# LR(k)구문분석기의 국부적 최적축소
## (Locally Optimal Reduction of LR(k) Parsers)

박 우 전 † 이 명 준 †† 최 광 무 †††
(Woo-Jun Park) (Myung-Joon Lee) (Kwang-Moo Choe)

**요 약** 정축소 (*well-defined reduction*)와 국부적최적축소 (*locally optimal reduction*)라는 새로운 형식론을 도입하여 주어진 LR(k)구문분석기의 상태들을 정적으로 병합하는 문제를 다룬다. 또 정축소와 국부적 최적축소를 계산하는 알고리즘을 제안하고, LR(k)상태들을 생성해가며 병합하는 동적인 방법과의 차이점에 대해 논한다 위에서 '국부적' 이라함은 현재의 코아블록에서의 최적병합이 반드시 다른 코아블록에서의 최적병합을 보장하지 않음을 의미한다.

***Abstract*** The problem reducing the number of states in a given LR(*k*) parser is treated from the standpoint of *static merging*, introducing a well-defined reduction and a locally optimal reduction of the parser. In addition, algorithms to compute a *well-defined reduction* and a *locally optimal reduction* of an LR(*k*) parser are presented. The word 'locally' here means that optimal merging in the current core block does not always guarantee optimal merging in another core bolck. And the differences between the proposed method and related works is discussed.

## 1. 서 론

LR(*k*)문법의 발견[5]이래 파싱테이블의 크기를 가능한한 작게해주는 구문분석기 구축에 대한 많은 연구가 행해졌다. 그와같은 연구의 결과로서 SLR(*k*)/LALR(*k*) 구문분석[2,3]은 LR(0)상태와 적절한 룩어헤드 정보를 이용하는 기법으로서 보편적인 기법으로 인식되었다 그러나 주어진 LR(k)문법이지만 LALR(*k*)가 아닌경우에는 주어진 문법에 대한 변환을 행하여 동등한 LALR(*k*) 문법을 얻어야 한다. 이와같은 변환은 많은 계산을 요하며 크기가 너무커서 사용할수 없는 문법이 되게하는 경우기 많다[6]

Pager[7]는 호환상태 (compatible states)의 개념을 도입하고 같은코아를 가진 호환상태들을 병합하는 코아제한적(core-restricted)방법을 제안했으며, 그 방법에서는 상태들을 동적으로, 즉 LR(*k*)파싱 테이블을 모두 구축 후가 아닌 생성해가는 시점에서 병합한다. 호환상태들의 개념은 병합해도 아무런 파싱충돌을 일으키지 않는 상태들을 의미하며 Heilbrunner의 아이템문법 (item grammars)과 파싱오토마타[4]에 의해 훌륭하게 재 고찰된 바 있다.

본 논문에서는 호환상태들의 개념을 전체 LR(*k*)파싱 테이블을 구축한 후에 LR(*k*)상태들을 병합하는 즉 정적 병합 입장에서 다시 다룬다. 새로운 호환상태개념을 이용하여, 동일코아를 가진 상태들을 어떤 충돌도 일으키지않게 병합하는 새로운 형식론을 개발하고 새로운 개념인 LR(*k*)구문분석기에 대한 정축소 (*WDR*, well-defined reduction)를 도입한다[9]. 또한 최적축소에 대한 근사치로서 하나의 유용한 코아제한적인 방법인 LR(*k*)구문분석기의 국부적최적축소 (*LOR*, locally optimal reduction) 및 *LOR*계산을 위한 알고리즘을 제안하고, *LOR*과 Pager[7] 및 Heilbrunner[4]의 연구를 비교 분석한다.

본 논문의 구성은 다음과 같다. 제2절에서 LR(*k*)구문분석기와 집합론에 관한 관련표기법 및 정의들을 살펴보고 제3절에서 LR(*k*)상태 병합을 위한 새로운 관계를 제시하며 제4절에서는 LR(*k*)구문분석기의 정축소와 정축소 및 그와 관련된 LR(*k*)-based 구문분석기 계산을 위

· 본 연구는 한국학술진흥재단 '94 자유공모과제 연구비 지원에 의한 것임
† 종신회원 한남대학교 전자계산학과 교수
†† 종신회원 울산대학교 전자계산학과 교수
††† 종신회원 한국과학기술원 전산학과 교수
논문접수 1996년 3월 15일
심사완료 1996년 5월 30일

한 알고리즘을 개발한다[9]. 또한 LR($k$)구문분석기의 국부적최적축소와 최적축소에 관해 논하고 국부적최적축소 계산을 위한 알고리즘 및 다른 병합방법과의 차이점에 대하여 설명한다.

## 2. 표기법 및 정의

LR($k$)구문분석기에 관한 참고문헌[1]의 표기법이 독자들에게 친숙하다고 본 논문에서는 가정한다. 특히 다음과 같은 개념들을 자주 사용하기로 한다. 즉 관계 $\Rightarrow_{rm}$ (우측유도), $FIRST_k$, $EFF_k$, LR($k$)-*item*, *lookahead*, 및 함수 *closure*가 그렇다. 아래에서 Aho와 Ullman의 몇 가지 정의와 집합론[11]에 관한 사항을 필요하면 변형된 형태로 살펴본다. 문맥자유문법(Context-Free Grammar)은 $G=(N, \Sigma, P, S)$로 표기하며, $N$은 넌터미날들의 유한집합, $\Sigma$는 터미날들의 유한집합, $N \cap \Sigma = \emptyset$, $P$는 $N \times V^*$의 유한부분집합이며 여기서 $V$(어휘)는 $N \cup \Sigma$이고 $P$의 각원 $(A, \alpha)$를 생성규칙이라하며 $A \rightarrow \alpha$로 표기한다. 또 $S$는 시작기호이다. LR($k$)파싱의 서술에서의 편의상 문법 $G$는 확장되었다고 가정한다. 즉 $P$는 특별한 생성규칙 $S' \rightarrow S$를 가지며 $S'$는 다른생성규칙에 출현하지 않는다는 의미이다. 또한 $\Sigma$는 다른 어떤 생성규칙에도 나오지 않는 특별한 마침기호 "$"를 포함한다고 가정한다. LR($k$)-item $[A \rightarrow \alpha \cdot \beta, u]$는 $\alpha \neq \varepsilon$이거나 $A = S'$ 이면 kernel item[8]이라 한다 $q$가 아이템집합일때

$$kernel(q)=\{I \in q \mid I \text{ is a kernel item}\},$$

$$semikernel(q)=kernel(q) \cup \{A \rightarrow \cdot \mid A \rightarrow \cdot \in q\}$$

