SLR(1) and LALR(1) Parsing for Unrestricted Grammars

Lawrence A. Harris

Abstract

Simple LR(1) and lookahead LR(1) phrase structure grammars are defined and corresponding deterministic two-pushdown automata which parse all sentences are given. These grammars include a wide variety of grammars for non context-free languages. A given phrase structure grammar is one of these types if the parse table for the associated automaton has no multiple entries. A technique for construction of this parse table is given which in the lookahead case involves elimination of inverses in a grammar for lookahead strings for LR(0) items and computation of first sets for strings of symbols in the given grammar.

This paper extends the well-known SLR(1) and LALR(1) methods of deterministic bottom-up parsing to unrestricted grammars for use in compilers of higher level programming languages. Previous work developing deterministic parsing techniques for various classes of unrestricted grammars has been done, for example, by Loeckx [12], Walters [19], Barth [4], Sebesta and Jones [16], Turnbull and Lee [17], Kunze [11], Wegner [20], Vold'man [18] and Fisher [8]. In particular, the LR method of Knuth [10] has been extended to context sensitive grammars by Walters and, in a less general way, to unrestricted grammars by Turnbull and Lee.

Our SLR(1) and LALR(1) parsing methods show the same compactness, efficiency and simplicity as in the context-free case [6, 1] and apply to many grammars not covered under the methods given in [19] and [17]. Thus context dependent constructs can be included in a grammar specifying the syntax of a programming language and there is less likelihood that it will be necessary to transform this grammar into a canonical form, which frequently enlarges the grammar considerably and complicates the writing of semantic actions. A disadvantage of our method (and of the methods of [19] and [17]) is that there is no algorithm which produces the SLR(1) or LALR(1) parse tables for a given unrestricted grammar. However, we indicate a computational procedure which can be carried out by hand in many cases. We believe that a procedure might be found which can be implemented on a computer and which will construct the SLR(1) and LALR(1) parse tables for most of the grammars programming language designers would want to consider.

We begin in Section 1 by recalling basic definitions and fixing notation. In Section 2, we give a construction for the LR(0) sets of items and define unrestricted SLR(1) and LALR(1) grammars. We also introduce a grammar associated with the sets of items, called the lookahead grammar, where there is a nonterminal corresponding to each item in each state which derives all possible lookahead strings for the item. In Section 3, we define an LR(1) automaton as a special kind of nondeterministic two-pushdown automaton having only shift and reduce moves. When this automaton is given the SLR(1) parse table for an unrestricted grammar G it is called the SLR(1) automaton for G and when the automaton is given the LALR(1) parse table for G it is called the LALR(1) automaton for G. Both these types of automata have the same recognition power as that of a Turing machine. As expected, G is an unrestricted SLR(1) or LALR(1) grammar precisely when the corresponding automaton is deterministic.

Our main results are given in Section 4. The basic fact is that unrestricted SLR(1) and LALR(1) grammars are parsed deterministically by their corresponding automata and these simulate the reverse of a canonical derivation. For example, this provides a method to show that an unrestricted grammar is unambiguous or that a language is the language of a deterministic two-pushdown automaton. Also, context-free LALR(k) grammars can sometimes be converted to equivalent unrestricted SLR(1) or LALR(1) grammars which can be parsed efficiently. Further, a condition is given which insures that our automata do not read past an error.

In Section 5, we show that our methods apply to a large number of unrestricted grammars. In particular, we give unrestricted SLR(1) grammars for some of the most well-known languages that are not context free. We also give an unrestricted LALR(1) grammar where the associated automaton generates all the n digit binary numbers in reverse order on input of n zeros. Finally, we give an unrestricted LALR(1) grammar where the associated automaton loops through the same sequence of configurations on certain inputs. Proofs of all our results are collected in Section 6.

1. Unrestricted Grammars and Canonical Derivations

An *unrestricted grammar* is a 4-tuple $G = (\Sigma, V, P, S)$, where V is a finite set of symbols, $\Sigma \subseteq V$, $S \in V - \Sigma$ and P is a finite set of pairs $\lambda \rightarrow \mu$, where $\lambda, \mu \in V^*$ and λ contains at least one element of $V - \Sigma$. The elements of Σ are called *terminals,* the elements of $N = V - \Sigma$ are called *nonterminals* and the elements of P are called *productions.* Unrestricted grammars are also known as *phrase structure* or *Chomsky type 0* grammars. Clearly G is a context-free grammar exactly when $|\lambda| = 1$ for all productions $\lambda \rightarrow \mu$. We assume that all given grammars satisfy $\$ \notin V$ and we follow the usual typographical conventions [1, p. 87] for denoting terminals, nonterminals and strings of grammar symbols.

Let $n \geq 1$. A sequence $\sigma_1, ..., \sigma_{n+1} \in V^*$ is called a *derivation* and written $\sigma_1 \Longrightarrow ... \Longrightarrow \sigma_{n+1}$ if there exists strings $\phi_1, ..., \phi_n \in V^*$ and $\psi_1, ..., \psi_n \in V^*$ and productions $\lambda_1 \to \mu_1$, ..., $\lambda_n \to \mu_n$ such that $\sigma_i = \phi_i \lambda_i \psi_i$ and $\sigma_{i+1} = \phi_i \mu_i \psi_i$ for $i = 1, ..., n$. This is just the condition that it is possible to obtain the next string in the sequence by replacing a substring λ of the current string by the right-hand side μ of a production $\lambda \to \mu$. We write $\sigma_1 \overset{*}{\Longrightarrow} \sigma_{n+1}$ and say that σ_1 *derives* σ_{n+1} if $\sigma_{n+1} = \sigma_1$ or if there exists a derivation $\sigma_1 \Longrightarrow ... \Longrightarrow \sigma_{n+1}$. Define the *language of* G by

$$L(G) = \{w \in \Sigma^* : S \overset{*}{\Longrightarrow} w\}.$$

A derivation is said to be *(right-) canonical* if ϕ_{i+1} is a proper prefix of $\phi_i \mu_i$ for $i = 1, ..., n-1$. In other words, the string replaced at a derivation step starts at or to the left of the last symbol introduced in the previous step. We write $\sigma_1 \underset{c}{\Longrightarrow} ... \underset{c}{\Longrightarrow} \sigma_{n+1}$ if $\sigma_1 \Longrightarrow ... \Longrightarrow \sigma_{n+1}$ is a canonical derivation, and we write $\sigma_1 \overset{*}{\underset{c}{\Longrightarrow}} \sigma_{n+1}$ if $\sigma_1 = \sigma_{n+1}$ or if there exists a derivation $\sigma_1 \underset{c}{\Longrightarrow} ... \underset{c}{\Longrightarrow} \sigma_{n+1}$. For context-free grammars, any rightmost derivation is clearly a canonical derivation but a canonical derivation is not necessarily a rightmost derivation except, for example, when the string derived is in Σ^*. Griffiths [9] has defined an equivalence relation on derivations corresponding to a change in the order of application of production rules which do not interact with each other and has shown that each of these equivalence classes contains exactly one canonical derivation. (One can translate his results from left-canonical derivations to right-canonical derivations by replacing the left- and right-hand sides of the production rules in the given grammar by their reverses. See also [5].) For example, if $S \overset{*}{\Longrightarrow} \sigma$ then $S \overset{*}{\underset{c}{\Longrightarrow}} \sigma$.

An unrestricted grammar $G = (\Sigma, V, P, S)$ is said to be *ambiguous* if there exists a $w \in \Sigma^*$ and two distinct derivations $S \overset{*}{\underset{c}{\Longrightarrow}} w$. Two derivations are considered distinct unless both sequences of strings are the same and the productions applied are the same and are applied in the same order at the same locations. Note that the notation we employ for derivations, although standard, does not specify explicitly the production rules applied at each step and therefore may be ambiguous. In our discussion, the rules applied will always be clear from the context.

