} \cup \Sigma$, is denoted by $\boldsymbol{V}$. For notational convenience, we classify $\boldsymbol{N}$ into four disjoint sets, $\{S\}$, $\boldsymbol{N}_N$, $\boldsymbol{N}_\Sigma$, and $\boldsymbol{N}_P$, as follows:

$N_N=\{A^q \mid A\in N, A^q\in N\}$, $N_\Sigma=\{a^q \mid a\in\Sigma, a^q \in N\}$, $N_P=(\pi^q \mid \pi\in P, \pi^q\in N\}$; and the set $N_N \cup N_\Sigma$ is denoted by N_V.

(2) $P=\{S \rightarrow S^{q_0})$

$\cup \bigcup_{q\in C_k} \{A^q \rightarrow \theta(q,\alpha) \cdot \pi^q \mid A \neq S', [A\rightarrow . \alpha, u]\in q, \pi \text{ is } A\rightarrow\alpha, q' = GOTO(q,\alpha)\}$

$\cup \bigcup_{a^q\in N_\Sigma} \{a^q \rightarrow a\} \cup \bigcup_{\pi^q\in N_p} \{\pi^q \rightarrow \varepsilon\}$,

where θ is a function from $C_k \times V^*$ to N_V^* defined by

if X^q is in N_V (or equivalently $GOTO(q,X)$ is in C_k). We call θ *the LR(k)-coloring function for* G.

Properties of LR(k)-colored Grammars

We present basic properties of LR(k)-colored grammars developed by one of the author[Lee91, L&C91], which are useful for identify-ing LR(1) states over an LR(0) automation.

Definition 3.3. A fine homomorphism h: $V \rightarrow V \cup \{\varepsilon\}$ is

$$h(X)=\begin{cases} S', \text{ if } X=S \\ \varepsilon, \text{ if } X\in N_P \\ X, \text{ if } X=X^q\in N_V \\ a, \text{ if } X=a\in\Sigma. \end{cases}$$

Property 3.4.

(1) For an arbitrary LR(k) state q, $\theta(q,\alpha)$ θ $(GOTO(q,\alpha), \beta)=\theta(q, \alpha\beta)$.

(2) $\theta(q_0, \gamma)$ is defined if and only if γ is a viable prefix of G.

(3) Let $\gamma_1=\theta(q_0, \gamma_1)$ and $\gamma_2=\theta(q_0, \gamma_2)$. Then there is a derivation in G such that $S\Rightarrow_{rm}^{*} \gamma_1 w_1 \Rightarrow_{rm}^{*} \gamma_2 w_2 w_1 \Rightarrow^{*} z$ if and only if there is a derivation in G such that $S\Rightarrow_{rm}^{*} \gamma_1 w_1 \Rightarrow_{rm}^{*} \gamma_2 w_2 w_1 \Rightarrow^{*} z$.

(4) If there is a derivation in G such that $S \Rightarrow_{rm}^{*} \gamma X^q \alpha$, then $q=GOTO(q_0, h(\gamma))$.

A useful one-to-one correspondence between rightmost derivations in G and valid LR(k) parsing sequences over G can be established by the following three properties.

Property 3.5. *Let* $\gamma_1=\theta(q_0, \gamma_1)$ *and* $\gamma_2=\theta(q_0, \gamma_2)$. *Then there is a derivation in G such that*

$$S\Rightarrow_{rm}^{*} \gamma_1 w_1 \Rightarrow_{rm}^{*} \gamma_1 w_2 w_1 \Rightarrow^{*} z$$

if and only if there is an LR(k) parsing sequence over G such that

$(q_0, z\$, \varepsilon) \vdash^{*} (\sigma_{\gamma_2}, w_2 w_1\$, \Pi) \vdash^{*} (\sigma_{\gamma_1}, w_1\$, \Pi') \vdash^{*} (\varepsilon, \$, \Pi'')$ for *some* $\Pi,\Pi',\Pi''\in P^*$.

Property 3.6. *Let* a^q *be a nonterminal in* N_Σ. *Then there is a derivation in* G *such that*

$$S\Rightarrow_{rm}^{*} \gamma\ a^q w \Rightarrow^{*} z$$

if and only if there is a valid LR(k) parsing sequence such that

$(q_0, z\$, \varepsilon) \vdash^{*} (\sigma_\gamma, aw\$, \Pi) \vdash_{shift} (\sigma_{\gamma a}, w\$, \Pi) \vdash^{*} (\varepsilon, \$, \Pi')$

where $\gamma = \theta(q_0, \gamma)$, $top(\sigma_\gamma)=q$, and $\Pi,\Pi'\in P^*$.