와 같이 표기한다. 한 상태에서 점이 찍힌 생성규칙 $A \rightarrow \alpha \cdot \beta$를 가진 모든 LR($k$)아이템들의 집합을 $[A \rightarrow \alpha \cdot \beta, U]$로 표기하고 여기서 $U$는 해당아이템들의 lookahead들의 집합이다 그런 아이템들의 집합을 주어진 상태의 아이템그룹이라 한다.

**정의 2.1** (LR 오토마톤[1,4,6]) $G$의 LR($k$)오토마톤은

$$M_k(G)=(C_k, V, \delta_k, q_{0\,k}, \emptyset)$$

와 같이 표기하며 위에서 $C_k$는 $G$의 LR($k$)아이템집합들의 정규모음으로서

$$C_k = {}_s\{q_{0\,k}\} \cup \{\delta_k(q, X) \mid q \in C_k, X \in V\}$$

(여기서 $q_{0\,k} = closure(\{S' \rightarrow \cdot S, \$\})$)이며

$$\delta_k(q, X) = closure(\{[A \rightarrow \alpha X \cdot \beta, u] \mid [A \rightarrow \alpha \cdot X \beta, u] \in q\})$$ 임)

와 같이 재귀적으로 (recursively) 정의된다. 표기법 "$Q = {}_s f(Q)$" 는 $Q$가 조건 $Q = f(Q)$를 만족하는 최소의 집합임을 의미한다. $C_k$의 원소를 문법 $G$의 LR($k$)상태라고 한다. 함수 $\delta_k$ 의 정의역은 아래와 같이 $C_k \times V^*$ 또는 $2^{V^*} \times V^*$ 로 확장된다

$$\delta_k(q, \varepsilon) = q$$
$$\text{and } \delta_k(q, X\gamma) = \delta_k(\delta_k(q, X), \gamma)$$
$$\text{or } \delta_k(Q, \alpha) = \{\delta_k(q, \alpha) \mid q \in Q\}.$$

문법 $G$에 대한 LR(0)오토마톤을 $M_0(G)=(C_0, V, \delta_0, q_{0\,0}, \emptyset)$ 로 표기한다. 함수 *core*는 LR($k$)아이템들의 집합으로부터 LR(0) 아이템들의 집합으로의 사상(mapping)으로서

$core(q) = \{[A \rightarrow \alpha \cdot \beta] \mid [A \rightarrow \alpha \cdot \beta, u] \in q\}$에의해 정의된다. 문법 $G$에대한 (코아제한적인) 파싱오토마톤 $M(G)=(Q, V, \delta, q_0, \emptyset)$ 은 다음조건들을 만족할 경우 LR($k$)-*based*라고 한다.

조건 1) $Q$내의 각 상태는 동일코아를 가진 LR($k$)상태들의 집합이다

2) $\delta : Q \times V \rightarrow Q$는 $core(\delta(q, X)) = \delta_0(core(q), X), q \in Q$를 만족하는 상태 천이함수로서 함수 *core*의 정의역은 $core(q) = \{I \in core(q) \mid q \in q\}$에의해 $2^Q$로 확장된다.

3) $q_0 = \{q_{0\,k}\}$로서 $M$의 초기상태이다.

명백히 $M(G)$는 correct prefix성질을 가진다.

**정의 2.2** [11] (집합의 파티션과 커버링) 주어진 집합 $\mathcal{S}$에서 $Q=\{T_1, T_2, \ldots, T_m\}$ (단 각$T_i$, $i=1, \cdots, m$, 는 $\mathcal{S}$의 부분집합임) 는 $\bigcup_{i=1}^{m} T_i = \mathcal{S}$ 이면, 집합 Q는 $\mathcal{S}$의 커버링이다 또 만일 $\mathcal{S}$의 부분집합들인 Q의 원소들이 서로 공통부분이 없을 경우 $Q$를 $\mathcal{S}$의 파티션이라 부르고 집합 $T_1, T_2, \ldots T_m$을 그 파티션의 블록 (*block*)들이라 한다.

**정의 2.3** (코아동치 파티션) $M_k(G)$의 $C_k$에 대한 코아동치파티션(core-equivalence partition)은 $C_k$에서 동일코아 $q_i$를 가지는 상태들의 집합을 $Bq_i$ $(q_i \in C_0)$로 표기할 때 (즉 $B_{qi} = \{p \mid core(p) = q_i, p \in C_k\}$) $\{B_{q1}, B_{q2}, \ldots, B_{q\,m}\}$이다. 각 $B_{qi}$를 이 파티션의 관련된 LR(0)상태 $q_i$에 대한 코아블록이라고 한다.