2. Unrestricted SLR(1) and LALR(1) Grammars

Let $G = (\Sigma,\ V,\ P,\ S)$ be an unrestricted grammar. Augment G by adding a new nonterminal S' to V and a new production $S' \to S$ to P. We call $[\,\lambda \to \mu_1 \cdot \mu_2\,]$ an *LR(0) item for G* if $\lambda \to \mu_1\mu_2$ is a production. We view the LR(0) items as the states of a nondeterministic finite automaton with starting state $[\,S' \to \cdot S\,]$ and transitions as follows: For each item $[\,\lambda \to \mu_1 \cdot X\mu_2\,]$ there is a transition on X to the item $[\,\lambda \to \mu_1 X \cdot \mu_2\,]$ and an ϵ-transition to any item $[\,X\delta \to \cdot\eta\,]$, where $X \in V$. Then applying the subset construction [2, p.91], we obtain a deterministic finite automaton with states $Q = \{q_0, ..., q_N\}$ and transition function GOTO: $\subseteq Q \times V \to Q$. (We extend the function GOTO to a maximal subset of $Q \times V^*$ by iteration.) Each of the states q_n is a non-empty set of items and the starting state q_0 contains the item $[\,S' \to \cdot S\,]$. A basic property is that the nondeterministic finite automaton arrives in a state $[\,\lambda \to \mu_1 \cdot \mu_2\,]$ upon reading a string γ if and only if $[\,\lambda \to \mu_1 \cdot \mu_2\,]$ is in the state GOTO$(q_0,\ \gamma)$. We call the set Q the *preliminary LR(0) collection for G* and we call a set in this collection a *preliminary LR(0) state.* (Clearly the preliminary LR(0) collection for G is just the canonical LR(0) collection in the sense of [1, p.384] for the context-free grammar one obtains from G by ignoring all symbols but the first on the left-hand side of productions and viewing these first symbols as nonterminals.) Where convenient, we follow the usual convention of referring to a state q_n as state n and listing the items without brackets in tables of states.

Proposition 1. *Let* $\lambda \to \mu_1\mu_2$ *be a production of* G *and let* $\phi,\, \psi \in V^*$*. If*

$$S' \underset{c}{\overset{*}{\Longrightarrow}} \phi\lambda\psi \underset{c}{\Longrightarrow} \phi\mu_1\mu_2\psi \tag{1}$$

then $q_n = GOTO(q_0,\, \phi\mu_1)$ *exists and* $[\,\lambda \to \mu_1 \cdot \mu_2\,] \in q_n$*.*

If $\gamma \in V^*$ and $\gamma = \phi\mu_1$, where ϕ and μ_1 are as in (1), then we call any prefix of γ a *viable prefix* and we say that $[\,\lambda \to \mu_1 \cdot \mu_2\,]$ *is valid for* γ. Unlike the context-free case, there are strings γ such that $q = \mathrm{GOTO}(q_0,\, \gamma)$ exists and either of the conditions holds:

a) γ is not a viable prefix,

b) one item in q is valid for γ while another is not. (See Example 1 in Section 5 below.)

Let $\Omega(G)$ denote the set of all $X \in V$ such that X appears on the left-hand side of a production of G in a position other than the first and put $\Theta(G) = \Sigma \cup \Omega(G)$. Define FOLLOW($\lambda$) to be the set of all $W \in \Theta(G) \cup \{\$\}$ such that $S' \underset{c}{\overset{*}{\Longrightarrow}} \phi\lambda\psi \underset{c}{\Longrightarrow} \phi\mu\psi$ for some $\phi,\, \psi \in V^*$ and some production $\lambda \to \mu$ where W is the first symbol of $\psi\,\$$.

Definition. An unrestricted grammar is called *unrestricted SLR(1)* if

a) $X \notin$ FOLLOW(λ) whenever $[\,\gamma \to \nu_1 \cdot X\nu_2\,]$ and $[\,\lambda \to \mu\cdot\,]$ are in the same preliminary LR(0) state.

b) FOLLOW(λ) $\cap$ FOLLOW(γ) $= \emptyset$ whenever $[\,\lambda \to \mu\cdot\,]$ and $[\,\gamma \to \nu\cdot\,]$ are distinct items in the same preliminary LR(0) state.

Clearly our definition agrees with the usual definition of SLR(1) for context-free grammars [1]. A state having a pair of items as in the latter part of (a) or (b) above is said to be *inadequate.*

For each $X \in \Omega(G)$, create a new symbol X^{-1} and define

$$(X_1 \cdots X_n)^{-1} = X_n^{-1} \cdots X_1^{-1},$$

for X_1, ..., $X_n \in \Omega(G)$. We define the *lookahead grammar for G* to be the grammar with productions

$$[\,0,\, S' \rightarrow \cdot S\,] \rightarrow \$, \tag{2}$$

$$[\,m,\, \lambda \rightarrow \mu_1 X \cdot \mu_2\,] \rightarrow [\,n,\, \lambda \rightarrow \mu_1 \cdot X \mu_2\,], \tag{3}$$

$$[\,n,\, X\delta \rightarrow \cdot \eta\,] \rightarrow \delta^{-1}\mu_2\,[\,n,\, \lambda \rightarrow \mu_1 \cdot X\mu_2\,], \tag{4}$$

$$X^{-1}X \rightarrow \epsilon, \quad X \in \Omega(G), \tag{5}$$

and including all productions of G. Here $[\,\lambda \rightarrow \mu_1 \cdot X\mu_2\,] \in q_n$, GOTO$(q_n, X) = q_m$, and $X\delta \rightarrow \eta$ is a production of G. We define the set of terminals of the lookahead grammar to be $\Theta(G) \cup \{\$\}$. Put $\Omega^{-1}(G) = \{X^{-1} : X \in \Omega(G)\}$. By Gaussian elimination [1, p. 106], one can obtain a regular expression over $V \cup \Omega^{-1}(G) \cup \{\$\}$ for a set of strings which derive the same strings in $V^*\$$ that $[\,n,\, \lambda \rightarrow \mu_1 \cdot \mu_2\,]$ derives. The following result shows that these strings are all possible lookahead strings for the item $[\,\lambda \rightarrow \mu_1 \cdot \mu_2\,]$ of state n.

Proposition 2. *Let $\lambda \rightarrow \mu_1\mu_2$ be a production of G and let $\psi \in V^*$. Then $S' \overset{*}{\underset{c}{\Longrightarrow}} \phi\lambda\psi \underset{c}{\Longrightarrow} \phi\mu_1\mu_2\psi$ and GOTO(q_0, $\phi\mu_1$)$=$ q_n for some $\phi \in V^*$ if and only if $[\,n,\, \lambda \rightarrow \mu_1 \cdot \mu_2\,] \overset{*}{\Longrightarrow} \psi\,\$$.*

For example, by Proposition 2 if G is context free then $[\,A \rightarrow \alpha \cdot \beta,\, w\,]$ is an item in a set q of items in the LALR(k) collection for G if and only if $[\,A \rightarrow \alpha \cdot \beta\,]$ is an item in the core of q and $w \in \mathrm{FIRST}_k([\,n,\, A \rightarrow \alpha \cdot \beta\,])$. The latter set can be computed as in [1, p.357]. (See [3] for a related approach.)

In general, when I is an item in state q_n of the preliminary LR(0) collection for G, we define FIRST($[n, I]$) to be the set of all $W \in \Theta(G) \cup \{\$\}$ such that there is a derivation $[n,\, I] \overset{*}{\underset{c}{\Longrightarrow}} \sigma$ in the lookahead grammar where $\sigma \in V^*\$$ and W is the first symbol of σ. We define the LR(0) collection to be the preliminary LR(0) collection where an item I is removed from a state q_n when $[n,\, I]$ does not derive any string in $V^*\$$ and where any resulting empty states are removed. A transition from a state on a symbol X is removed when the state no longer contains an item of the form $[\,\lambda \to \mu_1 \cdot X\mu_2\,]$. Clearly Propositions 1 and 2 still hold for the LR(0) collection.

Definition. An unrestricted grammar is called *unrestricted LALR(1)* if

a) $X \notin \mathrm{FIRST}([\,n,\, \lambda \to \mu\cdot\,])$ whenever $[\,\lambda \to \mu\cdot\,]$ and $[\,\gamma \to \nu_1 \cdot X\nu_2\,]$ are in state q_n of the LR(0) collection.

b) $\mathrm{FIRST}([\,n,\, \lambda \to \mu\cdot\,]) \cap \mathrm{FIRST}([\,n,\, \gamma \to \nu\cdot\,]) = \emptyset$, whenever $[\,\lambda \to \mu\cdot\,]$ and $[\,\gamma \to \nu\cdot\,]$ are distinct items in state q_n of the LR(0) collection.

Any unrestricted SLR(1) grammar is unrestricted LALR(1) since

$$\mathrm{FIRST}([\,n,\, \lambda \to \mu\cdot\,]) \;\subseteq\; \mathrm{FOLLOW}(\lambda) \tag{6}$$

for all productions $\lambda \to \mu$ of G by Proposition 2. Clearly our definition agrees with the usual definition [1] of LALR(1) for context-free grammars by Proposition 2.