Property 3.7. *Let* π^q *be a nonterminal in* N_P *with* π *being* $A\rightarrow\gamma_2$. *Then, there is a derivation in* G *such that*

$S \overset{*}{\Rightarrow}_{rm} \gamma\, a^q w \Rightarrow^{\cdot} z$

if and only if there is a valid LR(k) parsing sequence such that

$(q_0, z\$, \varepsilon) \vdash^{\cdot} (\sigma_{\gamma_1\gamma_2}, w\$, \Pi) \vdash_{\pi} (\sigma_{\gamma_1 A}, w\$, \Pi\pi) \vdash^{\cdot} (\varepsilon, \$, \Pi')$,

where $\gamma = \theta(q_0, \gamma_1\gamma_2)$, $top(\sigma_{\gamma_1\gamma_2})=q$, and $\Pi, \Pi' \in P^*$.

Property 3.8. *Let a description sentence of z be α string a such that $\alpha \Rightarrow^{\cdot} z$ and $\alpha \in L(G_d)$, where $G_d=(S \cup N_N, N_\Sigma \cup N_P, P_d, S)$ with*

$$P_d = P - (\{a^q \to a \mid a^q \in N_\Sigma\} \cup \{\pi^q \to \varepsilon \mid \pi^q \in N_P\}).$$

Then, there is a valid LR(k) parsing sequence of z if and only if there is a decription sentence of z. (In other words, the shift and/or reduce moves in an LR(k) parsing sequence can be described by the a^q and/or π^q symbols in a description sentence of z.)

Identifying LR(1) states over an LR(0) automaton

We define some new notions for capturing the relationship between LR(1) parsing sequences and LALR(1) parsing sequences. Since LALR (1) parsing uses LR(0) states, the relationship is presented eventually in terms of LR(1)-colored grammars and LR(0)-colored grammars.

Definition 3.9. A partial function *transym*:

$$Q \times Q \to V \text{ is}$$

$$transym(p, q) = \begin{cases} X, \text{ if } GOTO(p, X)=q \\ undefined, otherwise. \end{cases}$$

Definition 3.10. For a given string of LR(k) states $\rho=q_0q_1 \cdots q_n \in C_k^+$, we define a partial function *viable*: $C_k^+ \to V^*$ as follows:

$viable(\sigma)=X_0 X_1 \cdots X_{n-1}$,

where $X_i=transym(q_i, q_{i+1})$ for $1 \le i \le n-1$.

Obviously, $\rho=\sigma_\gamma$ iff $\gamma = viable(\rho)$. Note that function *viable* is always defined for the *stack strings*—the strings of LR(k) states contained in the parsing stack.

In what follows, the LR(1) machine for G is denoted by $LRM_1(G)=(C_1, GOTO_1, ACTION_1, q_{0:1})$ and the LR(0) machine for G is denoted by $LRM_0(G)=(C_0, GOTO, ACTION, q_0)$. The fundamental relationship between LR(1) parsing sequences and LR(0) parsing sequences is captured is captured by the following lemma.

Lemma 3.11. There is a valid LR(0) parsing sequence such that

$(q_0, z\$, \varepsilon) \vdash^{\cdot} (\rho, w\$, \Pi) \vdash^{\cdot} (\varepsilon, \$, \Pi')$

if and only if there is a valid LR(1) parsing sequence such that

$(q_{0:1}, z\$, \varepsilon) \vdash^{\cdot} (\rho', w\$, \Pi) \vdash^{\cdot} (\varepsilon, \$, \Pi')$

where $viable(\rho)=viable(\rho')$

Proof. From Property 2.8, we have: there is a valid LR(0) parsing sequence such that

$(q_0, z\$, \varepsilon) \vdash^{\cdot} (\rho, w\$, \Pi) \vdash^{\cdot} (\varepsilon, \$, \Pi')$

if and only if there is a derivation in G such that

$S \overset{*}{\Rightarrow}_{rm} \gamma w \Rightarrow^{\cdot} z$, and $\gamma = viable(\rho)$.

Again, From Property 2.8, we have: there is a derivation in G such that

$S' \overset{*}{\Rightarrow}_{rm} \gamma w \Rightarrow^{\cdot} z$, and $\gamma = viable(\rho)$.

if and only if there is a valid LR(1) parsing sequence such that

$(q_{0:1}, z\$, \varepsilon) \vdash^{\cdot} (\rho', w\$, \Pi) \vdash^{\cdot} (\varepsilon, \$, \Pi')$

Since $\rho' = \sigma_\rho$ i.e., $viable(\rho')=viable(\rho)$, the lemma holds. □.

Definition 3.12. For a given LR(0) stack string $\rho \in C_\sigma^+$ A function $LR(1)\text{-}state(\rho)= GOTO_1(q_{0:1}, viable(\rho))$.

According to lemma 3.11 and Definition 3. 12, we get the following theorem.

