UNIVERSITY OF WATERLOO COMPUTER SCIENCE DEPARTMENT
UNIVERSITY OF WATERLOO COMPUTER SCIENCE DEPARTMENT
UNIVERSITY OF WATERLOO COMPUTER SCIENCE DEPARTMENT
UNIVERSITY OF WATERLOO COMPUTER SCIENCE DEPARTMENT

On The Relationship Between The LL(1) And LR(1) Grammars

John C. Beatty

CS-79-36

October 1979

ON THE RELATIONSHIP BETWEEN THE LL(1) AND LR(1) GRAMMARS

John C. Beatty

Department of Computer Science
University of Waterloo
Waterloo, Ontario, Canada N2L 3G1

ABSTRACT

It is shown that every p-reduced LL(1) grammar is LALR(1), and as a corollary that every Λ-free LL(1) grammar is SLR(1). A partial converse to this result is also demonstrated: if there is at most one marked rule in the basis of every state set in the canonical collection of sets of LR(k) items for a grammar G then G is LL(k).

ON THE RELATIONSHIP BETWEEN THE LL(1) AND LR(1) GRAMMARS*

John C. Beatty

Department of Computer Science
University of Waterloo
Waterloo, Ontario, Canada N2L 3G1

1. INTRODUCTION

The LL(1) and LALR(1) grammar classes are pre-eminent among the sub-context free classes for the practicality and usefulness of their parsing algorithms. Hence properties of these grammar classes, and especially their inter-relationships, are of particular interest.

Kral and Demner [10] established that exactly one production appears in the basis of any element of the canonical collection of LR(0) items for a Λ-free LL(1) grammar. We show that this property of Λ-free LL(1) grammars can be extended to any LL(1) grammar whose variables are all able to derive a non-null string (a *p-reduced* grammar). It is then possible to show that every such grammar is LALR(1). As an easy corollary we show that the Λ-free LL(1) grammars are all SLR(1), a result which was stated but not proven by Hunt and Szymanski [9]. We also exhibit a pathological grammar which demonstrates how LL(1) grammars containing variables which derive only Λ may fail to be LALR(1), although this is not always the case.

Finally we make use of our knowledge about the size of the LALR(1) basis sets for p-reduced LL(1) grammars to investigate the relative sizes of the LL(1) parser for a p-reduced LL(1) grammar G and the LALR(1) parser for G.

2. DEFINITIONS

For the standard notation, definitions and theory of context-free and LR(k) grammars the reader is referred to Harrison [6]. The remainder of this section introduces less familiar material and notation which we will need subsequently. In what follows let $G = (V,\Sigma,P,S)$ be a reduced cfg, let $N = V - \Sigma$, and let $\Sigma_\Lambda = \Sigma \bigcup \{\Lambda\}$.

The *size* $|G|$ of G is defined to be $\sum_{A \to \alpha \in P} \lg(A\alpha)$. A variable of G is said to be a *null variable* if it derives only the null word Λ; otherwise it derives at least one non-null string and is said to be *p-reduced*. G is said to be p-reduced if all the variables of G are p-reduced. The sets $\mathrm{first}_k(\beta)$ and $\mathrm{follow}_k(\beta)$ are defined for any $\beta \in V^*$ as

* Research partly supported by the U.S. Department of Energy under Contract No. W-7405-Eng-48 and by NSERC under grant A-3022.

$$\begin{aligned}
\mathrm{first}_k(\beta) = \{\, & w \in \Sigma^* \mid \\
& \lg(w) < k \text{ and } \beta \Rightarrow^* w \text{ or} \\
& \lg(w) = k \text{ and } \beta \Rightarrow^* wy \text{ for some } y \in \Sigma^* \,\}
\end{aligned}$$

$$\begin{aligned}
\mathrm{follow}_k(\beta) = \{\, & w \in \Sigma^* \mid \text{ for some } \alpha \in V^* \\
& \lg(w) < k \text{ and } S \Rightarrow^* \alpha\beta w \text{ or} \\
& \lg(w) = k \text{ and } S \Rightarrow^* \alpha\beta wy \text{ for some } y \in \Sigma^* \,\}
\end{aligned}$$

For a terminal string x we may write ${}^{(k)}x$ instead of $\mathrm{first}_k(x)$.

A cfg $G = (V,\Sigma,P,S)$ is LL(k) iff for any $A \in N$; $w, x, y \in \Sigma^*$; $\beta, \beta', \gamma \in V^*$; and any two derivations

$$S \Rightarrow_L^* wA\gamma \Rightarrow_L w\beta\gamma \Rightarrow_L^* wx$$

$$S \Rightarrow_L^* wA\gamma \Rightarrow_L w\beta'\gamma \Rightarrow_L^* wy$$

for which ${}^{(k)}x = {}^{(k)}y$ we necessarily have $\beta = \beta'$. G is SLL(k) iff for any $A \in N$; $w, w', x, y \in \Sigma^*$; $\beta, \beta', \gamma, \gamma' \in V^*$; and any two derivations

$$S \Rightarrow_L^* wA\gamma \Rightarrow_L w\beta\gamma \Rightarrow_L^* wx$$

$$S \Rightarrow_L^* w'A\gamma' \Rightarrow_L w'\beta'\gamma' \Rightarrow_L^* w'y$$

for which ${}^{(k)}x = {}^{(k)}y$ we necessarily have $\beta = \beta'$. This is equivalent to requiring for every variable A of G that

$$\mathrm{first}_k(\beta\,\mathrm{follow}_k(A)) \cap \mathrm{first}_k(\beta'\,\mathrm{follow}_k(A)) = \emptyset$$

for every distinct pair of rules $A \rightarrow \beta$ and $A \rightarrow \beta'$ in P. [2]

A cfg $G = (V, \Sigma, P, S)$ is LR(k) for some $k \geqslant 0$ iff $S \Rightarrow_R^+ S$ is impossible in G and for any $w, w', x \in \Sigma^*$; $\alpha, \alpha', \beta, \beta' \in V^*$; $A, A' \in N$; and derivations

$$S \Rightarrow_R^* \alpha A w \Rightarrow_R \alpha\beta w$$

$$S \Rightarrow_R^* \alpha' A' x \Rightarrow_R \alpha'\beta' x = \alpha\beta w'$$

if ${}^{(k)}w = {}^{(k)}w'$ then $(A \rightarrow \beta, |\alpha\beta|) = (A' \rightarrow \beta', |\alpha'\beta'|)$.

For an algorithm which computes the set $V_k(\gamma)$ of valid LR(k) items (also called *state sets*) for the viable prefix γ of G, and for definitions of the canonical collection S_k of sets of LR(k) items, of the function goto(s,X) where s is a state set and X is a grammar symbol, and of LR(k) consistency, the reader is again referred to Harrison. [6]

For an LR(k) state set s, the *basis* s_b of s is the set of LR(k) items in s in which the dot is preceded by at least one symbol, while the *closure* s_c of s is the set of items in s in which the dot is not preceded by a grammar symbol. These definitions are equivalent to those used by DeRemer [4].

The *core* of a set s of LR(k) items is the set of dotted first components (*marked rules*) of the items in that set. Thus the core of $\{ (A \rightarrow \alpha \cdot \beta, u) \}$ is $\{ (A \rightarrow \alpha \cdot \beta) \}$.

It will sometimes be convenient when discussing a set of items s to write $(A \rightarrow \alpha \cdot \beta, R)$, where R is the set of all the second components (*lookaheads*) of items in s having the core $A \rightarrow \alpha \cdot \beta$. If $R = \{u_1, \ldots, u_n\}$ we will also write $(A \rightarrow \alpha \cdot \beta, u_1 /\ldots/ u_n)$ rather than $(A \rightarrow \alpha \cdot \beta, \{u_1, \ldots, u_n\})$.