3. LR(1) automata

Intuitively, the automata we define to parse unrestricted grammars are deterministic finite automata where the states traversed and symbols read are saved in stacks. When a state is obtained where the last symbols read agree with a given string and when the next input symbol is right, the automaton returns to the state where it was just before the string was read and an alternate string to be read is inserted at the front of the remaining unread input. More formally, a *LR(1) automaton* is a 6-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, S')$, where Q is a finite set with $q_0 \in Q$, Γ is a finite set of symbols satisfying $\Sigma \subseteq \Gamma$ and \$, $S' \notin \Gamma$, and $\delta : Q \times (\Gamma \cup \{\$\}) \rightarrow F(A)$ is a function, where

$$A = \{(S, q) : q \in Q, q \neq q_0\} \cup \{(R, \lambda, \mu) : \lambda \in \Gamma^* \cup \{S'\}, \mu \in \Gamma^*\}$$

and $F(A)$ is the set of all finite subsets of A. An interpretation of these symbols is as follows: Q is a set of states with q_0 the initial state, Σ is the alphabet of permissible input strings, Γ is the stack alphabet, δ defines a table of parsing actions from A, and S' signals the end of a successful parse. A *configuration of M* is a triple $(\pi, \alpha, \beta\,\$)$, where $\pi \in Q^*$ with $\pi \neq \epsilon$, $\alpha \in \Gamma^*$ and $\beta \in \Gamma^* \cup \{S'\}$. We call π the *state stack,* α the *parsing stack* and $\beta\,\$$ the *input stack.* We consider the top of π and α to be on the right and the top of $\beta\,\$$ to be on the left. Suppose $\pi = \pi' q$ and $\beta\,\$ = X\beta'$. If $(S, r) \in \delta(q, X)$ and $X \neq \$$, we write

$$(\pi, \alpha, \beta\,\$) \vdash (\pi r, \alpha X, \beta') \tag{7}$$

and we call (7) a *shift move to state r.* If $(R, \lambda, \mu) \in \delta(q, X)$ and $\alpha = \alpha_0\mu$, we write

$$(\pi, \alpha, \beta\,\$) \vdash (\pi_0, \alpha_0, \lambda\beta\,\$), \tag{8}$$

where $\pi = \pi_0\hat{\pi}_0$ and $|\hat{\pi}_0| = |\mu|$, and we call (8) a *reduction move by* $\lambda \rightarrow \mu$. Note that the possible next moves in a given configuration depend only on the symbols on top of the state and input stacks. Clearly the parsing and state stacks operate in parallel so that $|\pi| = |\alpha| + 1$. The parsing stack may be viewed as part of the output of the automaton.

If $(\pi_1,\, \alpha_1,\, \beta_1\$)$ and $(\pi_n,\, \alpha_n,\, \beta_n\$)$ are configurations, we write

$$(\pi_1,\, \alpha_1,\, \beta_1\$) \overset{*}{\vdash} (\pi_n,\, \alpha_n,\, \beta_n\$)$$

if $(\pi_1,\, \alpha_1,\, \beta_1\,\$) \;=\; (\pi_n,\, \alpha_n,\, \beta_n\$)$ or if there exists a sequence of moves

$$(\pi_1,\, \alpha_1,\, \beta_1\$) \vdash \cdots \vdash (\pi_n,\, \alpha_n,\, \beta_n\$).$$

The language of M is defined by

$$L(M) \;=\; \{w \in \Sigma^* : \, (q_0,\, \epsilon,\, w\$) \overset{*}{\vdash} (q_0,\, \epsilon,\, S'\,\$)\,\} \tag{9}$$

and the configurations in (9) are called the *initial* and *final* configurations, respectively. Note that the final configuration is determined by the top symbols of the state and input stacks and that there is no move possible in the final configuration. It is not difficult to show that an LR(1) automaton can be simulated by a (nondeterministic) two-pushdown automaton as defined in [12] or [15] and that the languages accepted are the same. Our definition is similar to the definition of the CS(1) processors given by Walters [19].

Define an LR(1) automaton M to be *deterministic* if $\delta(q,\, X)$ contains at most one element for each $q \in Q$ and $X \in \Gamma \cup \{\$\}$. When M is deterministic we regard δ as a function defined on a subset of $Q \times (\Gamma \cup \{\$\})$ with values in A. Given an unrestricted grammar $G \;=\; (\Sigma,\; V,\; P,\; S)$, we construct an LR(1) automaton $M \;=\; (Q,\, \Sigma,\, \Gamma,\, \delta,\, q_0,\, S')$ in two different ways. In both, $\Gamma \;=\; V$ and S' is the new nonterminal introduced to augment G. Define the *SLR(1) automaton for* G to be the automaton M where Q is the preliminary LR(0) collection and where $\delta(q,\, X)$ consists of all pairs $(S,\, r)$ satisfying $r \;=\mathrm{GOTO}(q,\, X)$ and all triples $(R,\, \lambda,\, \mu)$ satisfying $[\,\lambda \;\rightarrow\; \mu\cdot\,] \;\in q$ and $X \in \mathrm{FOLLOW}(\lambda)$. Define the *LALR(1) automaton for* G to be the automaton M where Q is the LR(0) collection and where $\delta(q,\, X)$ consists of all pairs $(S,\, r)$ satisfying $r \;=\mathrm{GOTO}(q,\, X)$ and all triples $(R,\, \lambda,\, \mu)$

satisfying $[\lambda \to \mu\cdot] \in q$, $q = q_n$ and $X \in \mathrm{FIRST}([n, \lambda \to \mu\cdot])$. *Note that the SLR(1) (resp., LALR(1)) automaton for G is deterministic if and only if G is unrestricted SLR(1) (resp., LALR(1)).*

The next result shows that, unlike the context-free case, there is no algorithm which can compute every SLR(1) or LALR(1) parsing table.

Proposition 3. *For an arbitrary unrestricted grammar G, it is undecidable whether $\delta(q_0, \$)$ contains at most one element, where δ is the parsing action function of the SLR(1) (resp., LALR(1)) automaton for G.*

Let M be an SLR(1) or LALR(1) automaton for G. Clearly there is at most one shift move in any configuration. Also, the parsing stack can be obtained directly from the state stack in any configuration $(\pi, \alpha, \beta\,\$)$ arrived at by the automaton beginning in an initial configuration. Indeed, $\alpha = f(\pi)$, where $f : Q^* \to V^*$ is the multiplicative extension of the function which takes a state q to the symbol preceding the dot of an item in the kernel of q. Moreover, after each reduction move by a production $X\delta \to \eta$ different from $S' \to S$, there exists a shift move by X. (See Lemma 2 below.) We shall always assume that this shift move is taken after the reduction move.

In implementations of SLR(1) or LALR(1) automata, it is convenient to read a symbol from the input string into the input stack only when the input stack is empty. Thus the automaton's input stack may be viewed as the implemented input stack followed by the remaining unread input. It is also convenient to combine a reduction move as above with the following shift move so that δ is pushed onto the input stack and X and the appropriate state are pushed onto the grammar and state stacks, respectively.

4. Main Results

Proposition 4 (below) shows that SLR(1) and LALR(1) automata simulate the reverse of a canonical derivation. An immediate consequence is the following theorem, which establishes that unrestricted SLR(1) and LALR(1) grammars are parsed deterministically by their corresponding automata. The author has found that a large number of interesting unrestricted grammars are covered by this theorem.

Theorem 1. *Let G be an unrestricted SLR(1) (resp., LALR(1)) grammar and let $w \in \Sigma^*$. Then $w \in L(G)$ if and only if the SLR(1) (resp., LALR(1)) automaton with initial configuration $(q_0,\, \epsilon,\, w\$)$ halts in configuration $(q_0,\, \epsilon,\, S'\$)$. In this case, the successive reduction moves are reductions by the productions used in a canonical derivation but in reverse order.*

Corollary 1. *No unrestricted LALR(1) grammar is ambiguous.*

Proposition 4. *Let M be the SLR(1) or LALR(1) automaton for an unrestricted grammar G. Then the following are equivalent:*

a) There exists a derivation

$$S' = \phi_1 \lambda_1 \psi_1 \underset{c}{\Longrightarrow} \cdots \underset{c}{\Longrightarrow} \phi_n \mu_n \psi_n = \sigma,$$

where the productions $\lambda_1 \to \mu_1$, ..., $\lambda_n \to \mu_n$ have been applied successively and the first symbol of each ψ_i is in $\Theta(G)$ when $\psi_i \neq \epsilon$.

b) There exist moves of M beginning in the initial configuration $(q_0,\, \epsilon,\, \sigma\,\$)$ such that M arrives at configurations

$$(\pi_n,\, \phi_n,\, \lambda_n \psi_n \$) \vdash \cdots \vdash (\pi_1,\, \phi_1,\, \lambda_1 \psi_1 \$) = (q_0,\, \epsilon,\, S'\$)$$

after each reduction move, where the reductions are by $\lambda_n \to \mu_n$, ..., $\lambda_1 \to \mu_1$, respectively.