Theorem 3.13. If $(q_0, z\$, \varepsilon) \vdash^* (\rho, w\$, \Pi) \vdash^* (\varepsilon, \$, \Pi')$ is a valid LR(0) parsing sequence and $(q_{0:1}, z\$, \varepsilon) \vdash^* (\rho', w\$, \Pi) \vdash^* (\varepsilon, \$, \Pi')$ is the corresponding LR(1) parsing sequence under the context of Lemma 3.11, then $top(\rho')$=LR (1)-$state(\rho)$.

Definition 3.14. A fine homomorphism h_0: $C_1 \rightarrow C_0$ is

$h_0(q_{i:j})=q_i$, *where* $q_i=core(q_{i:j})$ $(=\{[A\rightarrow\alpha\,.\,\beta] | [A\rightarrow\alpha\,.\,\beta,u]\in q_{i:j}\}$.

Henceforth, $\boldsymbol{G}_1=(\boldsymbol{N}1,\Sigma,\boldsymbol{P}1,\ \boldsymbol{S}1)$ denotes the LR(1)-colored grammar for G and $\boldsymbol{G}_0=(\boldsymbol{N}0,\Sigma,\boldsymbol{P}0,\boldsymbol{S}0)$ denotes the LR(0)-colored grammar for G.

Definition 3.15. A fine homomorphism h_{g0}: $\boldsymbol{V}1\rightarrow\boldsymbol{V}0$ is

$$h_{g0}(\boldsymbol{X})=\begin{cases}\boldsymbol{S}0, \text{ if } \boldsymbol{X}=\boldsymbol{S}1\\ \pi^{h_0(q)}, \text{ if } \boldsymbol{X}=\pi^q\in\boldsymbol{N}1_P\\ X^{h_0(q)}, \text{ if } \boldsymbol{X}=X^q\in\boldsymbol{N}1_V\\ a, \text{ if } \boldsymbol{X}=a\in\Sigma.\end{cases}$$

In this, $\boldsymbol{V}1=\boldsymbol{N}1\cup\Sigma$ and $\boldsymbol{V}0=\boldsymbol{N}0\cup\Sigma$

Definition 3.16. Let $G=(N,\Sigma,P,S)$ be a CFG. We define a function LC: $N\rightarrow V^*$ by $LC(A)=\{\gamma | S\Rightarrow\ \gamma Aw, A\in N,\ \gamma\in V^*, w\in\Sigma^*\}$.

From Property 3.5, Theorem 3.13 and Definition 3.16, we get the following lemma.

Lemma 3.17. For a given LR(0) stack string ρ, let $\gamma=viable(\rho)$. Then, $LR(1)\text{-}state(\rho)$ is the LR (1) state $q_{i:j}$ if and only if there is a symbol $X^{q_{i:j}}$ in $\boldsymbol{G}_1$ such that $\theta(q_{0:1}, \gamma)\ LC_1(X^{q_{i:j}})$.

The following lemma is easily derived from Property 3.5, Lemma 3.11 and Definition 3.15.

Lemma 3.18. *Let* $\gamma_{1:0}=\theta(q_0, \gamma_1)$, $\gamma_{2:0}=\theta(q_0, \gamma_2)$, $\gamma_{1:1}=\theta(q_{0:1}, \gamma_1)$ *and* $\gamma_{2:1}=\theta(q_{0:1}, \gamma_2)$.
Then obviously

$\gamma_{1:0}=h_{g0}(\gamma_{1:1})$, and $\gamma_{2:0}=h_{g0}(\gamma_{2:1})$;

in addition, there is a derivation in $\boldsymbol{G}_1$ *such that*

$\boldsymbol{S}1\Rightarrow^*_{rm}\gamma_{1:1}w_1\Rightarrow^*_{rm}\gamma_{2:1}w_2w_1\Rightarrow^* z$

if and only if there is a derivation in $\boldsymbol{G}_0$ *such that*

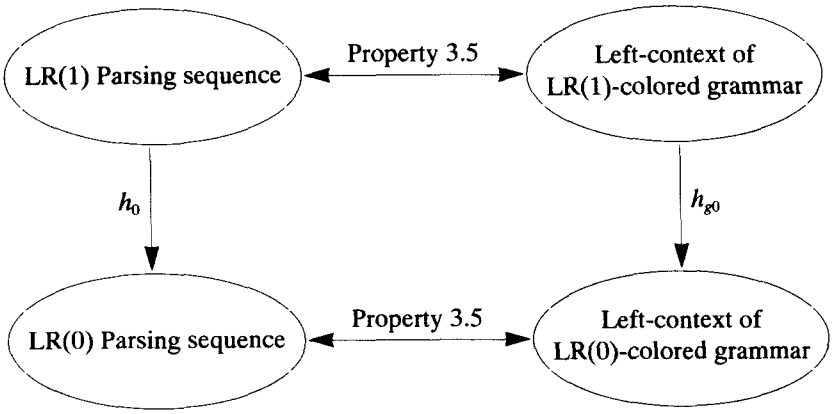
$\boldsymbol{S}0\Rightarrow^*_{rm}\gamma_{1:0}w_1\Rightarrow^*_{rm}\gamma_{2:0}w_2w_1\Rightarrow^* z$

Definition 3.19. For a string $\gamma = X_0^{q_0} X_1^{q_1}\cdots X_n^{q_n} \in \boldsymbol{N}$, $state\text{-}color(\gamma)=q_0q_1\cdots q_n$.