## 3. 호환 LR($k$)상태

주어진 LR($k$)구문분석기의 크기를 줄이기 위해서 같은

코아블록내의 일련의 LR($k$)상태들은 병합되어 한 개의 상태로 될 수 있다. 그러나 그들 상태들의 병합이 파싱충돌을 일으킬 수도 있다. LR($k$)상태들 $p$와 $q$를 병합해도 아무런 파싱충돌을 일으키지 않을 때 상태 $p$와 $q$는 서로 호환(*compatible*)이라고 한다. 이 절에서는 서로 호환인 LR($k$)상태들을 찾게해주는 새로운 관계를 정의한다. 이 절에서 $X$는 $V$에 속하는 기호이고 $\delta$는 정의 2.1의 $\delta_k$를 의미한다 먼저 합집합을 구해도 아무런 파싱충돌을 일으키지않는 LR($k$)아이템들의 집합들간에 성립하는 새로운 관계 $C_t$를 도입한다

**정의 3.1** (일시적 호환) $p$와 $q$를 LR($k$) 아이템들의 집합이라고 하자. $core(p)=core(q)$이고 모든 아이템그룹의 쌍, 즉
$[A\rightarrow\alpha\cdot\beta, U]$ ,$[B\rightarrow\gamma\cdot, W]\subseteq p$,
$[A\rightarrow\alpha\cdot\beta, U']$ ,$[B\rightarrow\gamma\cdot, W]\subseteq q$, $A\rightarrow\alpha\cdot\beta\neq B\rightarrow\gamma\cdot$
에 대해서 $W\cap EFF_k(\beta U)=\emptyset$ and $W\cap EFF_k(\beta U)=\emptyset$ 이 성립하면 $p\ C_t\ q$ 라고 한다.

$p\ C_t\ q$이면 $p\cup q$내에 아무런 shift-reduce 또는 reduce-reduce 충돌이 생기지 않는다는 것은 명백하다. 관계 $C_t$를 사용하여 새로운 호환관계를 아래와 같이 재귀적으로 정의한다.

**정의 3.2** (호환) $p$와 $q$가 LR($k$)상태일 때
$p\ C\ q$ iff $p\ C_t\ q\wedge(\forall X)\delta(p, X)\ C\ \delta(q, X)$
단 $X\in\{Y\mid\delta(p, Y)\neq p$ or $\delta(q, Y)\neq q\}$.
$p$와 $q$가 LR($k$)상태들의 집합이라고 하자. 관계 호환의 정의역은 다음과 같이 LR($k$)상태들의 집합으로 확장될 수 있다. $p\ C\ q$ iff for all $(p, q)\in p\times q$, $p\ C\ q$.

다시말하면 두 개의 LR($k$)상태들이 호환이라함은 그 상태들이 서로 일시적 호환이고 각 상태들로부터 같은 기호에 의해 천이해간 모든 상태들이 서로 호환인 경우이다.

**성질 3.3** (1) $C$와 $C_t$는 반사적이고 대칭적인 참고문헌[11]의 *compatibility* 관계이다.
(2) $p, q\in C_k$ 이고 $\alpha\in V^*$ 일 때 $p\ C\ q$이면 $(\forall\alpha)\ \delta(p, \alpha)\ C\ \delta(q, \alpha)$이다

LR($k$)상태들의 집합들에 대한 관계 $C_t$는 아래와 같이 *semikernels*만을 검사하여 계산할 수 있다.

**정리 3.4** $p$와 $q$가 LR($k$)상태일 때 $p\ C_t\ q$ iff semikernel($p$) $C_t$ semikernel($q$).

위 정리에 대한 증명은 참고문헌[9]에 기술되어있다.
관계 $C$를 직접 계산하는 대신 그것의 보집합(complement)을 계산하기로 한다. 왜냐하면 Tarski의 Fixed-Point정리 [10]가 $C$의 보집합 계산에 쉽게 적용될 수 있기 때문이다.

**정의 3.5** (비호환) $p$와 $q$를 LR($k$)상태라고 하자
$p\not C_t\ q$ iff $\neg(p\ C_t\ q)$ 이고
$p\not C q$ iff $\neg(p\ C\ q)$ (또는 $p\not C_t\ q\ \vee$ ($\exists X$ such that $\delta(p, X)\not C\ \delta(q, X)$),
단 $X\in\{Y\mid\delta(p, Y)\neq p$ or $\delta(q, Y)\neq q\}$) 이다
문헌[10]의 함수 $f$의 LFP (Least Fixed-Point)에 해당하는 관계 $\not C$는 다음 알고리즘과 같이 계산될 수 있다.

**알고리즘 3.6** (관계 $\not C$의 계산)
입력 $M_k(G)=(C_k, V, \delta_k, q_{0k}, \emptyset)$ /* 문법 $G$에 대한 LR($k$)구문분석기 */
출력: $R$ /* $C_k$에 대한 관계 $\not C$의 행렬 */
방법.
1. **for** all $p, q\in C_k$ **do** $R[p, q]\leftarrow 0$ **endfor**;
2 **for** all $p, q\in C_k$ such that $core(p)\neq core(q)$ **do** $R[p, q]\leftarrow 1$ **endfor**;
3. **for** all $p, q\in C_k$ with the same core **do**
 **if** semikernel($p$) $\not C_t$ semikernel($q$) **then** $R[p, q]\leftarrow 1$ **endif**;
 **endfor**;
4 **repeat**
 **for** all $p, q\subseteq C_k$ with the same core **do**
 **if** $R[p, q]=0$ **then**
 **for** all $X\in V$ s.t. that $\delta(q, X)$ is defined **do**
 **if** $R[\delta(p, X), \delta(q, X)]=1$ **then** $R[p, q]\leftarrow 1$
 **endif**
 **endfor**
 **endfif**
 **endfor**
 **until** no change in $R$.

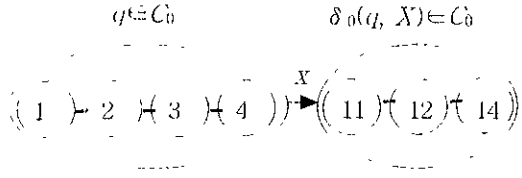
## 4. LR($k$)구문분석기의 축소

### 4.1 정축소

앞절에서의 논의와 같이 문맥자유문법 $G$에 대한 LR($k$)구문분석기가 주어지면 같은 코아블록내에서 서로

호환인 LR($k$)상태들은 충돌을 일으키지 않고 병합되어 하나의 상태로 될 수 있다. 얼핏보면 문법 $G$에 대한 LR($k$)-based구문분석기 구축을 위해 병합될 수 있는 한 블록내의 호환상태들을 결정하는 문제는 그 블록의 파티션을 찾는 문제처럼 보이지만, 다음 예에서 보이는 것처럼 그것은 블록에 대한 커버링을 계산하는 문제이다.