Corollary 2. *L(G) = L(M).*

In view of Proposition 4, grammars more general than the unrestricted LALR(1) grammars might still be parsed by methods employing backtrack or parallel computation.

Unlike the context-free case, an LALR(1) automaton for an unrestricted grammar may not detect an error until it has read arbitrarily many symbols beyond a symbol which cannot follow the string previously read. Moreover, the automaton can cycle indefinitely on certain (non-sentence) inputs. (See Examples 1 and 6 below.) The following result gives a condition which insures that errors will be detected as early as possible.

Proposition 5. *Let G be an unrestricted SLR(1) (resp., LALR(1)) grammar and suppose that γ is a viable prefix whenever GOTO(q_0, γ) is defined for the preliminary LR(0) (resp., LR(0)) collection. Then the SLR(1) (resp., LALR(1)) automaton for G reads a symbol only when it follows the symbols previously read in some sentential form.*

Clearly any context-free SLR(1) or LALR(1) grammar G satisfies the hypotheses of Proposition 5.

As an example of the usefulness of Theorem 1, we give a technique to convert certain context-free LALR(k) grammars to equivalent unrestricted SLR(1) or LALR(1) grammars where the associated automaton parses the original grammar in an obvious way. In many instances, this approach is a considerably simpler alternative to the methods of [14].

Define a production $A \to \alpha$ of a context-free grammar G to be an LALR(p) production of G if p is the least non-negative integer such that no completed LALR(p) item with production $A \to \alpha$ is involved in an LALR(p) conflict, i.e., there are no items $[\,A \to \alpha\cdot, x\,]$ and $[\,B \to \beta_1 \cdot \beta_2, y\,]$ in the same LALR(p)

state where $x \in \mathrm{EFF}_p(\beta_2 y)$. (See [1, p.381].) Given a context-free LALR(k) grammar $G = (\Sigma, V, P, S)$ where $k > 1$, choose $Z \notin V$ and let $G' = (\Sigma, V \cup \{Z\}, P', S')$ be the unrestricted grammar with productions $S' \to SZ$ and $Z \to \epsilon$ and productions $Au \to \alpha u$ (resp., $AuZ \to \alpha uZ$) where $A \to \alpha$ is an LALR(p) production of G, $u \in \Sigma^*$ and u (resp., $u\$$) is in $\mathrm{FOLLOW}_{p-1}(A)$. The latter set is assumed to be $\{\epsilon\}$ when $p \le 1$. It is not difficult to show that $L(G') = L(G)$. (See Example 5 below.)

5. Examples

Example 1. Let G_1 be the grammar

(1) $S \to aSBD$	(2) $S \to abD$	(3) $DB \to BD$
(4) $bB \to bb$	(5) $D \to c$.	

Then $L(G_1) = \{a^n b^n c^n : n \ge 1\}$ and $\Theta(G_1) = \{a, b, c, B\}$. The preliminary LR(0) states are given in Table 1. State 8 is the only inadequate state. Note that all the items of state 2 are valid for the viable prefix a^n when $n \ge 1$ except that $[\, bB \to \cdot bb\,]$ is not valid for a. For the SLR(1) case, one can easily obtain the FOLLOW sets of the left-hand sides of productions and thus the table for the function δ of the SLR(1) automaton M for G_1. (See Tables 4 and 5.) Clearly G_1 is unrestricted SLR(1) since Table 5 has no multiple entries. Table 6 shows the configurations and moves of M as it parses the sentence $a^2b^2c^2$. It can be shown that if $1 \le n < m$ then M reads the string $a^n b^m c^n$ before it halts signaling an error.

For the LALR(1) case, the productions of the lookahead grammar are constructed as the preliminary LR(0) states are generated and these are solved as a system of regular expression equations. (See Tables 2 and 3.) In these tables, $[n, k, p\,]$ denotes the item nonterminal $[n, \lambda \to \mu_1 \cdot \mu_2\,]$, where $\lambda \to \mu_1 \cdot \mu_2$ is the

kth production and $|\mu_1| = p$. In Table 3, all inverses in the regular expressions have been eliminated by the application of productions (5) and productions of G_1. In doing this, one must be careful to insure that on the application of each production the resulting regular expression derives the same strings of $V^*\$$ as the original one. Since each item nonterminal of the lookahead grammar derives a string in $V^*\$$, the LR(0) states and the preliminary LR(0) states agree. One then computes the FIRST sets of those item nonterminals where the item is completed and the table for the function δ of the LALR(1) automaton follows immediately. (See Tables 4 and 5.) For G_1 it happens that the SLR(1) and LALR(1) tables coincide.

Table 1: Preliminary LR(0) states for G_1

State 0
$S' \to \cdot S$
$S \to \cdot aSBD$
$S \to \cdot abD$

State 1
$S' \to S\cdot$

State 2
$S \to \cdot aSBD$
$S \to a \cdot SBD$
$S \to \cdot abD$
$S \to a \cdot bD$
$bB \to \cdot bb$

State 3
$S \to aS \cdot BD$

State 4
$S \to ab \cdot D$
$DB \to \cdot BD$
$bB \to \cdot bb$
$bB \to b \cdot b$
$D \to \cdot c$

State 5
$S \to aSB \cdot D$
$DB \to \cdot BD$
$D \to \cdot c$

State 6
$DB \to \cdot BD$
$DB \to B \cdot D$
$D \to \cdot c$

State 7
$S \to abD\cdot$

State 8
$bB \to \cdot bb$
$bB \to b \cdot b$
$bB \to bb\cdot$

State 9
$D \to c\cdot$

State 10
$S \to aSBD\cdot$

State 11
$DB \to BD\cdot$

Table 2: Lookahead grammar for G_1 (without productions of G_1)

$[0,0,0] \rightarrow \$$
$[0,1,0] \rightarrow [0,0,0]$
$[0,2,0] \rightarrow [0,0,0]$

$[1,0,1] \rightarrow [0,0,0]$

$[2,1,0] \rightarrow BD\,[2,1,1]$
$[2,1,1] \rightarrow [0,1,0]$
$[2,1,1] \rightarrow [2,1,0]$
$[2,2,0] \rightarrow BD\,[2,1,1]$
$[2,2,1] \rightarrow [0,2,0]$
$[2,2,1] \rightarrow [2,2,0]$
$[2,4,0] \rightarrow B^{-1}D\,[2,2,1]$
$[2,4,0] \rightarrow B^{-1}b\,[2,4,0]$

$[3,1,2] \rightarrow [2,1,1]$

$[4,2,2] \rightarrow [2,2,1]$
$[4,3,0] \rightarrow B^{-1}[4,2,2]$
$[4,4,0] \rightarrow B^{-1}b\,[4,4,0]$
$[4,4,0] \rightarrow B^{-1}[4,4,1]$
$[4,4,1] \rightarrow [2,4,0]$
$[4,5,0] \rightarrow [4,2,2]$

$[5,1,3] \rightarrow [3,1,2]$
$[5,3,0] \rightarrow B^{-1}[5,1,3]$
$[5,5,0] \rightarrow [5,1,3]$

$[6,3,0] \rightarrow B^{-1}[6,3,1]$
$[6,3,1] \rightarrow [4,3,0]$
$[6,3,1] \rightarrow [5,3,0]$
$[6,3,1] \rightarrow [6,3,0]$
$[6,5,0] \rightarrow [6,3,1]$

$[7,2,3] \rightarrow [4,2,2]$

$[8,4,0] \rightarrow B^{-1}b\,[8,4,0]$
$[8,4,0] \rightarrow B^{-1}[8,4,1]$
$[8,4,1] \rightarrow [4,4,0]$
$[8,4,1] \rightarrow [8,4,0]$
$[8,4,2] \rightarrow [4,4,1]$
$[8,4,2] \rightarrow [8,4,1]$

$[9,5,1] \rightarrow [4,5,0]$
$[9,5,1] \rightarrow [5,5,0]$
$[9,5,1] \rightarrow [6,5,0]$