According to Lemma 3.17, 3.18 and Definition 3.19, we have the following theorem.

Theorem 3.20. For a given LR(0) stack string ρ, let $\gamma=viable(\rho)$. Then, $LR(1)\text{-}state(\rho)$ is the LR(1) state $q_{i:j}$ if and only if there is a symbol $X^{q_{i:j}}$ in $\boldsymbol{G}_1$ such that $\rho\in state\text{-}color(h_{g0}(LC_1(X^{q_{i:}$

By virtue of Theorem 3.20, we can identify the LR(1) state corresponding to a given LR(0) stack string with the information which is statically computed from the LR(1)-colored grammar or LR(1) automation. The relationship found is depicted by the following figure.



^{j}))), or equivalently $\rho \in h_0(state\text{-}color(LC_1(X^{q_{i:j}}))$.

IV. Resolving LALR(1) Parsing Conflicts

Assume that the underlying grammar G is LR(1) and not LALR(1). Then the LALR(1) machine for G exhibits reduce-reduce parsing conflicts. In this section, we develop a method to resolve such parsing conflicts in accordance with Theorem 3.20.

Conflict-Resolving Automata

We begin by presenting a function from an LR(0) states to the set of LR(1) states with the same core:

Definition 4.1. *LR(1)-color*: $C_0 \rightarrow 2^{C_1}$ is defined by

$$LR(1)\text{-}color(q_i)=\{q_{i:j} \mid q_i=Core(q_{i:j})\}$$

Let q be an LR(0) state with LALR(1) parsing conflicts, and ρ be an LR(0) stack string whose top symbol is q. Then parsing conflicts might be arisen at the state q while LALR(1) parsing proceeds. For resolving those conflicts, according to Theorem 3.20, we need $h_0(state\text{-}colr(LC_1(X^{q_{i:j}})))$ for all $X^{q_{i:j}} \in N_V$ such that $q_{i:j} \in Core(q_{i:j})$. The information can be easily computed by the following Property

Property 4.2.

$$\bigcup_{X \in V1.\ state\text{-}color(X)=q_{i:j}} state\text{-}color(LC_1(X))=path(q_{0:1}, q_{i:j}),$$

where path($q_{0:1}$, $q_{i:j}$) is the set of sequences of states spelled out from $q_{0:1}$ to $q_{i:j}$ in the underlying LR(1) automation.

Theorem 4.3. *Let $q_{i:j}$ and $q_{i:k}$ be two distince LR(1) states. Then,*

$h_0(path(q_{0:1}, q_{i:j})) \cap h_0(path(q_{0:1}, q_{i:j}))=\varnothing$.

Proof. Assume that there is a LR(0) stack string ρ which is contained both in $h_0(path(q_{0:1}, q_{i:j}))$ and in $h_0(path(q_{0:1}, q_{i:k}))$. Let $\gamma = viable(\rho)$. Then, $q_{i:j}$ is $q_{i:k}$ because the underlying LR(1) automation is a deterministic finite automation which accepts viable prefixes and $q_{i:j}=GOTO(q_{0:1}, \rho)=q_{i:k}$. □

In virtue of Property 4.2 and Theorem 4.3, we are able to construct an automation which resolves LALR(1) parsing conflicts by idenifying LR(1) states. First, we construct an automaton which accepts the reverse of LR(1) stack strings.

Construction 4.4. Let an LR(1) automation for a CFG G be given, and q_i be an LR(0) state which exhibits LALR(1) parsing conflicts. Then, for each LR(1) state $q_{i:j} \in LR(1)\text{-}color(q_i)$, $RM(q_{i:j})$ is constructed as follows:

n :=0
for each $q_{i:j} \in C_1$ do $s[q_{i:j}]:= \gamma$ endfor
for each transition from $q'_{i:j}$ to $q''_{i:j}$ in the LR(1) automaton do
    if $s[q''_{i:j}]= \gamma$ then
        $n := n + 1$; $s[q''_{i:j}]:=t_n$; $s^{-1}[t_n]:=q''_{i:j}$
    endif
    if $s[q'_{i:j}]= \gamma$ then
        $n := n + 1$; $s[q'_{i:j}]:=t_n$; $s^{-1}[t_n]:=q'_{i:j}$
        $\delta(s[q''_{i:j}], q'_{i:j}):=s[q''_{i:j}]$
    endif
endfor

4.2 and Theorem 4.3.