For $n \geqslant 1$ and $\alpha, \beta_i \in V^*$ the sequence of items $(A_0 \rightarrow \alpha \cdot A_1 \beta_0, a_0)$, $(A_1 \rightarrow \cdot A_2 \beta_1, a_1)$, $(A_2 \rightarrow \cdot A_3 \beta_2, a_2), \ldots, (A_n \rightarrow \cdot A_{n+1} \beta_n, a_n)$ in which $a_i \in \mathrm{first}_k(\beta_{i-1} a_{i-1})$, for $0 < i \leqslant n$ is a *chain*. It is a sequence of items which will cause $(A_n \rightarrow \cdot A_{n+1} \beta_n, a_n)$ to be added to the closure of a set of items containing $(A_0 \rightarrow \alpha \cdot A_1 \beta_0, a_0)$. There is an obvious leftmost derivation $A_1 \Rightarrow_L^+ A_{n+1} \beta_n \cdots \beta_1$ corresponding to each such chain which involves no Λ-rules, except perhaps the rule $A_n \rightarrow A_{n+1} \beta_n$.

The *LALR(k) state sets* for a grammar G are formed by first computing the LR(k) state sets and then merging together LR(k) state sets having identical cores. G is said to be an *LALR(k) grammar* iff every such LALR(k) state set for G is LR(k) consistent. A language is said to be *LALR(k)* iff it is generated by an LALR(k) grammar.

A cfg grammar is said to be *SLR(k)* iff every LR(0) state set s for G has the following property for $A, B \in N$; $\alpha, \beta_1, \beta_2 \in V^*$; ${}^{(1)}\beta_2 \notin N$: if $(A \rightarrow \alpha \cdot) \in s$, $(B \rightarrow \beta_1 \cdot \beta_2) \in s$ and

$$\mathrm{follow}_k(A) \bigcap \mathrm{first}_k(\beta_2\, \mathrm{follow}_k(B)) \neq \emptyset$$

then $(A \rightarrow \alpha \cdot) = (B \rightarrow \beta_1 \cdot \beta_2)$. If this condition fails for some LR(0) state set s of G then we say that s has an *SLR(k) conflict*, or is *SLR(k) inconsistent*. The conflict is said to be an *SLR(k) reduce/reduce* or *read/reduce* conflict depending on whether, for the two items in question, β_2 is Λ or a terminal symbol. A language is said to be *SLR(k)* iff it is generated by an SLR(k) grammar.

3. RESULTS.

We begin by verifying that LL(1) grammars containing Λ-rules may fail to be SLR(1).

Theorem 1: The family of LL(1) grammars is incomparable with the family of SLR(1) grammars.

Proof: The p-reduced grammar G given by

$$S \rightarrow 1X \mid 2Ag$$
$$X \rightarrow Af \mid Bg$$
$$A \rightarrow 3 \mid \Lambda$$

$B \rightarrow 4 \mid \Lambda$

is SLL(1) since

$$\begin{aligned}
&\text{first}_1(\ 1X\ \ \text{follow}_1(S)\ ) = \{\ 1\ \}\\
&\text{first}_1(\ 2Ag\ \ \text{follow}_1(S)\ ) = \{\ 2\ \}\\
&\text{first}_1(\ Af\ \ \text{follow}_1(X)\ ) = \{\ 3, f\ \}\\
&\text{first}_1(\ Bg\ \ \text{follow}_1(X)\ ) = \{\ 4, g\ \}\\
&\text{first}_1(\ 3\ \ \text{follow}_1(A)\ ) = \{\ 3\ \}\\
&\text{first}_1(\ \Lambda\ \ \text{follow}_1(A)\ ) = \{\ f, g\ \}\\
&\text{first}_1(\ 4\ \ \text{follow}_1(B)\ ) = \{\ 4\ \}\\
&\text{first}_1(\ \Lambda\ \ \text{follow}_1(B)\ ) = \{\ g\ \}
\end{aligned}$$

and is therefore LL(1) since every SLL(1) grammar is LL(1) [1].

The canonical collection of sets of LR(0) items for this grammar contains the state set

$$V_0(1) = \left\{ \begin{array}{l} (S \rightarrow 1 \cdot X) \\ (X \rightarrow \cdot Af) \\ (X \rightarrow \cdot Bg) \\ (A \rightarrow \cdot 3) \\ (A \rightarrow \cdot) \\ (B \rightarrow \cdot 4) \\ (B \rightarrow \cdot) \end{array} \right\}$$

which contains the items $(A \rightarrow \cdot)$ and $(B \rightarrow \cdot)$. Since the follow_1 sets for A and B are {f, g} and {g}, respectively, this state set violates the SLR(1) definition so that the grammar is not SLR(1).

On the other hand, the grammar G given by

$S \rightarrow Sa \mid a$

is trivially SLR(1), but not LL(1) since it is left recursive, which can never be the case for an LL(k) grammar [11]. **Q.E.D.**

The corresponding language question is well-known, in that every LL(1) language is LR(1) [1,2], and every LR(1) language (in fact every LR(k) language) can be generated by an SLR(1) grammar [2,5]. Since there are LR(k) languages which cannot be generated by any LL(k) grammar [11], it follows that the LL(1) languages are a proper subset of the SLR(1) languages.

However, we will be able to show that if an LL(1) grammar is Λ-free then in fact it is SLR(1) as well, and also that p-reduced grammars such as the grammar presented in Theorem 1 are necessarily LALR(1). We obtain these results by examining the structure of the canonical sets of items for an LL(1) grammar. The next lemma, in which we generalize a result due to Kral and Demner [10], is the key to these results. In understanding this lemma it may help to observe that we could assume $|\mathrm{core}(s_b)| = 1$ rather than $|\mathrm{core}(s_b)| \leqslant 1$ were it not for $V_1(\Lambda)$, whose basis set is by convention empty and is therefore of size 0.

Lemma 2. Let $G = (V, \Sigma, P, S)$ be a reduced LL(1) grammar. Let s be an element of S_1, the canonical collection of sets of LR(1) items for G. If $|\mathrm{core}(s_b)| \leqslant 1$ and $s' = \mathrm{goto}(s,X) \neq \varnothing$ for some p-reduced symbol X in V then $|\mathrm{core}(s_b')| = 1$.

Proof: Suppose that $|\mathrm{core}(s_b')| > 1$. In particular, s_b' must contain at least two items with distinct cores, which we represent by

$$I_A' = (A \rightarrow \alpha_1 X \cdot \alpha_2, R_A) \in s_b'$$

$$I_B' = (B \rightarrow \beta_1 X \cdot \beta_2, R_B) \in s_b'$$

for some $\alpha_1, \alpha_2, \beta_1, \beta_2 \in V^*$. It follows that s must contain items of the form

$$I_A = (A \rightarrow \alpha_1 \cdot X \alpha_2, R_A) \in s$$

$$I_B = (B \rightarrow \beta_1 \cdot X \beta_2, R_B) \in s$$

from which I_A' and I_B' are obtained.

Claim A: I_A and I_B both belong to the closure s_c of s.

Proof of Claim A: Suppose, for the sake of a contradiction, that the claim is not true. Assume first that $X \in \Sigma$. Since $|\mathrm{core}(s_b)| \leqslant 1$, one of I_A and I_B must belong to s_c. But this is impossible, for then in the only item belonging to s_b the dot precedes a terminal, so that s_c must be empty.