**예 4.1** (커버링의 필요성) 그림1에서 쌍 (1, 2)와 (3, 4)는 $q \in C_0$의 코아블록상에서 관계 $C$의 원소라고 하자. 각 쌍의 두 개의 상태들이 서로 병합될 수 있으면 병합된 상태들의 $X$-successor들 (즉 $\delta(\{1, 2\}, X)$ 와 $\delta(\{3, 4\}, X)$)은 각각 {11, 12}와 {12, 14}를 포함해야 한다. 여기에서 상태 12는 양쪽 successor들 모두에 포함되어야함을 주목하기 바란다. 따라서 {{11,12}, {12,14}}는 {11,12,14}의 하나의 커버링이다




$\delta_k(1,X)=11$, $\delta_k(2,X)=12$, $\delta_k(3,X)=12$, $\delta_k(4,X)=14$

그림 1 두 코아블록상의 상태천이와 관계 $C$

LR($k$)구문분석기의 축소를 설명하기위한 기본적인 정의를 기술한다.

**정의 4.2** (LR($k$)구문분석기의 *reduction*) $\{\Pi_1, \Pi_2, \ldots, \Pi_n\}$을 LR($k$)구문분석기의 코아동치파티션이라고 하자. $K_i$가 $1 \le i \le n$에 대해 $\Pi_i$의 커버링이면 $\{K_1, K_2, \cdots, K_n\}$을 주어진 구문분석기의 *reduction* 이라한다

정의 4.2에서 $K_i$의 원소내의 상태들이 LR($k$)-based구문분석기의 새로운 상태를 구성할 때 어떤 파싱충돌도 생기지 않도록 하려면 reduction들은 *intra-consistent*해야 한다

**정의 4.3** (*Intra-consistent reduction*) $Q$를 동일코아인 LR($k$)상태들의 집합이라고 하자.
집합 $\mathcal{S}$의 $C$-*covering*은 커버링의 임의의 원소내의 어떤 쌍에 대해서도 관계 $C$가 성립하는 $\mathcal{S}$의 커버링을 말한다. LR($k$)구문분석기의 reduction $\{K_1, K_2, \cdots, K_n\}$은 각 $K_i$ ($1 \le i \le n$)가 관련된 코아블록의 $C$-*covering*이면 *intra-consistent*하다.

나아가 코아블록의 커버링들 사이의 관계도, 얻고자하는 LR($k$)-based구문분석기에 의해 인식되는 언어가 원래의 LR($k$)구문분석기의 그것과 다르지않도록 고려되어야 하는데, 다음정의가 말해준다.

**정의 4.4** (*Goto-consistent reduction*) $X \in V$이고, $K_i$와 $K_j$가 LR($k$)구문분석기의 코아블록에 대한 커버링들이라 하자. 각 $\kappa \in K_i$에 대해서 $K_i$가 코아 $p_0$에대한 코아블록의 커버링이고 $K_j$가 $\delta_0(p_0, X)$에 대한 코아블록의 커버링이면 언제나 $K_j$가 $\delta_k(\kappa, X)$를 포함하는 원소를 가지는 경우에 $K_i$는 $K_j$와 *goto-consistent*하다고 말한다. 모든 $i$, $j$ ($1 \le i, j \le n$)에 대해 $K_i$가 $K_j$와 goto-consistent할때 reduction$\{K_1, K_2, \cdots, K_n\}$은 *goto-consistent* 하다고 말한다.

우리는 위에서의 논의들을 바탕으로 LR($k$)-based 구문분석기를 자연스럽게 얻게 해주는 정축소 (*well-defined reduction*)라는 새로운 개념에 도달할 수 있다.

**정의 4.5** (LR($k$) 구문분석기의 정축소(*WDR*)) Reduction $\{K_1, K_2, \cdots, K_n\}$은 그것이 intra-consistent하고 goto-consistent할 때 *well-defined*라고 한다.

**정의 4.6** (정축소에 정합인(conformable to) LR($k$)-based구문분석기) LR($k$)-based 오토마톤 $M(G) = (Q, V, \delta, q_0, \emptyset)$은 다음 조건들을 만족한때 $M_k(G) = (C_k, V, \delta_k, q_{0k}, \emptyset)$의 정축소 $\{K_1, K_2, \cdots, K_n\}$에 정합이라고 한다.

(1) $Q = \{\kappa \mid \kappa \in K_i, 1 \le i \le n\}$

(2) $q_0 = \{q_{0k}\}$, and

(3) $\delta(\kappa, X)$는 어떤 $K_j$ ($1 \le j \le n$)가 $\delta_k(\kappa, X)$를 가지는 원소를 포함하면 정의된다.

기본적으로 $M_k(G)$의 정축소에 정합인 LR($k$)-based 파싱오토마톤 $M(G)$는 정의 2.1로부터 correct prefix 성질을 가진다. 정의 4.3에 의해 $M(G)$의 어떤 상태에도 파싱충돌이 있을 수 없다. 또 정의 4.4는 $S \Rightarrow_{rm}^{*} \gamma x$, $q = \delta_k(q_{0k}, \gamma)$이고 $q = \delta(q_0, \gamma)$이면 $q$는 $q$에 포함된다는 것을 말해준다. 따라서 주어진 스트링에 대해 $M(G)$에 의해 수행되는 파싱액션 (parsing actions)

은 그 스트링에 대한 $M_k(G)$에 의한 파싱액션과 동치이다 (equivalent to).