$[10,1,4] \rightarrow [5,1,3]$

$[11,3,2] \rightarrow [6,3,1]$

$B^{-1}B \rightarrow \epsilon$

Table 3: Regular expressions for lookahead grammar for G_1

$[0,0,0] = \$$
$[0,1,0] = \$$
$[0,2,0] = \$$

$[1,0,1] = \$$

$[2,1,0] = (BD)^+\$$
$[2,1,1] = (BD)^*\$$
$[2,2,0] = (BD)^+\$$
$[2,2,1] = (BD)^*\$$
$[2,4,0] = D^2(BD)^*\$$

$[3,1,2] = (BD)^*\$$

$[4,2,2] = (BD)^*\$$
$[4,3,0] = D(BD)^*\$$
$[4,4,0] = D^3(BD)^*\$$
$[4,4,1] = D^2(BD)^*\$$
$[4,5,0] = (BD)^*\$$

$[5,1,3] = (BD)^*\$$
$[5,3,0] = D(BD)^*\$$
$[5,5,0] = (BD)^*\$$

$[6,3,0] = D^2D^*(BD)^*\$$
$[6,3,1] = D^+(BD)^*\$$
$[6,5,0] = D^+(BD)^*\$$

$[7,2,3] = (BD)^*\$$

$[8,4,0] = D^4D^*(BD)^*\$$
$[8,4,1] = D^3D^*(BD)^*\$$
$[8,4,2] = D^2D^*(BD)^*\$$

$[9,5,1] = D^*(BD)^*\$$

$[10,1,4] = (BD)^*\$$

$[11,3,2] = D^+(BD)^*\$$

Table 4: SLR(1) and LALR(1) lookahead symbols for G_1

FOLLOW(S') $= \{\$\}$	FOLLOW(S) $= \{\$,\ B\}$
FOLLOW(DB) $= \{c,\ B\}$	FOLLOW(bB) $= \{c,\ B\}$
FOLLOW(D) $= \{\$,\ c,\ B\}$	
FIRST($[1, 0, 1]$) $= \{\$\}$	FIRST($[7, 2, 3]$) $= \{\$,\ B\}$
FIRST($[8, 4, 2]$) $= \{c,\ B\}$	FIRST($[9, 5, 1]$) $= \{\$,\ c,\ B\}$
FIRST($[10, 1, 4]$) $= \{\$,\ B\}$	FIRST($[11, 3, 2]$) $= \{c,\ B\}$

Table 5: SLR(1) and LALR(1) parse table for G_1

State	a	b	c	$	B	D	S
0	S2						S1
1				R0			
2	S2	S4					S3
3					S5		
4		S8	S9		S6	S7	
5			S9		S6	S10	
6			S9		S6	S11	
7				R2	R2		
8		S8	R4		R4		
9			R5	R5	R5		
10				R1	R1		
11			R3		R3		

Table 6: Parse of $a^2b^2c^2$

π	α	β \$	move
0	ϵ	aabbcc\$	S2
0 2	a	abbcc\$	S2
0 2 2	aa	bbcc\$	S4
0 2 2 4	aab	bcc\$	S8
0 2 2 4 8	aabb	cc\$	R4
0 2 2	aa	bBcc\$	S4
0 2 2 4	aab	Bcc\$	S6
0 2 2 4 6	aabB	cc\$	S9
0 2 2 4 6 9	aabBc	c\$	R5
0 2 2 4 6	aabB	Dc\$	S11
0 2 2 4 6 11	aabBD	c\$	R3
0 2 2 4	aab	DBc\$	S7
0 2 2 4 7	aabD	Bc\$	R2
0 2	a	SBc\$	S3
0 2 3	aS	Bc\$	S5
0 2 3 5	aSB	c\$	S9
0 2 3 5 9	aSBc	\$	R5
0 2 3 5	aSB	D\$	S10
0 2 3 5 10	aSBD	\$	R1
0	ϵ	S\$	S1
0 1	S	\$	R0
0	ϵ	S′\$	

It is easy to show as above that each of the grammars below is an unrestricted SLR(1) grammar with the indicated language and FOLLOW sets. The SLR(1) parse table can be obtained easily in each case.

$$L(G_2) \;=\; \{a^{2^n} : n \geq 0\}$$

(1) $S \rightarrow AaDE$	(2) $aD \rightarrow Da$	(3) $AD \rightarrow AC$
(4) $Ca \rightarrow aaC$	(5) $CE \rightarrow DE$	(6) $CE \rightarrow B$
(7) $aB \rightarrow Ba$	(8) $AB \rightarrow \epsilon$	

FOLLOW(S') = $\{\$\}$	FOLLOW(S) = $\{\$\}$
FOLLOW(aD) = $\{a,\ E\}$	FOLLOW(AD) = $\{a\}$
FOLLOW(Ca) = $\{a,\ E\}$	FOLLOW(CE) = $\{\$\}$
FOLLOW(aB) = $\{a,\ \$\}$	FOLLOW(AB) = $\{a\}$

$$L(G_3) \;=\; \{nha^n : n \geq 1\}$$

(1) $S \rightarrow TE$	(2) $T \rightarrow 1TaF$	(3) $T \rightarrow 0TF$
(4) $T \rightarrow h$	(5) $Fa \rightarrow aaF$	(6) $FE \rightarrow E$
(7) $aE \rightarrow a$		

FOLLOW(S') = $\{\$\}$	FOLLOW(S) = $\{\$\}$
FOLLOW(T) = $\{a,\ E\}$	FOLLOW(Fa) = $\{a,\ E\}$
FOLLOW(FE) = $\{\$\}$	FOLLOW(aE) = $\{\$\}$

$$L(G_4) \;=\; \{wcw : w \in (a|b)^*\}$$

(1) $S \rightarrow CD$	(2) $C \rightarrow aCA$	(3) $C \rightarrow bCB$
(4) $C \rightarrow c$	(5) $AD \rightarrow aD$	(6) $BD \rightarrow bD$
(7) $Aa \rightarrow aA$	(8) $Ba \rightarrow aB$	(9) $Ab \rightarrow bA$
(10) $Bb \rightarrow bB$	(11) $D \rightarrow \epsilon$	

$\text{FOLLOW}(S') = \{\$\}$	$\text{FOLLOW}(S) = \{\$\}$
$\text{FOLLOW}(C) = \{a,\ b,\ D,\ \$\}$	$\text{FOLLOW}(AD) = \{\$\}$
$\text{FOLLOW}(BD) = \{\$\}$	$\text{FOLLOW}(Aa) = \{a,\ b,\ D\}$
$\text{FOLLOW}(Ba) = \{a,\ b,\ D\}$	$\text{FOLLOW}(Ab) = \{a,\ b,\ D\}$
$\text{FOLLOW}(Bb) = \{a,\ b,\ D\}$	$\text{FOLLOW}(D) = \{\$\}$

$$L(G_5) = \{a^n b^m c^n d^m : n, m \geq 1\}$$

(1) $S \to aSC$	(2) $S \to aTC$	(3) $T \to bTD$
(4) $T \to bD$	(5) $DC \to CD$	(6) $bC \to bc$
(7) $cC \to cc$	(8) $D \to d$	

$\text{FOLLOW}(S') = \{\$\}$	$\text{FOLLOW}(S) = \{C,\ \$\}$
$\text{FOLLOW}(T) = \{d,\ C\}$	$\text{FOLLOW}(DC) = \{d,\ C,\ \$\}$
$\text{FOLLOW}(bC) = \{d,\ C\}$	$\text{FOLLOW}(cC) = \{d,\ C\}$
$\text{FOLLOW}(D) = \{d,\ C,\ \$\}$	

Example 2. Let G_6 be the grammar

(1) $S \to EAE$	(2) $EA \to EC$	(3) $CA \to AAC$
(4) $CE \to AAE$	(5) $A \to a$	(6) $E \to \epsilon$

Then $L(G_6) = \{a^{2^n} : n \geq 0\}$ and $\Theta(G_6) = \{a,\ A,\ E\}$. Since $\text{FOLLOW}(E) = \{a,\ A,\ \$\}$, there are both shift and reduce moves for a and A in states 3 and 7 of the SLR(1) automaton for G_6. Hence G_6 is not an unrestricted SLR(1) grammar. However, it is not difficult to show that G_6 is an unrestricted LALR(1) grammar. (See Tables 7 and 8.)