Therefore assume that $X \in N$. Again, since $|\mathrm{core}(s_b)| \leqslant 1$, at least one of I_A and I_B must belong to s_c. Without loss of generality suppose that $I_A \in s_b$ and $I_B \in s_c$. Then I_A and I_B have the form

$$I_A = (A \rightarrow \alpha_1 \cdot X \alpha_2, R_A)$$

$$I_B = (B \rightarrow \cdot X \beta_2, R_B)$$

and for some $m > 0$, $\gamma_i \in V^*$, $u_i \in \Sigma_\Lambda$ there exists a chain of items

$$J_0 = (A \rightarrow \alpha_1 \cdot X \gamma_0 \;, u_0)$$

$$
\begin{aligned}
J_1 &= (X \rightarrow \cdot C_1\gamma_1 \ , u_1) \\
J_2 &= (C_1 \rightarrow \cdot C_2\gamma_2 \ , u_2) \\
&\quad \cdots \\
J_{m-1} &= (C_{m-2} \rightarrow \cdot B\gamma_{m-1} \ , u_{m-1}) \\
J_m &= (B \rightarrow \cdot X\beta_2 \ , u_m)
\end{aligned}
$$

to which there corresponds the derivation $X =>^+ X\beta_2\gamma_{m-1} \cdots \gamma_1$. But this is impossible since G is LL(1) and therefore not left-recursive. □

Thus I_A and I_B both belong to s_c and must have the form

$$
\begin{aligned}
I_A &= (A \rightarrow \cdot X\alpha_2, R_A) \\
I_B &= (B \rightarrow \cdot X\beta_2, R_B)
\end{aligned}
$$

We will now select arbitrary elements of I_A and I_B and consider the chains $\mathcal{A}$ and $\mathcal{B}$ whereby they are added to s. The items which begin $\mathcal{A}$ and $\mathcal{B}$ must have the same core, since $|\mathrm{core}(s)| \leqslant 1$; however, we shall see that eventually the chains diverge, in that their respective cores differ, and the corresponding leftmost derivations will violate the LL(1) definition.

Let the basis of s be the set of items $I = (Y \rightarrow \sigma \cdot C_1\gamma_0, R_Y)$, and let

$$
\begin{aligned}
I_a &= (A \rightarrow \cdot X\alpha_2, a) \in I_A = (A \rightarrow \cdot X\alpha_2, R_A) \\
I_b &= (B \rightarrow \cdot X\beta_2, b) \in I_B = (B \rightarrow \cdot X\beta_2, R_B)
\end{aligned}
$$

I_a and I_b are arbitrary elements of I_A and I_B. Then for $\gamma_i, \delta_i \in V^*$; $u_i, v_i \in \Sigma_A$; $C_i, D_i \in N$, there must exist items

$$
\begin{aligned}
&(Y \rightarrow \sigma \cdot C_1\gamma_0, u_0) \in I = s_b \\
&(Y \rightarrow \sigma \cdot C_1\gamma_0, v_0) \in I = s_b
\end{aligned}
$$

and chains

$$
\begin{aligned}
\mathcal{A}_0 &= (Y \rightarrow \sigma \cdot C_1\gamma_0 \ , u_0) \\
\mathcal{A}_1 &= (C_1 \rightarrow \cdot C_2\gamma_1 \ , u_1) \\
\mathcal{A}_2 &= (C_2 \rightarrow \cdot C_3\gamma_2 \ , u_2) \\
&\quad \cdots \\
\mathcal{A}_{m-1} &= (C_{m-1} \rightarrow \cdot C_m\gamma_{m-1} \ , u_{m-1}) = (C_{m-1} \rightarrow \cdot A\gamma_{m-1} \ , u_{m-1}) \\
\mathcal{A}_m &= (C_m \rightarrow \cdot C_{m+1}\gamma_m \ , u_m) \quad = (A \rightarrow \cdot X\alpha_2 \ , a)
\end{aligned}
$$

and

$$
\begin{aligned}
\mathcal{B}_0 &= (Y \rightarrow \sigma \cdot D_1\delta_0 \ , v_0) = (Y \rightarrow \sigma \cdot C_1\gamma_0 \ , v_0) \\
\mathcal{B}_1 &= (D_1 \rightarrow \cdot D_2\gamma_1 \ , v_1)
\end{aligned}
$$

$$B_2 \quad = (D_2 \rightarrow \cdot D_3\delta_2 \quad , v_2)$$

$$\ldots$$

$$B_{n-1} = (D_{n-1} \rightarrow \cdot D_n\delta_{n-1} \quad , v_{n-1}) = (D_{n-1} \rightarrow \cdot B\delta_{n-1} \quad , v_{n-1})$$

$$B_n \quad = (D_n \rightarrow \cdot D_{n+1}\delta_n \quad , v_n) \qquad = (B \rightarrow \cdot X\beta_2 \quad , b)$$

of items, where m, n $\geqslant$ 1 by virtue of Claim A, $C_1 = D_1$ and $\gamma_0 = \delta_0$.

Claim B: The A and B chains must diverge. That is, there exists an index l such that

(1) $0 \leqslant l < \min(m,n)$

(2) $\text{core}(A_i) = \text{core}(B_i)$, $0 \leqslant i \leqslant l$

(3) $\text{core}(A_{l+1}) \neq \text{core}(B_{l+1})$

Proof of Claim B: Suppose not. Since $\text{core}(A_0) = \text{core}(B_0)$ there exists an index which is $< \min(m,n)$ that satisfies (2). We must show that the *largest* such index is $< \min(m,n)$. This follows immediately if m=n, since I_A and I_B are distinct. Suppose, without loss of generality, that $l = m < n$ and that

$$\text{core}(A_i) = \text{core}(B_i) , \ 0 \leqslant i \leqslant m$$

Let r = n−m. Then corresponding to chain A, and its continuation to form chain B, is the derivation

$$Y \Rightarrow^m \sigma X \gamma_m\gamma_{m-1} \cdots \gamma_0 = \sigma X \delta_m\delta_{m-1} \cdots \delta_0 = \sigma X \alpha_2\delta_{m-1} \cdots \delta_0$$

$$\Rightarrow^r \sigma X \delta_n \cdots \delta_0 = \sigma X \beta_2\delta_{n-1} \cdots \delta_0$$

which contains the left recursion $X \Rightarrow_L^+ X\beta_2\delta_{n-1} \cdots \delta_{m+1}$. Since G is LL(1) this cannot happen, and a satisfactory index l must exist. □

Next let $u \in L(\sigma)$. As a consequence of Claim B we know that corresponding to chains A and B are the leftmost derivations

$$Y \Rightarrow_L^* uC_{l+1}\tau$$

$$\Rightarrow_L uC_{l+2}\gamma_{l+1}\tau$$

$$\Rightarrow_L^* uX\alpha_2\gamma_{m-1} \cdots \gamma_{l+1}\tau$$

and

$$Y \Rightarrow_L^* uC_{l+1}\tau$$

$$\Rightarrow_L uD_{l+2}\delta_{l+1}\tau$$

$$\Rightarrow_L^* uX\beta_2\delta_{n-1} \cdots \delta_{l+1}\tau$$

where $C_{l+2}\gamma_{l+1} \neq D_{l+2}\delta_{l+1}$ and $\tau = \gamma_1 \cdots \gamma_0 = \delta_1 \cdots \delta_0$.