주어진 LR($k$)구문분석기에대해 정축소와 그 정축소에 정합인 파싱오토마톤을 계산하는 나태한 알고리즘을 다음에 제시한다. 이 알고리즘에서 $\tau$가 LR($k$)상태집합의 집합일 경우 $\tau$의 모든원소들의 합집합을 $uncover(\tau)$로 표기한다. 예를들면 {{1, 2}, {2, 3}}가 새로운 LR($k$)-based 상태일 경우 이 상태는 {1, 2, 3} (=$uncover$({{1, 2}, {2, 3}}))으로 인식해도된다. 왜냐하면 하나의 LR($k$)-based상태는 정의 2.1에따라 그것을 구성하는 LR($k$)상태들에 의해 특징지워지기 때문이다. 다음 알고리즘의 실행 예는 참고문헌[9]에 제시되어있다

**알고리즘 4.7** (LR($k$)의 구문분석기의 정축소 및 정합 LR($k$)-based 파싱오토마톤의 계산)

입력: $M_k(G)=(C_k, V, \delta_k, q_{0k}, \emptyset)$,
$M_0(G)=(C_0, V, \delta_0, q_{00}, \emptyset)$,
$R$ /* $C_k$상에서의 $C$의 관계행렬 */

출력: $\{K_q \mid q\in C_0\}$ /* a $WDR$ of $M_k(G)$ */
$M(G)=(Q, V, \delta, q_0, \emptyset)$ /* $WDR$ 에 정합인 LR($k$)-based 파싱오토마톤 */

방법:

1 $Q\leftarrow \emptyset$; $\delta\leftarrow \emptyset$,

2. **for** each $q\in C_0$ **do** $K_q\leftarrow \{\{p\} \mid p\in C_k, core(p)=q\}$ **endfor** ;
/*초기에는 check off된 $K_q$는 없음 */

3. $Merge(q_{00})$. /* 상태병합 */

**procedure** $Merge(q)$ /* $q\in C_0$ */
$Q\leftarrow Q-K_q$;
**compute** the relation $C$ on $K_q$ using $R$;
$T_q \leftarrow$ a $C$-covering of $K_q$
$K_q \leftarrow \{ uncover(\tau) \mid \tau\in T_q \}$
$Q\leftarrow Q\cup K_q$;
check off $K_q$;
/* predecessors로부터의 천이를 수정하고 안쓰이는 천이를 제거 */
**for** each $\tau \in T_q$ **do**
**for** each $r\in \tau$ **do**
**for** each $p$ s.t. $\exists E\in V$ with $\delta(p, E)=r$ **do**
/* $E$는 상태 $q$의 entry symbol */
$\delta\leftarrow\delta-\{(p, E, r)\}$, $\delta\leftarrow\delta\cup\{(p, E, uncover(\tau))\}$,
/* $uncover(\tau)$: 새로운 상태 */
**endfor**
**for** each $s$ s.t $\exists X\in V$ with $\delta(r, X)=s$ **do**
$\delta\leftarrow\delta-\{(r, X, s)\}$
**endfor**
**endfor**;
**endfor**;

/* successors로의 천이를 추가하거나 수정 */
**for** each $X\in V$ such that $\delta_0(q, X)$is defined **do**
**for** each $K\in K_q$ **do**
$K_{succ}\leftarrow\delta_k(K, X)$;
**if** $K_{succ}\subseteq K'$ for some $K'\in K_{\delta(\delta_0(q, X))}$ **then**
$\delta\leftarrow\delta\cup\{(K, X, K')\}$,
**if** $K_{\delta(\delta_0(q, X))}$ is not checked off **then**
$Merge(\delta_0(q, X))$ **endif**
**else** $\delta\leftarrow\delta\cup\{(K, X, K_{succ})\}$,
$K_{\delta(\delta_0(q, X))}\leftarrow K_{\delta(\delta_0(q, X))}\cup\{K_{succ}\}$;
$Merge(\delta_0(q, X))$
**endif**
**endfor**
**endfor**
**endprocedure**

### 4.2 국부적 최적축소

주어진 집합에 대한 $C$-$covering$은 여러개 있을 수 있으므로, $C$-$covering$의 원소수가 관련된 상태들의 개수를 결정한다. 따라서 가장 적은 원소수를 가진 $C$-$covering$을 다음과 같이 기술한다.

**정의 4.8** (*Minimal C-covering*) $K$가 집합 $\mathcal{S}$의 $C$-$covering$이고 원소수가 $\mathcal{S}$의 어떤 $C$-$covering$의 그것보다 작거나 같을 때 $K$를 $\mathcal{S}$의 *minimal C-covering*이라고 한다.

주어진 $\mathcal{S}$의 *minimal C-covering*을 계산하기 위해 다음의 개념이 필요하다.

**정의 4.9** (*Maximal compatibility block, mcb-covering*[11]) $R$이 주어진 집합 $\mathcal{S}$에서의 compatibility 관계라고 하자. $A\times A\subseteq R$ 인 경우에 $\mathcal{S}$의 부분집합 $A$를 $\mathcal{S}$상의 관계 $R$의 *compatibility block* (*cb*)이라고 한다. 다시말하면 *compatibility block*내의 임의의 원소쌍에

대해서 관계 $R$이 성립한다. $\mathcal{S}$상의 관계 $R$의 *maximal compatibility block(mcb)*은 다른 어떤 *compatibility block*에 대해서도 부분집합이 될 수 없는 *compatibility block*을 말한다.

$\mathcal{S}$상의 관계 $R$의 *mcb-covering*은 $\mathcal{S}$상의 관계 $R$의 *mcb*들의 집합이다.