Let $M(q_{i:j})=(\{t_k \mid 1 \le k \le n\}, C_1, \delta, s[q_{i:j}], \varnothing)$ be the finite automation constructed above. Then, $RM(q_{i:j})$ is the automation resulted from the removal of all inaccessible states in $M(q_{i:j})$.

The conflict-resolving automation for each LR(0) state with LALR(1) parsing conflicts, is constructed by utilizing the above automata:

Construction 4.5. Let q_i be an LR(0) state with LALR(1) parsing conflicts. Then, *CRM*

for each $q_i \in C_0$ do $s_0[q_i] := \gamma$ endfor
$n := 0$; $s_0[q_i] := r_0$; $d[r_0] := LR(1)\text{-}color(q_i)$;
for each $q_{i:j} \in LR(1)\text{-}color(q_i)$ do
for each transition from t to t' on $q'_{i:j}$ over $RM(q_{i:j})$ do
if $s_0[h_0(s^{-1}[t])] = \lambda$ then
$n := n + 1$ $s_0[h_0(s^{-1}[t])] := r_n$; $d[r_n] := \varnothing$
endif
if $s_0[h_0(s^{-1}[t'])] = \lambda$ then
$n := n + 1$ $s_0[h_0(s^{-1}[t'])] := r_n$; $d[r_n] := \varnothing$
endif
$\delta(s_0[h_0(s^{-1}[t])], h_0(q'_{i:j})) := s_0[h_0(s^{-1}[t'])]$
$d[s_0[h_0(s^{-1}[t'])]] := d[s_0[h_0(s^{-1}[t'])]] \{q_{i:j}\}$
endfor
endfor
for $0 \le k \le n$ do
if $|d[r_k]| \le 1$ then delete all the transitions from r_k endif
endfor

Let $M_0(q_i)=(\{r_k \mid 1 \le k \le n\}, C_0, \delta, s_0[q_i], F)$ be the finite automation constructed above. In this, $F=\{r_k \mid |d[r_k]|=1 \text{ for } 1 \le k \le n\}$. Then, $CRM(q_i)$ is the automation resulted from the removal of all inaccessible states in $M_0(q_i)$.

Example

Consider the following LR(1) grammar $G=(\{S,A,B\}, \{a,b,c,d,e,\$\}, P, S)$, where P is composed of the following productions:

$\pi_1: S \rightarrow aAdS$, $\pi_2: S \rightarrow \varepsilon$, $\pi_3: S \rightarrow bBd$,
$\pi_4: S \rightarrow aBe$, $\pi_5: S \rightarrow bAe$, $\pi_6: A \rightarrow c$,
$\pi_7: B \rightarrow c$.

The LR(1) automaton for G is presented in Figure 4.1. By Construction 3.1, the LR(1)-colored grammar for G is $\boldsymbol{G}=(\boldsymbol{N},\Sigma,\boldsymbol{P},\boldsymbol{S})$,

(1) $\boldsymbol{N} = \{S\} \cup N_N \cup N_\Sigma \cup N_P$, where
$\boldsymbol{N_N} = \{S^{q_0}, S^{q_7}, A^{q_2}, A^{q_3}, B^{q_2}, B^{q_3}\}$
$\boldsymbol{N_\Sigma} = \{a^{q_0}, a^{q_7}, b^{q_0}, b^{q_7}, c^{q_2}, c^{q_3}, d^{q_6}, d^{q_{13}}, e^{q_4}, e^{q_{11}}\}$
$\boldsymbol{N_P} = \{\pi_1^{q_8}, \pi_2^{q_0}, \pi_2^{q_7}, \pi_3^{q_{14}}, \pi_4^{q_5}, \pi_5^{q_{12}}, \pi_6^{q_9}, \pi_6^{q_{10}}, \pi_7^{q_9}, \pi_7^{q_{10}}\}$;

(2) $\boldsymbol{P}$ is composed of the following productions:

$S \rightarrow S^{q_0}$,

$S^{q_0} \rightarrow \pi_2^{q_0} \mid a^{q_0} A^{q_2} d^{q_6} S^{q_7} \pi_1^{q_8} \mid a^{q_0} B^{q_2} e^{q_4} \pi_4^{q_5} \mid b^{q_0} B^{q_3} d^{q_{13}} \pi_3^{q_{14}} \mid b^{q_0} A^{q_3} e^{q_{11}} \pi_5^{q_{12}}$.

$S^{q_7} \rightarrow \pi_2^{q_7} \mid a^{q_7} A^{q_2} d^{q_6} S^{q_7} \pi_1^{q_8} \mid a^{q_7} B^{q_2} e^{q_4} \pi_4^{q_5} \mid b^{q_7} B^{q_3} d^{q_{13}} \pi_3^{q_{14}} \mid b^{q_7} A^{q_3} e^{q_{11}} \pi_5^{q_{12}}$.