Since X is a p-reduced symbol either $X = a \in \Sigma$ or $X \in N$ and there exists a terminal a and string $v \in \Sigma^*$ such that $X \overset{*}{=>}_L av$. Then we have derivations

$$\begin{aligned} Y &\overset{*}{=>}_L uC_{l+1}\tau \\ &=>_L uC_{l+2}\gamma_{l+1}\tau \\ &\overset{*}{=>}_L uav\alpha_2\gamma_{m-1} \cdots \gamma_{l+1}\tau \\ Y &\overset{*}{=>}_L uC_{l+1}\tau \\ &=>_L uD_{l+2}\delta_{l+1}\tau \\ &\overset{*}{=>}_L uav\beta_2\delta_{n-1} \cdots \delta_{l+1}\tau \end{aligned}$$

where $C_{l+1} \rightarrow C_{l+2}\gamma_{l+1} \neq C_{l+1} \rightarrow D_{l+2}\delta_{l+1}$. Thus

$$\mathrm{first}_1(C_{l+2}\gamma_{l+1}\,\mathrm{follow}_1(C_{l+1})) \bigcap \mathrm{first}_1(D_{l+2}\delta_{l+1}\,\mathrm{follow}_1(C_{l+1})) \neq \varnothing$$

contradicting the fact that G is LL(1).
Therefore we cannot have two such distinct items I_A and I_B in s. Consequently $|\,\mathrm{core}(s_b')\,| = 1$, as desired. **Q.E.D.**

We then immediately apply this lemma in the obvious way to obtain the following essential fact.

Lemma 3: Let $G = (V, \Sigma, P, S)$ be a p-reduced LL(1) grammar and let $s \in \mathcal{S}_1$ be an LR(1) state set for G. Then $|\,\mathrm{core}(s_b)| \leqslant 1$.

Proof: Every such set s is $V_1(\gamma)$ for some $\gamma \in V^*$. The lemma is clearly true for $V_1(\Lambda)$, whose basis set is empty. A trivial induction, making use of Lemma 2, then suffices. **Q.E.D.**

Since the proof of Lemma 2 did not actually make use of the lookahead, we have also the following result.

Corollary 4: Let $G = (V, \Sigma, P, S)$ be a p-reduced LL(1) grammar and let $s \in \mathcal{S}_0$ be an LR(0) state set for G. Then $|\,\mathrm{core}(s_b)| \leqslant 1$.

It is easy to prove directly from the LL(1) definition that a Λ-free LL(1) grammar G generates a prefix-free language. From [7, Thm. 3.5] it follows that such a language is strict deterministic, and therefore LR(0) [8, Thm. 4.1]. However, using the above corollary we can prove the following somewhat stronger result.

Theorem 5: Let G = (V,Σ,P,S) be a Λ-free LL(1) grammar. Then G is LR(0).

Proof: Suppose that G were not LR(0). Then there is some LR(0) state s for G which is inconsistent. The inconsistency must involve a reduction.

If the reduce item in question is in s_b then we know from corollary 4 that it constitutes the entire basis. The closure of s must then be empty, so that there is no other item in s with which the reduce item might conflict.

But the reduce item cannot belong to the closure of s, for such an item must have a Λ-rule as its core, and G is Λ-free.

Therefore s cannot be inconsistent and the theorem is established. **Q.E.D.**

Parenthetically we note that it is not the case that every Λ-free LL(1) grammar is strict deterministic [7]. For example, it is easy to see that the grammar

$$S \rightarrow A$$
$$A \rightarrow B \mid y$$
$$B \rightarrow x$$

is LL(1) but not strict deterministic.

We are now ready to establish the conditions under which an LL(1) grammar is LALR(1). Our proof of the following theorem makes essential use of the fact that every LL(1) grammar is LR(1). For a good proof of this fact see [3].

Theorem 6: Let G = (V , Σ , P , S) be a p-reduced LL(1) grammar. Then G is LALR(1).

Proof: We know that G must be LR(1). Suppose, however, that G *fails* to be LALR(1). Then there exists an LALR(1) state set s, formed by the merger of LR(1) state sets $t_1, \ldots, t_h$ ($h \geqslant 2$), which contains a conflict.

It follows from Lemma 3 that $|\mathrm{core}(s_b)| \leqslant 1$. The conflict may be either (I) a read/reduce conflict or (II) a reduce/reduce conflict.

(I) Suppose that s has a read/reduce conflict. Then s contains two distinct s-items

$$I_a = (A \rightarrow \sigma \cdot a\alpha, b)$$
$$I_B = (B \rightarrow \beta \cdot, a)$$

for some $a \in \Sigma$ and $b \in \Sigma_\Lambda$. Since $|\mathrm{core}(s_b)| \leqslant 1$, at least one of these items must belong to s_c. However, if s_b contains either I_a or I_B, then it consists exactly of that item, and the closure must be empty, which is not the case. Hence both items belong to s_c. This implies that $\beta = \Lambda$, so that we have

$$I_a = (A \rightarrow \sigma \cdot a\alpha, b)$$

$$I_B = (B \rightarrow \cdot, a)$$

By definition every t_i, for $1 \leqslant i \leqslant h$, contains an item of the form $I_a = (A \rightarrow \sigma \cdot a\alpha, \cdots)$, albeit with distinct lookahead sets. $I_B = (B \rightarrow \cdot, a)$ must belong to at least one t_i, say t_n. But then t_n contains both I_a and I_B, so that the LR(1) state set t_n has a conflict. But this is impossible since G is LR(1). Therefore s has no read/reduce conflicts.

(II) Suppose that s has a reduce/reduce conflict. Since $|\text{core}(s_b)| \leqslant 1$, if the basis of s contains any reduce item, then the closure of s is empty and s consists of exactly one item, which is not the case. Therefore the reduce items which are in conflict must both belong to the closure of s. Hence we may represent them by

$$I_A = (A \rightarrow \cdot, R_A)$$

$$I_B = (B \rightarrow \cdot, R_B)$$

Since these items are in conflict, there is some $c \in \Sigma_A$ such that $c \in R_A$ and $c \in R_B$. We will be concerned henceforth with the items

$$I_a = (A \rightarrow \cdot\,, c)$$

$$I_b = (B \rightarrow \cdot\,, c)$$

Every LR(1) state set t_i contains an item with core $A \rightarrow \cdot$ and an item with core $B \rightarrow \cdot$. At least one state set t_A contains the item $(A \rightarrow \cdot\,, c)$, and at least one state set t_B contains the item $(B \rightarrow \cdot\,, c)$. Furthermore, t_A and t_B must be distinct LR(1) state sets – if they were not we would have an LR(1) state set for G which contained a conflict, which is impossible. Let

$$\{ (C \rightarrow \sigma \cdot X\tau\,, u_1/\ldots/u_r) \}$$

be the basis of t_A and let

$$\{ (C \rightarrow \sigma \cdot X\tau\,, v_1/\ldots/v_s) \}$$

be the basis of t_B. For $\tau = \alpha_0 = \beta_0$ let

$$b_A = (C \rightarrow \sigma \cdot X\alpha_0\,, u_i)$$

$$A_1 = (X \rightarrow \cdot A_1\alpha_1\,, a_1)$$

$$A_2 = (A_1 \rightarrow \cdot A_2\alpha_2\,, a_2)$$

$$\cdots$$

$$A_m = (A_{m-1} \rightarrow \cdot A\alpha_m\,, a_m)$$

$$I_a = (A \rightarrow \cdot\,, c)$$

be a chain of m+2 items ($m \geqslant 0$) leading from the basis of t_A to I_a and let

$$b_B = (C \rightarrow \sigma \cdot X\beta_0\,, v_j)$$

$$B_1 = (X \rightarrow \cdot B_1\beta_1\,, b_1)$$

$B_2 = (B_1 \rightarrow \cdot B_2\beta_2 \,,\, b_2)$

$\cdots$

$B_n = (B_{n-1} \rightarrow \cdot B\beta_n \,,\, b_n)$

$I_b = (B \rightarrow \cdot \,,\, c)$

be a chain of n+2 items ($n \geqslant 0$) leading from the basis of t_B to I_b. Consider the A chain. Either $c = u_i$ for some i ($1 \leqslant i \leqslant r$) and $\alpha_m \cdots \alpha_0 =>^* \Lambda$, or $c \in \mathrm{first}_1(\alpha_m \cdots \alpha_0)$.