*Mcb*들은 반드시 서로 disjoint하지 않아도 됨을 주목해야한다. 그것들은 주어진 집합에 대한 하나의 커버링을 정의한다. Compatibility 관계에 대한 *mcb*들을 찾게 해주는 2가지 방법이 참고문헌[11]에 기술되어있다

**정의 4.10** (커버링에서 유도된 *minimal covering*) $K=\{K_1, \cdots, K_m\}$을 주어진 집합$\mathcal{S}$의 커버링이라고 하자 $\mathcal{S}$의 커버링 $K'$는, $K'$의 각 원소가 $K$의 원소일경우에 $K$에서 유도된 커버링이라고 한다 커버링 $K'$는, $K'$의 원소수가 $K$로부터 유도된 다른 어떤 커버링의 원소수보다 작거나 같을 때 *minimal*하다고 부른다

정의4.8, 4.9 및 4.10에 의해 다음 성질이 성립한다.

**성질 4.11** $K$가 집합 $\mathcal{S}$의 $C$-*covering*이라고 하자. $K$상의 관계 $C$의 *mcb-covering*에서 유도된 *minimal covering*은 $K$의 *minimal C-covering*이다.

**예 4.12** (*Minimal C-covering*, $\mathcal{S}$상의 관계 $C$의 *mcb-covering*에서 유도된 *minimal covering*) 그림 2는 $\mathcal{S}$(={1, 2, 3, 4, 5, 6})상의 관계 $C$를 말해주는 그래프이다

(i) *Minimal C-covering* : {{1, 4, 5}, {2, 3, 6}} (=$K_1$)

(ii) *maximal compatibility blocks* : {1, 2}, {3, 4}, {1, 4, 5}, {2, 3, 6};
$C$의 *mcb-covering* : {{1, 4, 5}, {1, 2}, {3, 4}, {2, 3, 6}}

(iii) $C$의 *mcb-covering*에서 유도된 커버링 들 : $K_2$ = {{1, 4, 5}, {1, 2}, {3, 4}, {2, 3, 6}}
$K_3$ = {{1, 4, 5}, {3, 4}, {2, 3, 6}},
$K_4$ = {{1, 4, 5}, {2, 3, 6}}등.
$C$의 *mcb-covering*에서 유도된 *minimal covering*.
$K_5$ = {{1, 4, 5}, {2, 3, 6}} ( =$K_4$ =$K_1$)

위의 예에서 만일 상태들이 (1, 2, 3, 4, 5, 6)과 같은 순서로 생성된다면 Pager[7]의 동적병합방법에서는 상태1과 2를 병합한후 상태3과 4를 병합할 것이다. 상태5는 위의 병합으로 생기는 어떤상태들과도 호환이 아니므로 별도의 상태로 남게 되고 상태6도 마찬가지다. 따라서 Pager의 방법으로는 4개의 상태로 되는데 비해 주어진 6개의 상태들에 대한 *minimal C-covering*에 의하면 2개의 상태가 된다

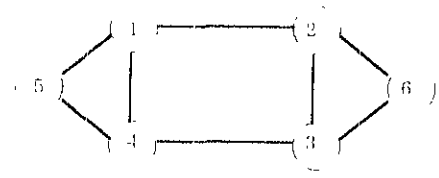



그림 2 $\mathcal{S}$상의 관계 $C$의 그래프

LR($k$)구문분석기의 국부적 최적축소(*locally optimal reduction, LOR*)는 알고리즘 4.7의 프로시저 *Merge*의 제3단계인

"$T_q$ ← a $C$-covering of $K_q$ ; "를

"$T_q$ ← a minimal $C$-*covering* of $K_q$ ; "로 대체하여 얻어지는 정축소이다. LALR($k$)문법 $G$에 대해서는 $G$의 LR($k$)구문분석기의 각 코아블록내의 상태들간의 관계 $C$를 그래프로 표현하면 완전그래프(complete graph)이므로 *LOR*은 오직 한 개뿐이고 그 *LOR*에 정합인 피싱오토마톤은 문법 $G$의 LALR($k$)구문분석기로서 유일하다는 사실에 유의해야한다. 국부적최적축소는 최적축소(*optimal reduction*)에 대한 근사값이라고 생각할수 있다.

LR($k$)구문분석기의 *LOR*계산을 위해, 알고리즘 4.7보다 효율적이고 LR(0)상태들의 선형순서를 필요로하는 또 하나의 알고리즘을 제안한다. 모든 LR(0)상태의 선형순서생성은 다음과 같이 얻을수 있다.

1) $\delta_0$내의 모든 자신으로의 천이 (self-transitions)를 제거한다.
2) Tarjan의 알고리즘 [12]을 사용하여 모든 nontrivial *strongly connected component* (*SCC*)를 찾아내어 이것을 *SCC*내에 포함된 임의의 상태로 대체한다
3) 새로 얻어진 $\delta_0$-그래프에 의해 주어진 부분순서(partial ordering)에 대한 topological sorting을 구한다
4) 하나의 선형순서를 선택하여 동일 *SCC*내의 모든 상태들의 순서가 연속적으로 되게 조정한다

주어진 선형순서하에서 reduction의 원소를 얻고자할

때 먼저 코아블록과 관련된 커버링의 *minimal C-covering*을 계산한다. 이어서 현재의 커버링과 각 관련된 커버링들이 서로 goto-consistent 하도록 현재 코아블록의 LR(0)상태의 각 successor상태들에 원소를 추가한다. 추가된 원소는 *minimal C-covering*의 원소내의 LR($k$)상태들의 $\alpha$-successor상태들의( $\alpha \in V^*$) 집합이다 (procedure *Pseudo-Merge*). Successors는 물론이고 predecessors로의 천이를 수정함으로서 *minimal C-covering*의 각 원소들은 $M(G)$내의 새로운 상태가 된다 (procedure *Real-Merge*).