$A^{q_2} \rightarrow c^{q_2} \pi_6^{q_{10}}$, $A^{q_3} \rightarrow c^{q_3} \pi_6^{q_9}$, $B^{q_2} \rightarrow c^{q_2} \pi_7^{q_{10}}$, $B^{q_3} \rightarrow c^{q_3} \pi_7^{q_9}$

$X^q \rightarrow X$ for all $X^q \in \boldsymbol{N}_\Sigma$ $\pi^q \rightarrow \varepsilon$ for all $\pi^q \in \boldsymbol{N}_P$

(q_i), called by the conflict-resolving automation for q_i is constucted as follows:

In case of LALR(1) parsing, the states q_9 and q_{10} is merged into a single state, say $q_{9,10}$, because they have a common core. And the LR (0) state $q_{9,10}$ exhibits parsing conflicts for lookahead d and e. Applying Construction 4.4, we get $RM(q_9)$ and $RM(q_{10})$; and we get CRM $(q_{9,10})$ by Construction 4.5. Those are exposed in Figure 4.2.

We close this section by presenting the following parsing example which uses the developed method for resolving LALR(1) parsing conflicts.

Example 4.6. For the grammar G in Example 3.2 and the sentence $acdbcd$ of G, the developed LALR(1) parser, which has the conflict-resolving automation $CRM(q_{9,10})$ and the LR(1) lookahead information for the resucible items contained in q_9 and q_{10}, works as follows.

$(q_0, acdbcd\$, \varepsilon)$	$\vdash_{shift}$	$(q_0q_2, cdbcd\$, \varepsilon)$	
	$\vdash_{shift}$	$(q_0q_2q_{9,10}, dbcd\$, \varepsilon)$	reduce-reduce conflict(1)
	$\vdash_{\pi 6}$	$(q_0q_2q_6, dbcd\$, \pi_6)$	
	$\vdash_{shif}$	$(q_0q_2q_6q_7, bcd\$, \pi_6)$	
	$\vdash_{shif}$	$(q_0q_2q_6q_7q_3, cd\$, \pi_6)$	
	$\vdash_{shif}$	$(q_0q_2q_6q_7q_3q_{9,10}, d\$, \pi_6)$	reduce-reduce conflict(2)
	$\vdash_{\pi 7}$	$(q_0q_2q_6q_7q_3q_{13}, d\$, \pi_6\pi_7)$	
	$\vdash_{shif}$	$(q_0q_2q_6q_7q_3q_{13}q_{14}, \$, \pi_6\pi_7)$	
	$\vdash_{\pi 3}$	$(q_0q_2q_6q_7q_8, \$, \pi_6\pi_7\pi_3)$	
	$\vdash_{\pi 1}$	$(q_0q_1, \$, \pi_6\pi_7\pi_3\pi_1)$	
	$\vdash_{accept}$	$(\varepsilon, \$, \pi_6\pi_7\pi_3\pi_1)$	