In the latter case there must exist an l, $1 \leqslant l \leqslant m$, such that $c \in \mathrm{first}_1(\alpha_l)$ and $\alpha_m \cdots \alpha_{l+1} =>^* \Lambda$. But then it follows from the algorithm for computing the closure of t_B that the chain

$b_B = (C \rightarrow \sigma \cdot X\alpha_0 \,,\, v_j)$

$(X \rightarrow \cdot A_1\alpha_1 \,,\, a_1)$

$(A_1 \rightarrow \cdot A_2\alpha_2 \,,\, a_2)$

$\cdots$

$(A_{l-1} \rightarrow \cdot A_l\alpha_l \,,\, a_l)$

$(A_l \rightarrow \cdot A_{l+1}\alpha_{l+1} \,,\, c)$

$\cdots$

$(A_{m-1} \rightarrow \cdot A\alpha_m \,,\, c)$

$I_a = (A \rightarrow \cdot \,,\, c)$

will place I_a in t_B, which is not the case. Therefore $c = u_i$ for some i in the range $1 \leqslant i \leqslant r$ and $\alpha_m \cdots \alpha_0 =>^* \Lambda$.

An entirely symmetric argument ensures that $c = v_j$ for some j in the range $1 \leqslant j \leqslant s$ and $\beta_n \cdots \beta_0 =>^* \Lambda$.

But then the chain

$b_B = (C \rightarrow \sigma \cdot X\alpha_0 \,,\, c)$

$A_1 = (X \rightarrow \cdot A_1\alpha_1 \,,\, c)$

$A_2 = (A_1 \rightarrow \cdot A_2\alpha_2 \,,\, c)$

$\cdots$

$A_m = (A_{m-1} \rightarrow \cdot A\alpha_m \,,\, c)$

$I_a = (A \rightarrow \cdot \,,\, c)$

must place I_a in t_B, so that t_B contains both I_a and I_b, and is consequently inconsistent. But G is LL(1), hence LR(1), and t_B is an LR(1) state set for G, so that t_B *must* be consistent. It follows that s must be consistent.

Hence G is LALR(1). **Q.E.D.**

The obvious question to ask next is whether an LL(1) grammar which fails to be p-reduced must also fail to be LALR(1). The answer is trivial – the LL(1) grammar having rules $S \rightarrow Ax$ and $A \rightarrow \Lambda$ is not p-reduced, although it is LALR(1) and its basis sets all have size at most one.

Furthermore, the LL(1) grammar given by the rules $S \rightarrow Ax \mid Ay$ and $A \rightarrow \Lambda$ is also LALR(1) even though it has an LALR(1) state set $\{ (S \rightarrow A \cdot x, \Lambda), (S \rightarrow A \cdot y, \Lambda) \}$ whose basis set is comprised of two items. This state set is, of course, accessed by a null variable, and it also follows implicitly from previous arguments that an LL(1) grammar which fails to be LALR(1) must contain a null variable and have a state set the core of whose basis has size at least 2. We next demonstrate that such a grammar exists.

Theorem 7: The LL(1) grammar

$$
\begin{aligned}
S &\rightarrow aF \mid bG \\
F &\rightarrow Xc \mid Yd \\
G &\rightarrow Xd \mid Yc \\
X &\rightarrow IA \\
A &\rightarrow \Lambda \\
I &\rightarrow \Lambda \\
Y &\rightarrow IB \\
B &\rightarrow \Lambda
\end{aligned}
$$

is not LALR(1).

Proof: It is easy to verify that this grammar is SLL(1) and therefore LL(1). On the other hand, if we compute the canonical collection of sets of LR(1) items for this grammar, we obtain the two sets

$$
V_1(aI) = \left\{ \begin{array}{l} (X \rightarrow I \cdot A, c) \\ (Y \rightarrow I \cdot B, d) \\ (A \rightarrow \cdot, c) \\ (B \rightarrow \cdot, d) \end{array} \right\} \quad \text{and} \quad V_1(bI) = \left\{ \begin{array}{l} (X \rightarrow I \cdot A, d) \\ (Y \rightarrow I \cdot B, c) \\ (A \rightarrow \cdot, d) \\ (B \rightarrow \cdot, c) \end{array} \right\}
$$

which have the same core and are therefore merged to form the LALR(1) state set

$$
\left\{ \begin{array}{l} (X \rightarrow I \cdot A, c/d) \\ (Y \rightarrow I \cdot B, c/d) \\ (A \rightarrow \cdot, c/d) \\ (B \rightarrow \cdot, c/d) \end{array} \right\}
$$

which contains a reduce/reduce conflict. **Q.E.D.**

We may also enquire as to the reverse direction, namely whether every LALR(1) grammar whose basis sets have a core of size at most one is necesarily LL(1), and indeed this is true for all but an easily characterized set of LALR grammars. In fact we can establish this property for the LR(k) grammars in general. This is not really a surprising result, for if $|\,\mathrm{core}(s_b)| \leqslant 1$ for every LR(k) state set s of a grammar then an LR(k) parser "knows" which rule it will eventually parse at the time it begins reading the right hand side of the rule, and in an intuitive sense this is the property which distinguishes the LL(1) grammars.

Theorem 8: Let $G = (V,\Sigma,P,S)$ be a reduced LR(k) grammar such that for every LR(k) state set s of G $|\,\mathrm{core}(s_b)| \leqslant 1$, and in which Λ-free derivations of the form $S =>_L^+ S\gamma$ are impossible for every $\gamma \in V^*$. Then G is LL(k).

Proof: Let $G = (V,\Sigma,P,S)$ be a grammar satisfying the hypothesis but suppose the theorem to be false. Then it follows easily from the definition of an LL(k) grammar [2] that there exist two derivations

$$S =>_L^n wB\delta =>_L w\gamma_1\delta =>_L^* wx \qquad (1)$$

$$S =>_L^n wB\delta =>_L w\gamma_2\delta =>_L^* wy \qquad (2)$$

such that ${}^{(k)}x = {}^{(k)}y$ but $\gamma_1 \neq \gamma_2$, for some $w, x, y \in \Sigma^*$; $\gamma_1, \gamma_2, \delta \in V^*$; $B \in N$; $n \geqslant 0$. Let $A \rightarrow \beta B\delta'$ be the rule used in (1) and (2) to deposit the explicitly shown B in $wB\delta$, for some $A \in N$ and $\beta, \delta' \in V^*$. For some $w', w'' \in \Sigma^*$ and $\delta'' \in V^*$ we may write

$$S =>_L^* w'A\delta'' =>_L w'\beta B\delta'\delta'' =>_L^* w'w''B\delta'\delta'' = wB\delta$$

where $w = w'w''$ and $\delta = \delta'\delta''$. Then there exists [2] an $\alpha \in V^*$ and $x_1, x_2, x_3, y_1, y_2, y_3 \in \Sigma^*$ such that

$$S =>_R^* \alpha Ax_3 =>_R \alpha\beta B\delta' x_3 =>_R^* \alpha\beta Bx_2x_3 =>_R \alpha\beta\gamma_1 x_2x_3$$
$$=>_R^* \alpha\beta x_1x_2x_3 = \alpha\beta x =>_R^* wx$$
$$S =>_R \alpha Ay_3 =>_R \alpha\beta B\delta' y_3 =>_R^* \alpha\beta By_2y_3 =>_R \alpha\beta\gamma_2 y_2y_3$$
$$=>_R \alpha\beta y_1y_2y_3 = \alpha\beta y =>_R^* wy$$

where $\alpha =>^* w$, $B => \gamma_1 =>^* x_1$, $B => \gamma_2 =>^* y_1$, $\delta' =>^* x_2$, $\delta' =>^* y_2$, $\delta'' => x_3$, and $\delta'' =>^* y_3$. By definition the LR(k) items

$$I_1 = (A \rightarrow \beta\cdot B\delta' \,,\, {}^{(k)}x_3)$$
$$I_2 = (A \rightarrow \beta\cdot B\delta' \,,\, {}^{(k)}y_3)$$
$$I_3 = (B \rightarrow \cdot\gamma_1 \,,\, {}^{(k)}(x_2x_3))$$
$$I_4 = (B \rightarrow \cdot\gamma_2 \,,\, {}^{(k)}(y_2y_3))$$

are all valid for the viable prefix $\alpha\beta$, and therefore belong to the set of items