Table 7: LR(0) states for G_6

State 0
$S' \to \cdot S$
$S \to \cdot EAE$
$EA \to \cdot EC$
$E \to \epsilon \cdot$

State 1
$S' \to S \cdot$

State 2
$S \to E \cdot AE$
$EA \to E \cdot C$
$CA \to \cdot AAC$
$CE \to \cdot AAE$
$A \to \cdot a$

State 3
$S \to EA \cdot E$
$CA \to A \cdot AC$
$CE \to A \cdot AE$
$A \to \cdot a$
$E \to \epsilon \cdot$

State 4
$EA \to EC \cdot$

State 5
$A \to a \cdot$

State 6
$S \to EAE \cdot$

State 7
$CA \to \cdot AAC$
$CA \to AA \cdot C$
$CE \to \cdot AAE$
$CE \to AA \cdot E$
$A \to \cdot a$
$E \to \epsilon \cdot$

State 8
$CA \to A \cdot AC$
$CE \to A \cdot AE$
$A \to \cdot a$

State 9
$CE \to AAE \cdot$

State 10
$CA \to AAC \cdot$

Table 8: LALR(1) parse table for G_6

State	a	A	E	$	C	S
0	R6	R6	S2			S1
1				R0		
2	S5	S3			S4	
3	S5	S7	S6	R6		
4	R2	R2	R2	R2		
5	R5	R5	R5	R5		
6				R1		
7	S5	S8	S9	R6	S10	
8	S5	S7				
9				R4		
10	R3	R3	R3	R3		

Example 3. Let G_7 be the grammar

(1) $S \rightarrow FTE$	(2) $T \rightarrow 0T$	(3) $T \rightarrow 0$
(4) $0E \rightarrow 1E$	(5) $1E \rightarrow B0C$	(6) $0B \rightarrow 1A$
(7) $1B \rightarrow B0$	(8) $FB \rightarrow D$	(9) $A0 \rightarrow 0A$
(10) $AC \rightarrow E$	(11) $D0 \rightarrow 0D$	(12) $DC \rightarrow \epsilon$

Then $L(G_7) = \{0^n : n \geq 1\}$. It is not difficult to show that G_7 is an unrestricted LALR(1) grammar and to construct its LALR(1) parse table. (G_7 is not an unrestricted SLR(1) grammar.) Unlike the case of unambiguous context-free grammars [7] where a string of length n can always be parsed in time $O(n^2)$, the string 0^n requires time $O(2^n)$ to parse. Indeed, the LALR(1) automaton for G_7

requires $7 \cdot 2^n + n$ reductions (not counting the last reduction by $S' \rightarrow S$) and a total of three times as many moves to parse 0^{n+1} for $n \geq 0$. If the contents of the parsing stack followed by the contents of the input stack are printed just before each reduction by $T \rightarrow 0$, $0E \rightarrow 1E$ and $AC \rightarrow E$, then the output is a backward listing of the n digit binary numbers.

Example 4. Following Turnbull and Lee [17 p.200], let G_8 be the grammar

(1) $S \rightarrow ABSc$	(2) $S \rightarrow Abc$	(3) $Ab \rightarrow ab$
(4) $Aa \rightarrow aa$	(5) $Bb \rightarrow bb$	(6) $BA \rightarrow AB$

and note that $L(G_8) = \{a^n b^n c^n : n \geq 1\}$. It is easy to see that none of the preliminary LR(0) states is inadequate so G_8 is an unrestricted SLR(1) grammar. It is also easy to verify that if the transition function for the LR(0) collection is defined for a string γ, then γ is a prefix of one of the strings

$$S, \quad (AB)^n A^m aa, \quad (AB)^n A^m ab, \quad (AB)^n Abc,$$
$$(AB)^n ABSc, \quad (AB)^n A^{m+2} bb, \quad (AB)^n A^{m+2} B,$$

where $n, m \geq 0$, and therefore that γ is a viable prefix. Hence by Proposition 4 the LALR(1) automaton for G_8 reads a symbol only when it follows the symbols previously read in some sentential form. (This is not true of the SLR(1) automaton for G_8.)

Example 5. Let G_9 be the LALR(2) grammar

(1) $S \rightarrow bSS$	(2) $S \rightarrow a$	(3) $S \rightarrow aac$

It is easy to see that productions (1) and (3) are LALR(0) and that production (2) is LALR(2). Since FOLLOW(S) = $\{a, b, \$\}$, the equivalent unrestricted grammar G'_9 is

(1) $S' \to SZ$ (2) $Z \to \epsilon$ (3) $S \to bSS$
(4) $Sa \to aa$ (5) $Sb \to ab$ (6) $SZ \to aZ$
(7) $S \to aac$

and it is easy to verify that this grammar is unrestricted SLR(1). Note that the method of [14] obtains an equivalent context-free SLR(1) grammar with 16 productions. In fact, it is easy to verify that G' is unrestricted SLR(1) when G is any of the LR(2) examples of [14].

Example 6. Let G_{10} be the grammar

(1) $S \to AT$ (2) $S \to pTcd$ (3) $T \to U$
(4) $T \to bc$ (5) $Ab \to AU$ (6) $U \to bF$
(7) $Fcd \to cd$ (8) $A \to a$ (9) $U \to q$

Then G_{10} is an unrestricted LALR(1) grammar and the LALR(1) automaton for G_{10} on input *abcd* cycles infinitely through the same 9 configurations after 2 initial moves. The corresponding derivation cycle is

$$Abcd \underset{c}{\Longrightarrow} AUcd \underset{c}{\Longrightarrow} AbFcd \underset{c}{\Longrightarrow} Abcd.$$

Here the reduction of *cd* to Fcd is not correct since the item $[\,Fcd \to cd\cdot\,]$ is not valid for any viable prefix beginning with A. On the other hand, this item is valid for the sentence *pbcd* and the state arrived at on reading *pbcd* is the same as the state arrived at on reading *Abcd*. Hence the LALR(1) automaton must make the erroneous reduction.

It can be shown that when productions (1) and (5) are replaced by the productions $S \to AS$, $S \to T$ and $AAb \to AU$, the resulting grammar is still unrestricted LALR(1) and the corresponding LALR(1) automaton on input *abcd* cycles through a sequence of 10 moves in such a way that the parsing stack grows without bound. Note that although the two stack machine of Turnbull and Lee enters an infinite loop on certain inputs for the grammar of [17, p.195], the SLR(1) automaton for this grammar is deterministic and does not loop.

6. Proofs of Propositions and Theorems

Lemma 1. *Let $X\delta \to \eta$ be a production and let $S' \overset{*}{\underset{c}{\Longrightarrow}} \alpha X\delta\beta \underset{c}{\Longrightarrow} \alpha\eta\beta$ be a derivation with at least two steps. Then there exists a production $\lambda \to \nu_1 X \nu_2$ and ϕ, $\psi \in V^*$ such that the first steps of the given derivation are $S' \overset{*}{\underset{c}{\Longrightarrow}} \phi\lambda\psi \underset{c}{\Longrightarrow} \phi\nu_1 X\nu_2\psi$ and both $\phi\nu_1 = \alpha$ and $\nu_2\psi \overset{*}{\underset{c}{\Longrightarrow}} \delta\beta$.*

Proof. Let the given canonical derivation be $\sigma_1 \underset{c}{\Longrightarrow} \ldots \underset{c}{\Longrightarrow} \sigma_{n+1}$, where the σ_i's are given as in the definition of a derivation in §1. Let k be the smallest number less than n such that αX is a prefix of ϕ_i for all $k < i < n$. (Possibly $k = n-1$.) Then for each $k < i < n$, there is a $\theta_i \in V^*$ with $\alpha X\theta_i = \phi_i$ and

$$\theta_{k+1}\lambda_{k+1}\psi_{k+1} \underset{c}{\Longrightarrow} \ldots \underset{c}{\Longrightarrow} \theta_{n-1}\mu_{n-1}\psi_{n-1} = \delta\beta$$

if $k < n-1$. Also, since the given derivation is canonical and αX is a prefix of ϕ_{k+1} when $k < n-1$, we see that αX is a prefix of $\phi_k\mu_k$. Since αX is not a prefix of ϕ_k, we may write $\mu_k = \nu_1 X \nu_2$ and $\alpha = \phi_k\nu_1$. If $k < n-1$ then

$$\alpha X\nu_2\psi_k = \phi_k\mu_k\psi_k = \alpha X\theta_{k+1}\lambda_{k+1}\psi_{k+1}\,,$$

so $\nu_2\psi_k = \theta_{k+1}\lambda_{k+1}\psi_{k+1}$, and if $k = n-1$ then $\alpha X\nu_2\psi_k = \alpha X\delta\beta$, so $\nu_2\psi_k = \delta\beta$. Hence $\nu_2\psi_k \overset{*}{\underset{c}{\Longrightarrow}} \delta\beta$ and clearly

$$S' \overset{*}{\underset{c}{\Longrightarrow}} \phi_k\lambda_k\psi_k \underset{c}{\Longrightarrow} \phi_k\nu_1 X\nu_2\psi_k.$$

Proof of Proposition 1. It suffices to show that there exists a path of transitions through the NFA of LR(0) items from $[\,S' \to \cdot S\,]$ to $[\,\lambda \to \mu_1 \cdot \mu_2\,]$ reading $\phi\mu_1$. The proof is by induction on the number of steps of the derivation (1). Suppose the derivation is one step. Then $\phi = \psi = \epsilon$, $\lambda = S'$ and $\mu_1\mu_2 = S$. If $\mu_1 = \epsilon$, then $\phi\mu_1 = \epsilon$ and the path of no transitions takes $[\, S' \to \cdot S \,]$ to itself. If $\mu_1 = S$, then $\phi\mu_1 = S$ and there is a transition on S from $[\, S' \to \cdot S \,]$ to $[\, S' \to S \cdot \,]$.