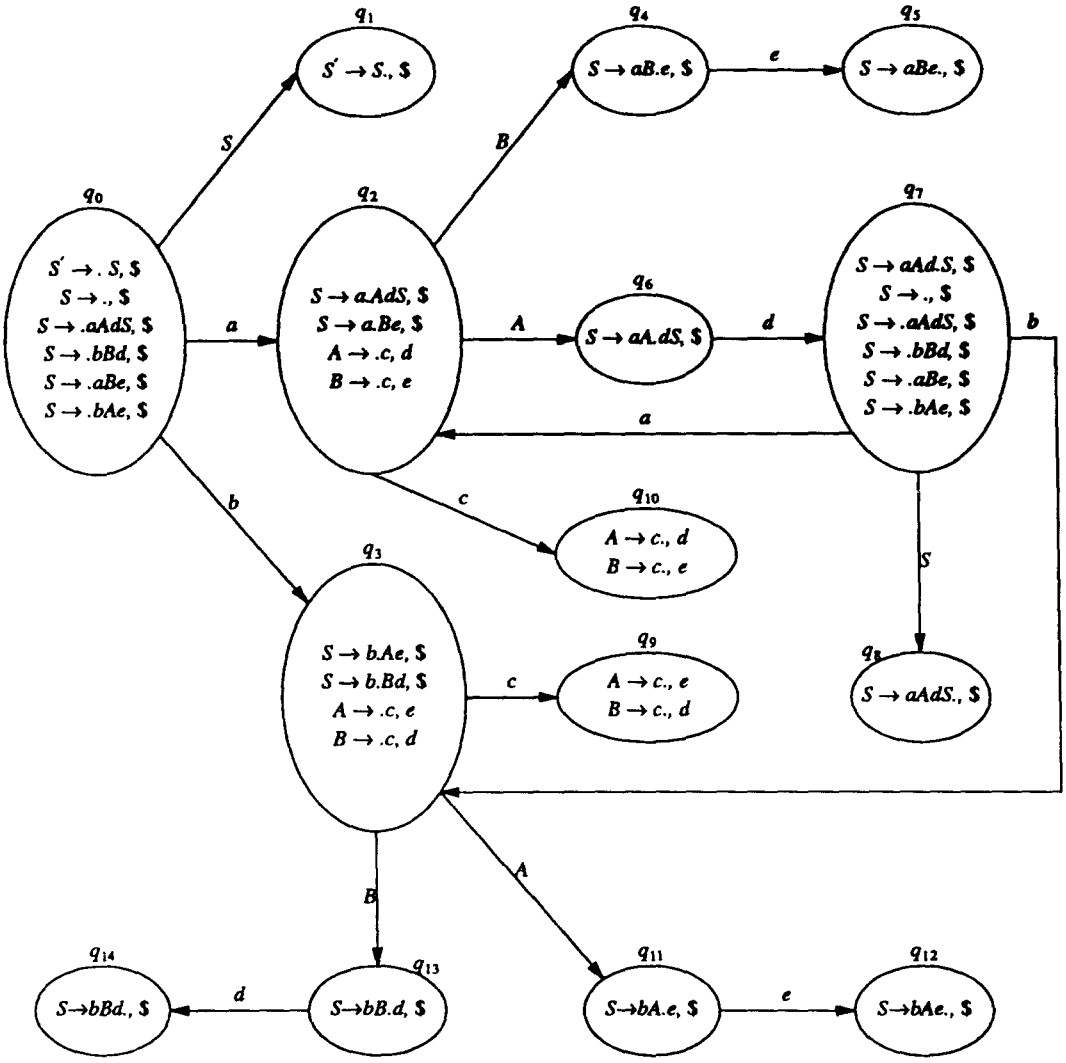


Figure 4.1 LR(1) automation for *G*

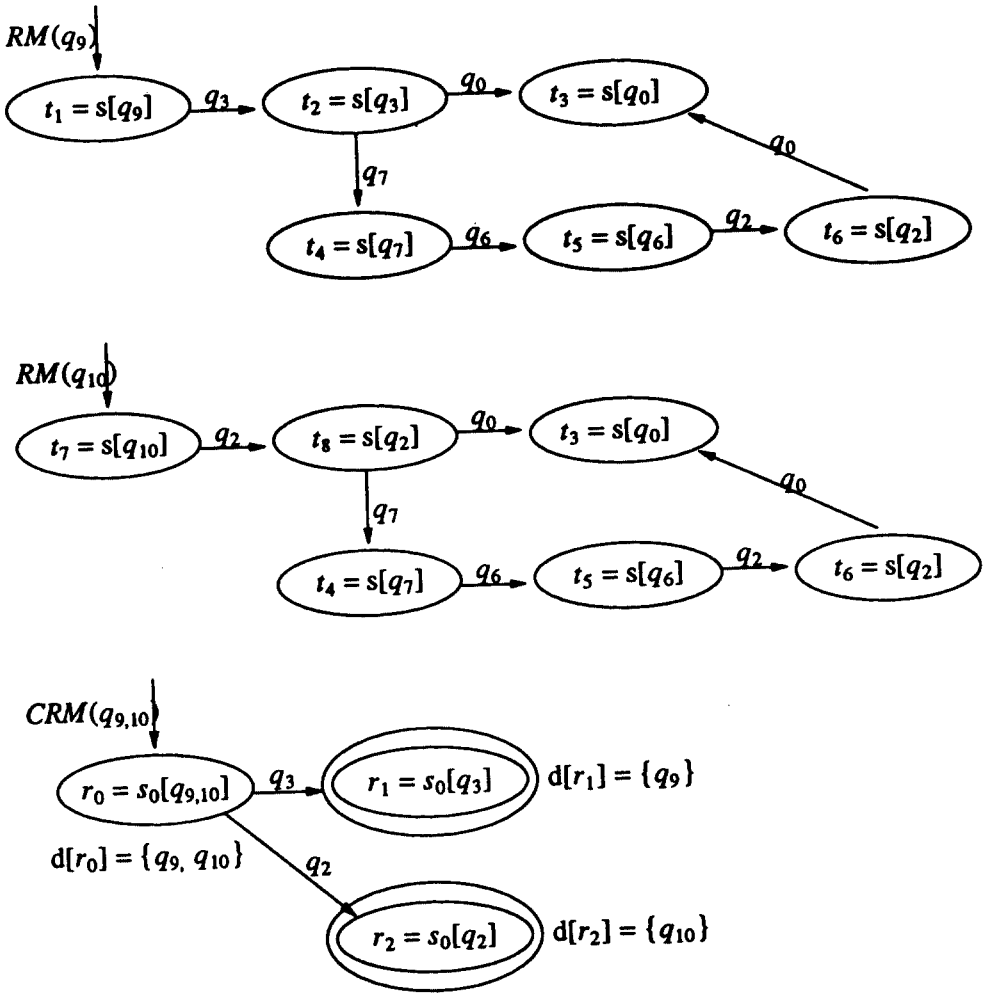


Figure 4.2

At the position reduce-reduce conflict (1), $CRM(q_{9,10})$ starts with the LR(0) stack string q_2q_0, and stops at state r_2. Since $d[r_2]=\{q_{10}\}$ and the lookahead d is in $LR_1(q_{10}, [A\rightarrow c.])$, the parsing proceeds as above. At the position reduce-reduce conflict (2), $CRM(q_{9,10})$ starts with the LR(0) stack string $q_3q_7q_6q_2q_0$, and stops at state r_1. Since $d[r_1]=\{q_9\}$ and the lookahead d is in $LR_1(q_9, [B\rightarrow c.])$, the parsing proceeds as above.

V. Conclusion

We have presented a new method for resolving parsing conflicts which may occur while the LALR(1) parser for a given LR(1) and non-LALR(1) grammar processes an input string. When a parsing conflict occurs at an LR (0) state, the method identifies the LR(1) state, which would be the top of the parsing stack if LR(1) parsing was performed, by the use of the current content of the parsing stack and a *conflict-resolving automaton* for the LR(0) state; and then the method selects the correct parsing action with the previously computed LR(1) lookahead information. For this, we have established the formalism for identifying LR(1) states by utilizing the properties of the *LR(1)-colored grammar* for the given LR(1) grammar. On the basis of the formalism, we have constructed the conflict-resolving automation for an LR(0) state which exhibits LALR(1) parsing conflicts with the information extracted from the LR(1) automaton for the grammar. It would be noted that the method can be easily applied the LR(or LALR) parser generating systems. It would be a meaningful continuation of this work to develop an efficient algorithm for constructing the conflict-resolving automaton.

REFERENCES

[A&U72] Aho, A. V., and Ullman, J. D., "The theory of parsing, translation and compiling," Vol. 1-2. Englewood Cliffs, N.J.: Prentice-Hall 1972, 1973.