$V_k(\alpha\beta)$. There are two cases:

(I) $\beta \neq \Lambda$, in which case the core of the basis of $V_k(\alpha\beta)$ is exactly $A \to \beta \cdot B\delta'$. Because $B => \gamma_1 =>^* x_1$ and $B => \gamma_2 =>^* y_1$, there exist chains $I_1, I_3 = J_0, J_1, \ldots, J_m$ and $I_2, I_4 = K_0, K_1, \ldots, K_n$ from I_3 and I_4 which terminate either in a reduce item whose core is a Λ-rule and whose lookahead is ${}^{(k)}x = {}^{(k)}y$, or in a shift item whose lookahead component is again ${}^{(k)}x = {}^{(k)}y$. Let us represent these chains in the following way.

$$\begin{array}{ll}
(A \to \beta \cdot B\delta', {}^{(k)}x_3) & (A \to \beta \cdot B\delta', {}^{(k)}y_3) \\
(B \to \cdot\gamma_1, {}^{(k)}(x_2x_3)) = & (B \to \cdot\gamma_2, {}^{(k)}(y_2y_3)) = \\
J_0 = (U_0 \to \cdot U_1\mu_1, u_1) & K_0 = (V_0 \to \cdot V_1\nu_1, v_1) \\
J_1 = (U_1 \to \cdot U_2\mu_2, u_1) & K_1 = (V_1 \to \cdot V_2\nu_2, v_2) \\
\ldots & \ldots \\
J_m = (U_m \to \cdot a\mu_{m+1}, u_{m+1}) & K_n = (V_n \to \cdot b\nu_{n+1}, v_{n+1})
\end{array}$$

where $a, b \in \Sigma_\Lambda$; if $a = \Lambda$ then $\mu_{m+1} = \Lambda$; if $b = \Lambda$ then $\nu_{n+1} = \Lambda$; for $0 \leqslant i \leqslant m$, $\mu_i \in V^*$, $u_i \in \Sigma^*$, $U_i \in N$; for $0 \leqslant i \leqslant n$, $\nu_i \in V^*$, $v_i \in \Sigma^*$, $V_i \in N$; ${}^{(k)}x = \mathrm{first}_k(a\mu_{m+1}u_{m+1}) = \mathrm{first}_k(b\nu_{n+1}v_{n+1}) = {}^{(k)}y$. From this last fact it follows that if $a \neq \Lambda \neq b$ then $a = b$.

Claim: $\mathrm{Core}(J_{m-i}) = \mathrm{core}(K_{n-i})$, $0 \leqslant i \leqslant \min(m,n)$.

Proof of Claim: The proof is by induction on i.

Basis. Consider the case in which $i=0$. If $a\mu_{m+1} = b\nu_{n+1} = \Lambda$ then we must have $U_m \to \cdot = V_n \to \cdot$, as otherwise $V_k(\alpha\beta)$ has a reduce-reduce conflict. Similarly, if $a \in \Sigma$ and $b\nu_{n+1} = \Lambda$ or $b \in \Sigma$ and $a\mu_{m+1} = \Lambda$ then $V_k(\alpha\beta)$ has a shift-reduce conflict, so this cannot happen. Finally if $a \neq \Lambda \neq b$, then $a = b$ and we must have $U_m \to \cdot a\mu_{m+1} = V_n \to \cdot b\nu_{n+1}$ since otherwise the core of the basis of $V_k(\alpha\beta a)$ would contain two distinct items, which is prohibited by assumption.

Induction Step: If $J_{m-i} = K_{n-i}$ for $0 \leqslant i \leqslant \min(m,n)$ then in particular $U_{m-i} = V_{n-i}$. It then follows that $\mathrm{core}(J_{m-i-1}) = \mathrm{core}(K_{n-i-1})$, since otherwise $V_k(\alpha\beta U_{m-i})$ would contain two distinct items $(U_{m-i-1} \to \cdot U_{m-i}\mu_{m-i}, u_{m-i})$ and $(V_{n-i-1} \to \cdot U_{m-i}\nu_{n-i}, v_{n-i})$. □

Without loss of generality we may assume that $m \leqslant n$. Furthermore we may not have $m = n$, for in that case the claim requires that we have $\gamma_1 = \gamma_2$, which is not the case. If, on the other hand, we have $m < n$, then for $j = n-m$ we have $I_3 = J_0 = (V_j \to \cdot V_{j+1}\nu_{j+1}, v_{j+1})$. It follows that $(V_{j-1} \to \cdot V_j\nu_j, v_j)$ may be written $(V_{j-1} \to \cdot B\nu_j, v_j)$, whose core clearly differs from $(A \to \beta \cdot B\delta', {}^{(k)}y_3)$ since $\beta \neq \Lambda$. But then the core of the basis of $V_k(\alpha\beta B)$ is too large. Thus it is not possible to have $\beta \neq \Lambda$.

(II) $\beta = \Lambda$. Then for some rule $X \to \sigma Y\tau$, where $\sigma, \tau \in V^*$ and $Y \in N$, the basis of $V_k(\alpha\beta) = V_k(\alpha)$ consists entirely of items having the core $X \to \sigma \cdot Y\tau$ and there exist chains

$$
\begin{array}{llll}
J_0 & = (X \to \sigma \cdot Y\tau, u_0) & K_0 & = (X \to \sigma \cdot Y\tau, v_0) \\
J_1 & = (Y \to \cdot U_1\mu_1, u_1) & K_1 & = (Y \to \cdot V_1\nu_1, v_1) \\
\dots & & \dots & \\
J_p & = (U_p \to \cdot U_{p+1}\mu_{p+1}, u_{p+1}) & K_q & = (V_q \to \cdot V_{q+1}\nu_{q+1}, v_{q+1}) \\
 & = (A \to \cdot B\delta', {}^{(k)}x_3) = I_1 & & = (A \to \cdot B\delta', {}^{(k)}y_3) = I_2 \\
J_{p+1} & = (U_{p+1} \to \cdot U_{p+2}\mu_{p+2}, u_{p+2}) & K_{q+1} & = (V_{q+1} \to \cdot V_{q+2}\nu_{q+2}, v_{q+2}) \\
 & = (B \to \cdot \gamma_1, {}^{(k)}(x_2x_3)) & & = (B \to \cdot \gamma_2, {}^{(k)}(y_2y_3)) \\
\dots & & \dots & \\
J_m & = (U_m \to \cdot a\mu_{m+1}, u_{m+1}) & K_n & = (V_n \to \cdot b\nu_{n+1}, v_{n+1})
\end{array}
$$

where $p, q \geqslant 0$; $a, b \in \Sigma_\Lambda$; if $a = \Lambda$ then $\mu_{m+1} = \Lambda$; if $b = \Lambda$ then $\nu_{n+1} = \Lambda$; $\mu_i \in V^*$, $u_i \in \Sigma^*$, $U_i \in N$ for $0 \leqslant i \leqslant m$; $\nu_i \in V^*$, $V_i \in \Sigma^*$, $V_i \in N$ for $0 \leqslant i \leqslant n$; ${}^{(k)}x = \text{first}_k(a\mu_{m+1}u_{m+1}) = \text{first}_k(b\nu_{n+1}v_{n+1}) = {}^{(k)}y$. Again it follows that if $a \neq \Lambda \neq b$ then $a = b$. Note that it is possible to have $p = 0$ and/or $q = 0$, with $\sigma = \Lambda$. This is the case in which $V_k(\alpha) = V_k(\Lambda)$, and it will require special attention.