**알고리즘 4.13** (*LOR* 및 이의 LR($k$)-based 오토마톤의 계산)

입력: $M_k(G)=(C_k, V, \delta_k, q_{0\,k}, \emptyset)$,
$M_0(G)=(C_0, V, \delta_0, q_{0\,0}, \emptyset)$,
$R$ /* $C_k$상에서의 $C$의 관계행렬 */, 및
$(q_1, q_2, \cdots, q_{|C0|})$ /* $C_0$내의 LR(0) 상태들의 선형순서 */

출력: $\{K_q \mid q \in C_0\}$ /* $M_k(G)$의 *LOR* */,
$M(G)=(Q, V, \delta, q_0, \emptyset)$

방법:

1. $Q \leftarrow \emptyset$, $\delta \leftarrow \emptyset$; $K_{q1} \leftarrow \{\{q_{0\,k}\}\}$, /* $q_1 = q_{0\,0}$ */
2. **for** $i$=2 **to** $|C_0|$ **do** $K_{qi} \leftarrow \emptyset$, $T_q \leftarrow \emptyset$ **endfor** ,
3. **for** $i$=1 **to** $|C_0|$ **do**
   a. $q \leftarrow q_i$ ;
   b. **compute** relation $C$ on $K_q$ using $R$ ;
   c $T_q \leftarrow$ a minimal $C$-covering of $K_q$ ;
   $K_q \leftarrow \{\text{uncover}(\tau) \mid \tau \in T_q\}$ ;
   d. **if** $q \in SCC$ **then**
   *Pseudo-Merge*($q$);
   **compute** relation $C$ on $K_q$ using $R$ ;
   $T_q \leftarrow$ a minimal $C$-covering of $K_q$ ,
   $K_q \leftarrow \{\text{uncover}(\tau) \mid \tau \in T_q\}$ ;
   **endif** ;
   e. *Real-Merge*($q$)
   **endfor**

**Procedure** *Pseudo-Merge*($q$), /* $q \in C_0$ */
**repeat**
 **for** each $X \in V$ such that $\delta_0(q, X)$ is defined **do**
  **for** each $\kappa \in K_q$ **do**
   $K_{\text{succ}} \leftarrow \delta_k(\kappa, X)$;
   **if** $K_{\text{succ}} \subseteq \kappa'$ for some $\kappa' \in K_{\delta 0(q,\ X)}$ **then**
    $\delta \leftarrow \delta \cup \{(\kappa, X, \kappa')\}$;
   **else**
    $\delta \leftarrow \delta \cup \{(\kappa, X, K_{\text{succ}})\}$
    $K_{\delta 0(q,\ X)} \leftarrow K_{\delta 0(q,\ X)} \cup \{K_{\text{succ}}\}$;
    *Pseudo-Merage*($\delta_0(q, X)$)
   **endif** ;
  **endfor**
 **endfor**
**until** $K_q$ is not changed
**endProcedure**

**Procedure** *Real-Merge*($q$) , /* $q \in C_0$ */
$Q \leftarrow Q \cup K_q$ ;
/* Predecessors로부터의 천이들을 수정하고 필요없게 된 천이들을 제거 */
**for** each $\tau \in T_q$ **do**
 **for** each $r \in \tau$ **do**
  **for** each $p$ s.t. $\exists E \in V$ with $\delta(p, E)=r$ **do**
  /* $E$는 상태 $q$의 entry symbol */
   $\delta \leftarrow \delta - \{(p, E, r)\}$,
   $\delta \leftarrow \delta \cup \{(p, E, uncover(\tau))\}$;
   **endfor**
   **for** each $s$ s.t. $\exists X \in V$ with $\delta(r, X)=s$ **do**
    $\delta \leftarrow \delta - \{(r, X, s)\}$
   **endfor**
  **endfor**
**endfor** ;

/* Successors로의 천이를 추가 또는 수정 */
**for** each $X \in V$ such that $\delta_0(q, X)$is defined **do**
 **for** each $\kappa \in K_q$ **do**
  $K_{\text{succ}} \leftarrow \delta_k(K, X)$;
  **if** $K_{\text{succ}} \subseteq \kappa'$ for some $\kappa'$ in $K_{\delta 0(q,\ X)}$ **then**
   $\delta \leftarrow \delta \cup \{(\kappa, X, \kappa')\}$,
  **else** $\delta \leftarrow \delta \cup \{(\kappa, X, K_{\text{succ}})\}$;
   $K_{\delta 0(q,\ X)} \leftarrow K_{\delta 0(q,\ X)} \cup \{K_{\text{succ}}\}$;
   *Pseudo-Merage*($\delta_0(q, X)$)
  **endif**
 **endfor**
**endProcedure**

**정의 4.14** (LR($k$)구문분석기의 최적축소) LR($k$)구문분석기의 정축소(*WDR*)는 그것의 모든 *C-covering*들의 원소수의 합이 다른 어떤 정축소의 *C-covering*들의 원소수의 합보다 작거나 같을 때 최적(optimal)이라고 한다.

**정리 4.15** LR($k$)구문분석기의 국부적최적축소가 반드시 최적축소인 것은 아니다

참고문헌[9]는 주어진 상태집합의 *minimal C-covering*이 다른 상태집합의 병합에서 항상 최상의 결과를 가져오지 않음을 보여주는 예를 들었다.

지금까지 제안된 *LOR*과 관련된 다른 연구를 적용가능성 및 병합과정에서의 되돌아감 (backtracking)의 유무를 고려하여 비교하면 다음과 같다.