[A&U72] Aho, A. V., Ullman, J.D., "Optimization of LR(k) parsers," *J. Computer and System Sciences* 6, 573-602, 1972.

[AEH73] Anderson, T., Eve, J., Horning, J.J. "Efficient LR(1) parsers," *Acta Informatica* 2, 12-39, 1973.

[Der69] DeRemer, F.L. "Practical Translator for LR(k) Languages," Ph.D. dissertion, MIT, Cambridge, Mass, 1969.

[D&P82] DeRemer, F.L. and Pennello, T.J. "Efficient computation of LALR(1) lookahead sets," *TOPLAS* 4:4, 615-649, 1982.

[Joh78] Johnson, S.C. "Yacc-yet another compiler compiler," Computing Science Technical Report 32, AT&T Laboratories, Murray Hill, N.J, 1978.

[Knu65] Knuth, D.E. "On the translation of languages from left to right," *Information and Control* 8:6, 607-639, 1965.

[LaL84] LaLonde, W.R., "Comments on Soisalon-Soininen' s "Inessential error entries"," *ACM Trans. on Prog. Lang. and Syst.* 3: 168-207, 1984.

[Lee91] Lee, M.J. "LR(k)-Colored Grammar and Its Application," Ph.D. dissertation, Korea Advanced Institute of Science and Technology, 1991.

[L&C91] Lee, M. J., and Choe, K. M., "SLR (k) covering for LR(k) grammars," *Information Processing Letters* 37:6, 337-347, 1991.

[Pag77a] Pager, D., "The lane-tracing algorithm for constructiong LR(k) parsers and ways of enhancing its efficiency," *Information Science* 12: 19-42, 1977.

[Pag77b] Pager, D., "A practical general method for constructing LR(k) parsers," *Acta Informatica* 7: 249-268, 1977.

[PCC85] Park, J.C.H., Choe, K.M. and Chang, C.H "A new analysis of LALR formalism," *ACM Trans. on Prog. Lang. and Syst.* 7: 159-175, 1985.

[S&S82] Soisalon-Soininen, E., "Inessential error entries and their use in LR parser optimization," *ACM Trans. on Prog. Lang. and Syst.* 4: 179-195, 1982.

[Tok81] Tokuda, T., "Eliminating unit reductions from LR(k) parsers using minimum contexts," *Acta Informatica* 15, 447-470, 1981.