As in (I) we can establish that $\text{core}(J_{m-i}) = \text{core}(K_{m-i})$, $0 \leqslant i \leqslant \min(m,n)$. Let $p' = m-p$ and $q' = n-q$, and suppose that $p' = q'$. Then $\text{core}(J_{p+1}) = \text{core}(K_{q+1})$ which requires that $\gamma_1 = \gamma_2$. Since we have assumed this not to be the case we must have $p' \neq q'$. Without loss of generality we may assume that $p' < q'$. Then in the K chain we have the items $K_p' = (A \to \cdot B\delta', v_{p+1})$ and $K_q' = (A \to \cdot B\delta', v_{q+1})$, so that at least one core is repeated in the K chain. Let K_i be the first item in the chain containing a core which is repeated and let K_j be the first item in which this core is repeated. (This represents the first instance of a left recursion in the k-chain). There are 3 subcases.

(a) $i = 0$. Then we must have $\sigma = \Lambda$ and consequently $\alpha = \Lambda$ (since otherwise K_0 must be in the closure of $V_k(\alpha\sigma)$), so that $V_k(\alpha\beta) = V_k(\Lambda)$. Then $V_i = V_j = S$ and $\text{core}(K_{j-1}) = V_{j-1} \to \cdot S\nu_j$, so that for some $\gamma \in V^*$ there exists a Λ-free derivation $S \Rightarrow_L^+ S\gamma$. Since this is prohibited by assumption, this case cannot occur.

(b) $i = 1$. Then $K_{i-1} = K_0 = (X \to \sigma \cdot V_1\tau, v_0)$ and $K_{j-1} = (V_{j-1} \to \cdot V_1\nu_j, v_j)$ have distinct cores, whence the basis of the core of $V_k(\alpha\beta V_1)$ has size at least two, which is prohibited.

(c) $i > 1$. Then $K_{i-1} = (V_{i-1} \to \cdot V_i\nu_i, v_i)$ and $K_{j-1} = (V_{j-1} \to \cdot V_i\nu_j, v_j)$ have distinct cores and again the basis of the core of $V_k(\alpha\beta V_i)$ has size at least two, which is prohibited.

It follows, then, that two such distinct derivations as (1) and (2) cannot exist, so that G is LL(k), as desird. **Q.E.D.**

It is necessary in Theorem 8 to assume that $S \Rightarrow_L^+ S\gamma$ is impossible, as is evident from the trivial grammar LR(0) grammar $S \rightarrow Sa \mid a$, which otherwise satisfies the hypothesis of Theorem 8 but fails to be LL since it is left recursive.

We might next logically enquire whether our results for the LL(1) grammars can be extended to larger values of k. Unfortunately that is not possible. For k > 1 the relationship between LL(k) and LALR(k) grammars (and by implication between the LL(k) and SLR(k) grammars, since every SLR(k) grammar is LALR(k)) is fixed by the following result:

Theorem 9: The following Λ-free LL(k) grammar is not LALR(k) for any $k \geqslant 2$.

$$\begin{aligned}
S &\rightarrow aF \mid bG \\
F &\rightarrow Xc \mid Yd \\
G &\rightarrow Xd \mid Yc \\
X &\rightarrow w \\
Y &\rightarrow w
\end{aligned}$$

Proof: It is not hard to verify that the above grammar is SLL(k) and therefore LL(k) for any $k \geqslant 2$. However, if we compute the LR(k) state sets for this grammar we obtain (among others) the sets

$$V_k(aw) = \left\{ \begin{matrix} (X \rightarrow w\cdot, c) \\ (Y \rightarrow w\cdot, d) \end{matrix} \right\} \qquad V_k(bw) = \left\{ \begin{matrix} (X \rightarrow w\cdot, d) \\ (Y \rightarrow w\cdot, c) \end{matrix} \right\}$$

Since these sets have the same core they are merged to form the LALR(1) state set $\{ (X \rightarrow w\cdot, c/d), (Y \rightarrow w\cdot, c/d) \}$, which is inconsistent. **Q.E.D.**

We can use Lemma 3 and Theorem 6 to investigate the relative sizes of LL(1) and LALR(1) parsers for a p-reduced LL(1) grammar, much as Kral and Demner did [10]. For a cfg G let us denote by r_G the sum of the lengths of the right hand sides of the rules of G.

Theorem 10: Let $G = (V, \Sigma, P, S)$ be a p-reduced LL(1) grammar. Then the number of LALR(1) state sets for G is $(1+r_G)$.

Proof: Let $A \rightarrow \alpha X \beta$ be an arbitrary non-Λ-rule of G, where $\alpha, \beta \in V^*$ and $X \in V$. Since G is reduced there exists a derivation $S \Rightarrow_R^* \sigma A w \Rightarrow_R \sigma\alpha X\beta w$ for some $\sigma \in V^*$ and $w \in \Sigma^*$. Then the LR(1) state set $s = V_1(\sigma\alpha X)$ must contain an item $(A \rightarrow \alpha X \cdot \beta, R)$ for some lookahead set R, since this item is valid for $\sigma\alpha X$. Since $X \neq \Lambda$, $(A \rightarrow \alpha X \cdot \beta, R)$ is an element of the basis of s. Since every LALR(1) state set has exactly one item in its basis, this means that there are at least r_G LALR(1) state sets which are accessed by a non-null symbol. There is exactly one LALR(1) state set which is *not* accessed by some non-null symbol X, namely $V_1(\Lambda)$, so that there are a total of at least $(1+r_G)$ state sets.

By definition there cannot be two LALR(1) state sets whose basis sets have the same core. Hence there are exactly $(1+r_G)$ LALR(1) state sets. **Q.E.D.**

Corollary 11: Let $G = (V, \Sigma, P, S)$ be a Λ-free LL(1) grammar. Then the number of LR(0) state sets for G is $(1+r_G)$.

We can now use Theorem 10 to compare the amount of data required for the standard representation of the LL(1) parser for an LL(1) grammar (see [1], for example) with the amount of data required for the standard representation of the LALR(1) parser for the same LL(1) grammar.

The primary data structure for an LL(1) parser [1] consists of an expansion function M(A,a) which, for $A \in N$ and 1-lookahead $a \in \Sigma_\Lambda$, yields the index of the rule for A which should be used to obtain the next left sentential form. The expansion function thus potentially requires $O(|N| \cdot |\Sigma_\Lambda|) = O(|G|^2)$ space (assuming that $|N|$ and $|\Sigma|$ are linearly proportional to $|G|$). Other information required for an LL(1) parser, such as the rules themelves, requires at most $O(|G|)$ space.

For an LALR(1) parser the storage needed is also $O(|G|)$, except for the tabulation of the parsing action and goto functions+ for each state, which can increase non-linearly with the size of the grammar (even though the number of states increases linearly, as we shall see).