Suppose such a path exists when the number of steps in (1) is less than m and let

$$S' \underset{c}{\overset{*}{\Longrightarrow}} \alpha X \delta \beta \underset{c}{\Longrightarrow} \alpha \eta_1 \eta_2 \beta \tag{10}$$

be a derivation with m steps, where $X\delta \to \eta_1\eta_2$ is a production. Then by Lemma 1 there is a path of transitions from $[\, S' \to \cdot S \,]$ to $[\lambda \to \nu_1 \cdot X\nu_2]$ reading $\alpha = \phi\nu_1$. Since there is an ϵ-transition from $[\, \lambda \to \nu_1 \cdot X\nu_2 \,]$ to $[\, X\delta \to \cdot\eta_1\eta_2]$ and there is a path of transitions from $[\, X\delta \to \cdot\eta_1\eta_2]$ to $[\, X\delta \to \eta_1 \cdot \eta_2]$ reading η_1, there is a path of transitions from $[\, S' \to \cdot S \,]$ to $[\, X\delta \to \eta_1 \cdot \eta_2]$ reading $\alpha\eta_1$.

Proof of Proposition 2. We prove the equivalence by induction on the number of steps in the given derivations. To prove the forward implication, suppose the given derivation has only one step. Then $\phi = \psi = \epsilon$, $\lambda = S'$ and $\mu_1\mu_2 = S$. If $\mu_1 = \epsilon$, then $n = 0$ and $[\, 0,\, S' \to \cdot S \,] \overset{*}{\Longrightarrow} \$$, and if $\mu_1 = S$, then

$$[\, n,\, S' \to S \cdot \,] \Longrightarrow [\, 0,\, S' \to \cdot S \,] \Longrightarrow \$,$$

as required.

Suppose the forward implication holds for all given derivations with fewer than m steps and let a derivation (10) be given with m steps and with GOTO$(q_0,\, \alpha\eta_1)$ $= q_n$. Then by Lemma 1, $[\, k,\, \lambda \to \nu_1 \cdot X\nu_2]\overset{*}{\Longrightarrow} \psi\,\$$ and $\delta^{-1}\nu_2\psi \overset{*}{\Longrightarrow} \beta$, where GOTO$(q_0,\, \alpha) = q_k$. Hence since $q_n =$ GOTO$(q_k,\, \eta_1)$, we have

$$[\, n,\, X\delta \to \eta_1 \cdot \eta_2 \,] \overset{*}{\Longrightarrow} [\, k,\, X\delta \to \cdot\eta_1\eta_2 \,] \Longrightarrow \delta^{-1}\nu_2 [\, k,\, \lambda \to \nu_1 \cdot X\nu_2 \,] \overset{*}{\Longrightarrow} \beta\,\$,$$

as required.

To prove the reverse implication, suppose the given derivation has only one step. Then it is $[\, 0,\, S' \to \cdot S \,] \Longrightarrow \$$ and clearly $S' \underset{c}{\overset{*}{\Longrightarrow}} S' \underset{c}{\Longrightarrow} S$ and GOTO$(q_0,\, \epsilon)$ $= q_0$, as required.

Suppose the reverse implication holds when the given derivation has m steps and let $[\,n,\, \lambda \;\rightarrow\; \mu_1 \cdot \mu_2] \overset{*}{\Longrightarrow} \psi\,\$$ be an $m+1$ step derivation. Consider the case where $\mu_1 \neq \epsilon$. Then we may write $\mu_1 = \mu_1' X$ and clearly

$$[n, \lambda \rightarrow \mu_1 \cdot \mu_2] \Longrightarrow [k, \lambda \rightarrow \mu_1' \cdot X\mu_2] \overset{*}{\Longrightarrow} \psi\$$$

for some state q_k with $\mathrm{GOTO}(q_k,\, X) \;=\; q_n$. By the induction hypothesis, there is a derivation

$$S' \underset{c}{\overset{*}{\Longrightarrow}} \phi\lambda\psi \underset{c}{\Longrightarrow} \phi\mu_1\mu_2\psi$$

and $\mathrm{GOTO}(q_0,\, \phi\mu_1') \;=\; q_k$. Hence $\mathrm{GOTO}(q_0,\, \phi\mu_1) \;=\; q_n$, as required. In the case where $\mu_1 \;=\; \epsilon$, we have

$$[\,n,\, X\delta \;\rightarrow\; \mu_1 \cdot \mu_2\,] \Longrightarrow \delta^{\,-1}\nu_2\,[\,n,\, \gamma \;\rightarrow\; \nu_1 \cdot X\nu_2\,] \overset{*}{\Longrightarrow} \psi\$,$$

where $\lambda \;=\; X\delta$. Since the last derivation may be taken to be canonical, there is a string β of symbols in $V \,\cup\, \Omega^{-1}(G)$ with $[\,n,\, \gamma \;\rightarrow\; \nu_1 \cdot X\nu_2\,] \underset{c}{\overset{*}{\Longrightarrow}} \beta\$$ and $\delta^{\,-1}\nu_2\beta \underset{c}{\overset{*}{\Longrightarrow}} \psi$. By the induction hypothesis, there exists a derivation

$$S' \underset{c}{\overset{*}{\Longrightarrow}} \alpha\gamma\beta \underset{c}{\Longrightarrow} \alpha\nu_1 X\nu_2\beta$$

and $\mathrm{GOTO}(q_0,\, \alpha\nu_1) \;=\; q_n$. Put $\phi \;=\; \alpha\nu_1$. Since $\nu_2\beta \underset{c}{\overset{*}{\Longrightarrow}} \delta\psi$, we have

$$S' \underset{c}{\overset{*}{\Longrightarrow}} \phi\lambda\psi \underset{c}{\Longrightarrow} \phi\mu_1\mu_2\psi\,,$$

as required.

Proof of Proposition 3. Given an unrestricted grammar G, we may suppose that S does not appear in the right-hand side of any production. Add the production $S \;\rightarrow\; \epsilon$ and consider the LALR(1) automaton of the resulting grammar. Clearly $[\,S \;\rightarrow\; \epsilon\cdot\,] \;\in\; q_0$ and $\$ \;\in \mathrm{FIRST}([\,0,\, S \;\rightarrow\; \epsilon\cdot\,])$. Hence $|\,\delta(q_0, \$)\,| \;>\; 1$

if and only if there is a production $\lambda \to \epsilon$ of G with $[\lambda \to \epsilon\cdot] \in q_0$ and $\$ \in \text{FIRST}([0, \lambda \to \epsilon\cdot])$. By Proposition 2, the latter condition is equivalent to $\epsilon \in L(G)$, and this is undecidable [15, p.102].

Given an unrestricted grammar G, create new symbols S' and Z, and let S' be the new start symbol. Add productions $S' \to \epsilon$ and $S' \to ZSZ$, and add the production $Z\lambda Z \to \lambda$ for each production $\lambda \to \epsilon$ of G. Consider the SLR(1) automaton of the resulting grammar. Clearly $[S' \to \epsilon\cdot] \in q_0$ and $\text{FOLLOW}(S') = \{\$\}$. Also $\$ \in \text{FOLLOW}(\lambda)$ where $\lambda \to \mu$ is a production if and only if $\mu = \epsilon$ and

$$S' \overset{*}{\Longrightarrow} Z\lambda Z \Longrightarrow \lambda \Longrightarrow \epsilon.$$

Hence $|\,\delta(q_0, \$)\,| > 1$ if and only if $\epsilon \in L(G)$.

Lemma 2. *Let p and q be (preliminary) LR(0) states and suppose that $[X\delta \to \eta\cdot] \in q$ and GOTO(p, η)$=q$. Then GOTO(p, X) is a (preliminary) LR(0) state unless $X\delta \to \eta$ is the production $S' \to S$.*

Proof. In both cases it is easy to show that $[X\delta \to \cdot\eta] \in p$. If the given states are preliminary LR(0) states, then there is an item $[\lambda \to \mu_1 \cdot X\mu_2] \in p$ since p is the closure of its kernel. The same conclusion also holds for LR(0) states by Lemma 1.