1) Pager[7]의 방법은 동적병합이라는 특성 때문에 최적축소가 고려될 수 없는데 비해, *LOR*에서 동일 코아블록내의 상태들을 병합시 그 블록에 대한 *minimal C-covering* 의 원소수가 병합후의 상태갯수가 되지만, 정리 4.15에의해 다른 코아블록에서의 최상의 결과를 가져오지 않으므로, 알고리즘 4.13에서 방법3 c, d, e 를 $K_q$ 의 *minimal C-covering*에 대해서만 수행하지 않고, $K_q$ 의 모든 C-covering에 대해서 각각 수행했을 때의 축소결과를 얻어하면, 결과중에 최적축소가 포함된다
2) Pager[7]에서는 시험된 어떤 실용적인 문법에 대해서도 successor들을 재생성할 필요가 없었다고 기술했으나, 본 연구를 통해 Pager[7]의 방법에 의하면 successor 재생성이 필요한 77개의 LR(1)상태를 가지는 아래와 같은 생성규칙을 가지며 동일 코아블록내에 최대 5개의 LR(1)상태를 가지는 문법을 찾을 수 있었다. (병합 후의 상태는 Pager[7]방법으로는 51개, *LOR*방법으로는 50개)

생성규칙: *S→Aa, S→B, S→C, S→bAc, S→bBd, S→bCb, S→cbAb, S→cbB, S→cbC, S→dcbAc, S→dcbBa, S→dcbCd, S→edcbAc, S→edcbBa, S→edcbCb, A→aD, B→afD, C→ag, D→fD, D→g, D→h*

Heilbrunner의 알고리즘[4]은 상태병합 도중이 아닌 상태병합 종료 후 천이함수를 계산하고 또 successor재생성에 대한 고려를 하지 않는다

3) *LOR*과 동적방법[7]에서 각각 사용된 호환관계는 적용가능한 문법 범위가 다르다. *LOR*의 관계 *C*가 $k \geq 1$인 어떤 LR($k$)상태들의 병합을 위해서도 사용될 수 있는데 비해, 관계 strongly compatible[7]은 오직 LR(1)상태들의 병합에만 적용될 수 있다

**정리 4.16** 상태들의 개수가 3이하인 코아블록에 대해서는 동적병합방법[7]에 의해 축소된 상태들의 개수는 *minimal C-covering*에 의한 것과 같다.

(증명) 전체상태의 개수가 3이하인 상태들에대한 모든 가능한 관계그래프가 이를 말해준다. 그러나 다음경우 (상태개수가 4인 경우)는 다르다.

$$C = \{(q_1, q_3), (q_1, q_2), (q_2, q_4)\}$$

위에서 첨자는 해당상태의 생성순서를 표기한다

## 5. 결 론

LR($k$)구문분석기에 대한 정축소개념을 도입하여 최적축소에 대한 하나의 근사치로서의 국부적최적축소 계산을 위한 알고리즘을 제안하고, 국부적최적축소와 기존의 Pager[7]의 병합방법과의 비교를 통해 3가지 차이점이 있음을 확인했다 본 연구를 통해 주어진 LR($k$)상태들의 집합에 대해 상태병합을 한 결과가 원래 집합에 대한 파티션이 아니고 커버링이라는 점이 발견된 것은 주목할 만한 결과이다. 여러 전략과 기법이 국부적최적축소 계산을 위해 적용될 수 있겠으나 코아블록들의 병합순서와 국부적최적축소와의 관계, 국부적최적축소들의 집합내에 반드시 최적축소가 포함되어있는지의 여부 등의 규명이 추후 연구되어야 할것이다.

## 참 고 문 헌


[1] A.V Aho and J.D Ullman, *The Theory of Parsing, Translation and Compiling, Vols 1 and 2* (Prentice-Hall, Englewood Cliffs, NJ, 1972 and 1973)

[2] F.L DeRemer, Practical translators for LR($k$) languages, Thesis, Massachusetts Institute of Technology, Cambridge, MA, 1969.

[3] F.L. DeRemer, Simple LR($k$) grammars, *Comm. ACM* **14** (1971) 453-460.

[4] S. Heilbrunner, A parsing automata approach to LR theory, *Theoret Comput. Sci.* **15** (1981) 117-157

[5] D.E. Knuth, On the translation of languages from left to right, *Inform. and Control* **8** (1965) 607-639.

[6] M.J Lee and K.M. Choe, SLR($k$) covering for LR($k$) grammars, *Inform. Process. Lett.* **37** (1991) 337-347.

[7] D Pager, A practical general method for constructing LR($k$) parsers, *Acta Inform.* **7** (1977)

249-268.
[8] J.C.H. Park, K.M. Choe and C.H. chang, A new analysis of LALR formalisms, *ACM Trans. Prog. Lang. Sys.* 7 (1985) 159-175.
[9] W.J. Park, M.J. Lee and K.M. Choe, On the Reduction of LR(k) Parsers, Tech. Rept. No. CS-TR-92-70, Dept. of Computer Science KAIST, 1992.
[10] A. Tarski, A lattice theoretical fixed-point theorem and its applications, *Pacific J. Math.* 5 (1955) 285-309.
[11] J.P. Tremblay and R. Manohar, Discrete Mathematical Structures with Applications to Computer Science, (McGraw-Hill, New York, 1975).
[12] J. Eve and R. Kurki-Suonio, On computing the transitive closure of a relation, *Acta Inform.* 8 (1977) 303-314.

박 우 전
1973년 서울대학교 전기공학과 졸업(학사). 1980년 (일본) 電氣通信大學 정보공학과 졸업(석사). 1993년 한국과학기술원 전산학과 졸업(박사). 1977년 2월부터 1988년 8월까지 한국전자통신연구소 근무 (선임연구원). 1988년부터 한남대학교 전자계산공학과 재직중 (부교수). 관심분야는 프로그래밍언어, 컴파일러 구성 및 객체지향 기법 등임.

이 명 준
1980년 2월 서울대학교 수학과 졸업(학사). 1982년 2월 한국과학기술원 전산학과 졸업 (석사). 1991년 8월 한국과학기술원 전산학과 졸업(박사). 1982년 3월부터 울산대학교 전자계산학과 근무 (현재 교수). 1993년 8월부터 1994년 7월까지 미국 버지니아대학 교환교수. 관심분야는 프로그래밍언어, 분산객체 프로그래밍시스템, 병행실시간 컴퓨팅, 인터넷 프로그래밍시스템 등임.

최 광 무
제 23권 제 3권(B) 참조