Suppose that this data is stored in arrays and accessed by indexed lookup. Then the expanion function for a grammar would be stored in an array of dimension $(|N|, |\Sigma_\Lambda|)$, so that an LL(1) parser would require $O(|G|^2)$ storage. The LALR(1) parser would require $O(|T| \cdot |V|)$ space for the goto functions and $O(|T| \cdot |\Sigma_\Lambda|)$ space for the parsing action functions, where T is the collection of LALR(1) parsing tables for G, and $|T|$ is equal to the number of LALR(1) state sets for the grammar.

In view of Theorem 10, the storage required for both kinds of parsers is thus $O(|G|^2)$.

+ The parsing action function f(T,a) yields one of the actions shift, reduce and error as a function of the table (state set) T and lookahead $a \in \Sigma_\Lambda$. The goto or transition function g(T,X) yields the next table (state set) or error as a function of the current table and the grammar symbol $X \in V$. See [1] for details.

If we assume that the expansion, goto and parsing action functions are stored as lists, so as to squeeze out the error entries, then the story is somewhat different. Even though we know the *exact* number of LALR(1) state sets for a p-reduced LL(1) grammar, we can show that there are LL(1) grammars for which the LL(1) parser will be substantially smaller than the LALR(1) parser for the same grammar (in that fewer pieces of data will be required), and *vice – versa*.

Consider the family of grammars

$$
\begin{array}{lcl}
S & \rightarrow & A_1 \\
A_1 & \rightarrow & A_2 \\
A_2 & \rightarrow & A_3 \\
 & \cdots & \\
A_{n-1} & \rightarrow & A_n \\
A_n & \rightarrow & 1 \mid 2 \mid 3 \mid \cdots \mid n
\end{array}
$$

A particular grammar from this family has size 4n. There are (n+1) variables, and the expansion function for each such variable A_i (let $A_0 = S$) has n pairs (a,p), where $a \in \{1,\ldots,n\}$ and p is the rule $A_i \rightarrow A_{i+1}$ if $i < n$ and $A_i \rightarrow a$ if $i = n$, so that the total list storage required for the LL(1) parser is $O(|G|^2)$. The LALR(1) state sets for any member of this family look like

$$
\begin{array}{ll}
\begin{array}{c} \text{2n transitions} \\ \text{n lookahead strings} \end{array} & \\
\left\{ \begin{array}{l}
(S \rightarrow \cdot A_1 \, , \, \Lambda) \\
(A_1 \rightarrow \cdot A_2 \, , \, \Lambda) \\
\quad \cdots \\
(A_{n-2} \rightarrow \cdot A_{n-1} \, , \, \Lambda) \\
(A_{n-1} \rightarrow \cdot A_n \, , \, \Lambda) \\
(A_n \rightarrow \cdot 1 \, , \, \Lambda) \\
\quad \cdots \\
(A_n \rightarrow \cdot n \, , \, \Lambda)
\end{array} \right\}
&
\begin{array}{l}
\left. \begin{array}{l}
\{ (S \rightarrow A_1 \cdot \, , \, \Lambda) \} \\
\quad \cdots \\
\{ (A_{n-1} \rightarrow A_n \cdot \, , \, \Lambda) \}
\end{array} \right\} \begin{array}{l} \text{no transitions} \\ \text{n lookahead strings} \end{array} \\
\\
\left. \begin{array}{l}
\{ (A_1 \rightarrow 1 \cdot \, , \, \Lambda) \} \\
\{ (A_2 \rightarrow 2 \cdot \, , \, \Lambda) \} \\
\quad \cdots \\
\{ (A_n \rightarrow n \cdot \, , \, \Lambda) \}
\end{array} \right\} \begin{array}{l} \text{no transitions} \\ \text{n lookahead strings} \end{array}
\end{array}
\end{array}
$$

There are a total of 2n non-error goto's and n non-error lookahead strings (action entries), in the initial table (associated with the state set on the left above). The other tables contribute no non-error transitions and 2n non-error lookahead strings, so that the total storage required for the LALR(1) parser is $O(|G|)$, while the LL(1) parser required $O(|G|^2)$ space.

Consider, on the other hand, the family of grammars given by

$$
\begin{array}{lcll}
S & \rightarrow & A_1 A_1 \cdots A_1 A_1 & \quad (\text{n occurences of } A_1) \\
A_1 & \rightarrow & A_2 &
\end{array}
$$

$A_2 \rightarrow A_3$

$\cdots$

$A_{n-1} \rightarrow A_n$

$A_n \rightarrow 1$

A particular grammar in this family has size 3n+1. The expansion function for each variable A_i (again let $A_0 = S$) of any grammar in this family consists of the single element $(1,p_i)$, where p_i is the unique rule for A_i, so that the total storage required by an LL(1) parser for such a grammar is $O(|G|)$. However, the LALR(1) state sets for a grammar in this family are of the form

$$s_0 = \left\{ \begin{array}{l} (S \rightarrow \cdot A_1 \cdots A_n \, , \Lambda) \\ (A_1 \rightarrow \cdot A_2 \, , 1) \\ \quad \cdots \\ (A_n \rightarrow \cdot 1 \, , 1) \end{array} \right\} \quad \cdots \quad s_{n-1} = \left\{ \begin{array}{l} (S \rightarrow A_1 \cdots \cdot A_1 \, , \Lambda) \\ (A_1 \rightarrow \cdot A_2 \, , \Lambda) \\ \quad \cdots \\ (A_n \rightarrow \cdot 1 \, , \Lambda) \end{array} \right\}$$

$$s_n = \{(A_1 \rightarrow A_2 \cdot , 1/\Lambda)\} \qquad \cdots \qquad s_{2n-1} = \{(A_n \rightarrow 1 \cdot , 1/\Lambda)\}$$

$$s_{2n} = \{(S \rightarrow A_1 \cdots A_1 \cdot , \Lambda)\}$$

There are a total of n(n+1) non-error goto's and 3n+1 non-error lookahead strings, so that the total list storage required for the LALR(1) parser is $O(|G|^2)$, while the LL(1) parser required only $O(|G|)$ space.

ACKNOWLEDGEMENTS

The author is grateful for the suggestions of Professors M. A. Harrison and K. S. Booth, and for S. G. McCaulay's assistance in typesetting.

REFERENCES

[1] A. V. Aho and J. D. Ullman, *The Theory of Parsing, Translating, and Compiling,* Vols. I and II (Prentice-Hall 1972 and 1973).

[2] J. C. Beatty, Iteration Theorems for the LL(k) Languages, Ph.D. Thesis, University of California, Berkeley, California (1977). Available as UCRL - 52379 from the Technical Information Department, Lawrence Livermore Laboratory, Livermore, California.

[3] J. C. Beatty, Two Iteration Theorems for the LL(k) languages, to appear in *Theoretical Computer Science.*

[4] F. L. DeRemer, Simple LR(k) grammars, *CACM* 14:7 pp. 453-460 (July 1971).

[5] M. M. Geller, unpublished notes.

[6] M. A. Harrison, *Introduction to Formal Language Theory* (Addison-Wesley 1978).

[7] M. A. Harrison and I. M. Havel, Strict deterministic grammars, *JCSS* 7:3 pp. 237-277 (June 1973).

[8] M. A. Harrison and I. M. Havel, On the parsing of strict deterministic grammars, *JACM* 21:4 pp. 525-548 (October 1974).

[9] H. B. Hunt and T. G. Szymanski, Corregendum to lower bounds and reduction between grammar problems, *JACM* 25:4 pp. 687-688 (October 1978).

[10] J. Kral and J. Demner, A note on the number of states of the DeRemer recognizer, Institute for Computation Techniques of the Czech Technical University, Praha 2, Horska 3, Czechoslovakia (May 1972).

[11] D. J. Rosenkrantz and R. E. Stearns, Properties of deterministic top-down grammars, *Information and Control* 17:3 pp. 226-256.