Proof of Proposition 4. To avoid considering the initial configuration separately, define $\pi_{n+1} = q_0$, $\phi_{n+1} = \lambda_{n+1} = \epsilon$ and $\psi_{n+1} = \sigma$. Suppose (b) holds and suppose M is in a configuration $(\pi_{k+1}, \phi_{k+1}, \lambda_{k+1}\psi_{k+1}\$)$, where $1 \le k \le n$. Let $\theta_{k+1} \in V^*$ be the string of symbols shifted from the input stack until the reduction move by $\lambda_k \to \mu_k$ and let $\hat{\pi}_{k+1} \in Q^*$ be the string of the corresponding states entered. Since the configuration after that move is $(\pi_k, \phi_k, \lambda_k\psi_k\$)$, we have $\phi_{k+1}\theta_{k+1} = \phi_k\mu_k$ and $\pi_{k+1}\hat{\pi}_{k+1} = \pi_k\hat{\pi}_k$, where $|\mu_k| = |\hat{\pi}_k|$. Also, the next move of M is a shift move by our assumption so $\theta_k \neq \epsilon$. Hence

$$\phi_k\lambda_k\psi_k \Longrightarrow \phi_k\mu_k\psi_k = \phi_{k+1}\lambda_{k+1}\psi_{k+1},$$

where the production $\lambda_k \to \mu_k$ has been applied and ϕ_{k+1} is a proper prefix of $\phi_k\mu_k$. Thus (a) holds.

Suppose (a) holds and suppose M arrives at a configuration $(\pi_{k+1},\, \phi_{k+1},\, \lambda_{k+1}\psi_{k+1}\$)$, where π_{k+1} is the string of states traversed upon reading ϕ_{k+1} from state q_0 and $1 \leq k \leq n$. Since the given derivation is canonical, there is a $\theta_{k+1} \in V^*$ with $\phi_{k+1}\theta_{k+1} = \phi_k\mu_k$ and $\theta_{k+1} \neq \epsilon$ if $k \neq n$. By Proposition 1, there is a string $\hat{\pi}_{k+1}$ of states traversed upon reading θ_{k+1} from the last state of π_{k+1}. Then we may write $\pi_{k+1}\hat{\pi}_{k+1} = \pi_k\hat{\pi}_k$, where π_k is the string of states traversed upon reading ϕ_k and $|\hat{\pi}_k| = |\mu_k|$. Hence since $\phi_{k+1}\lambda_{k+1}\psi_{k+1} = \phi_k\mu_k\psi_k$, we have

$$(\pi_{k+1},\, \phi_{k+1},\, \lambda_{k+1}\psi_{k+1}\$) \overset{*}{\vdash} (\pi_k\hat{\pi}_k,\, \phi_k\mu_k,\, \psi_k\$)$$

through successive shift moves. By Proposition 1, the item $[\,\lambda_k \to \mu_k\cdot\,]$ is in the last state q of $\pi_k\hat{\pi}_k$, and the first symbol W of $\psi_k\$$ is in $\Theta(G) \cup \{\$\}$. Hence by Proposition 2, $\delta(q,\, W)$ contains $(R,\, \lambda_k,\, \mu_k)$, so

$$(\pi_k\hat{\pi}_k,\, \phi_k\mu_k,\, \psi_k\$) \vdash (\pi_k,\, \phi_k,\, \lambda_k\psi_k\$)$$

through a reduction move by $\lambda_k \to \mu_k$. This proves (b).

Theorem 1 and Corollary 2 follow immediately from Proposition 4 and the observation that if $\sigma \in \Sigma^*$ then $\psi_i \in \Theta(G)^*$ for all $1 \leq i \leq n$; indeed, if ψ_i contains a suffix γ with first symbol in $V - \Theta(G)$ then γ is also a suffix of ψ_{i+1}.

Proof of Proposition 5. Let M be the mentioned automaton and suppose M reads a string $w \in \Sigma^*$, i.e., there exists a $\pi \in Q^*$ and an $\alpha \in V^*$ with $(q_0,\, \epsilon,\, w\$) \overset{*}{\vdash} (\pi,\, \alpha,\, \$)$. It can be shown as in the proof of Proposition 4 that

$\alpha \overset{*}{\underset{c}{\Longrightarrow}} w$. Clearly GOTO($q_0$, α) is defined since π is a path of transitions (in the appropriate collection of states) reading α. Hence by hypothesis, there exists a derivation $S' \overset{*}{\underset{c}{\Longrightarrow}} \phi\lambda\psi \underset{c}{\Longrightarrow} \phi\mu\psi$, where α is a prefix of $\phi\mu$. Write $\alpha\beta = \phi\mu$. Then $S' \overset{*}{\underset{c}{\Longrightarrow}} \alpha\beta\psi \overset{*}{\underset{c}{\Longrightarrow}} w\beta\psi$, as required.

Remark. To simplify computations of parse tables, we have chosen the most restrictive of three obvious definitions for unrestricted SLR(1) and LALR(1) grammars. These different definitions arise from the fact that a sentential form in an unrestricted grammar may not derive a sentence. (It is often difficult to determine whether this is the case for a given sentential form.) One can obtain a more general definition by requiring that $\psi \in \Theta(G)^*$ and $\sigma \in \Theta(G)^*\$$ in the definitions of FOLLOW(λ) and FIRST($[n, I]$), respectively. The LR(0) collection is obtained by removing all items I from q_n where FIRST($[n, I]$) $= \emptyset$. Then Proposition 4 holds when $\sigma \in \Theta(G)^*$.

One can obtain a still more general definition by requiring that the derivation in the definition of FOLLOW(λ) and the derivation in Proposition 2 giving an equivalent formulation of the definition of FIRST($[n, I]$) derive sentences. The LR(0) collection is defined in terms of this definition of FIRST as before. Then Proposition 4 holds when $\sigma \in \Sigma^*$. This definition has the advantage that each table entry is exercised in the parse of some sentence.

Note that all three definitions agree for reduced context-free grammars.

References

1. Aho, A.V., Ullman, J.D.: The theory of parsing, translation, and compiling, Vols. I & II. Englewood Cliffs, N.J.: Prentice-Hall 1972

2. Aho, A.V., Ullman, J.D.: Principles of compiler design. Reading, Mass.: Addison-Wesley 1977

3. Ancona, M., Dodero, G., Gianuzzi, V.: Building collections of LR(k) items with partial expansion of lookahead strings. ACM SIGPLAN Notices **17** (4), 25-28 (1982)

4. Barth, G.: Fast recognition of context-sensitive structures. Computing **22,** 243-256 (1979)

5. Buttelmann, H.W.: On the syntactic structures of unrestricted grammars, I & II. Information and Control **29,** 29-101 (1975)

6. DeRemer, F.L.: Practical translators for LR(k) languages. MIT, Cambridge, Mass.. Project MAC Report TR-65, 1969

7. Earley, J.: An efficient context-free parsing algorithm. Communications of the ACM **13,** 94-102 (1970)

8. Fisher, A.J.: Practical LL(1)-based parsing of van Wijngaarden grammars. Acta Informatica **21,** 559-584 (1985)

9. Griffiths, T.V.: Some remarks on derivations in general rewriting systems. Information and Control **12,** 27-54 (1968)

10. Knuth, D.E.: On the translation of languages from left to right. Information and Control **8,** 607-639 (1965)

11. Kunze, M.: Grammars for efficient non-context-free parsing. Conference on Information Sciences and Systems, University of Patras, Greece, 1979 (D.G. Lainiotis and N.S. Tzannes, eds.), Applications of Information and Control Systems. pp.217-225. Dordrecht-Boston-London: D. Reidel 1980

12. Loeckx, J.: The parsing for general phrase-structure grammars. Information and Control **16,** 443-464 (1970)

13. Mayer, O.: On deterministic canonical bottom-up-parsing. Information and Control **43,** 280-303 (1979)

14. Mickunas, M.D., Lancaster, R.L., Schneider, V.B.: Transforming LR(k) grammars to LR(1), SLR(1), and (1,1) bounded right-context grammars. Journal of the ACM **23,** 511-553 (1976)

15. Revesz, G.E.: Introduction to formal languages. New York: McGraw-Hill 1983

16. Sebesta, R.W., Jones, N.D.: Parsers for indexed grammars. Int. J. of Computer and Information Sciences **7,** 345-359 (1978)

17. Turnbull, C.J.M., Lee, E.S.: Generalized deterministic left to right parsing. Acta Informatica **12,** 187-207 (1979)

18. Vold'man, G.Sh.: A parsing algorithm for context-sensitive grammars. Programming and Computer Software **7,** 302-307 (1981)

19. Walters, D.A.: Deterministic context-sensitive languages, I & II. Information and Control **17,** 14-61 (1970)

20. Wegner, L.M.: On parsing two-level grammars. Acta Informatica **14,** 175-193 (1980)

Mathematics Department
University of Kentucky
Lexington Kentucky 40506