Cleaned $SLL(1)$ Grammars are $SLR(1)$

Berthold Hoffmann

Universität Bremen, Germany

Abstract. We establish the following specific relation between context-free grammars that have $LL(k)$ and $LR(k)$ parsers, respectively: after cleaning an $SLL(1)$ grammar (by unfolding its empty productions), the resulting grammar is $SLR(1)$.

Keywords: context-free grammar, $SLL(1)$ parsing, $SLR(1)$ parsing

1 Introduction

In the theory of context-free grammars and languages, it is well-known for the deterministic parsing methods known as $LL(k)$ [6] and $LR(k)$ [5] that every $LL(k)$ grammar is $LR(k)$. This may raise the hope that analoguous results are known for their subclasses $SLL(k)$ [7] and $SLR(k)$ [1], but this is not the case. In particular, we are interested in a relation where the number k of lookahead symbols is equal to one. Our motivation comes from our recent work on deterministic graph parsing algorithms for hyperedge replacement grammars [4] (HR grammars, for short):

- *predictive top-down parsing* lifts $SLL(1)$ parsing to HR grammars [2], and
- *predictive shift-reduce parsing* lifts $SLR(1)$ parsing to HR grammars [3].

These parsing algorithms cannot easily be extended to a lookahead of $k > 1$ symbols. A relation between $SLL(1)$ and $SLR(1)$ grammars could be lifted to predictive top-down and predictive shift-reduce grammars generating languages of graphs that represent strings.

The rest of this paper ist structured as follows. The next section recalls context-free grammars and the "cleaning" of such grammars, by unfolding of empty productions. In Section 3 we define $SLL(k)$ and $SLR(k)$ grammars, and recall known relations between these grammars in Section 4, before we present, in Section 5, two theorems that are new to the best of our knowledge. Section 6 mentions implications of our results, and indicates future work.

2 Context-free Grammars

We recall context-free grammars with some notions needed for parsing, and define the removal of empty productions.

A *vocabulary* is a finite set of elements called symbols. Σ^* denotes the set of finite strings over Σ, where ε denotes the empty string. A subset $L \subseteq \Sigma^*$ is called a *language*.

Definition 1 (Context-free Grammar). A *context-free grammar* $G = (\Sigma, \mathcal{N}, \mathcal{P}, S)$ consists of a *vocabulary* Σ, *nonterminals* $\mathcal{N} \subseteq \Sigma$ (so that $T = \Sigma \backslash \mathcal{N}$ distinguishes *terminals*), a finite set $\mathcal{P} \subseteq \mathcal{N} \times \Sigma^*$ of *productions*, and a *startsymbol* $S \in \mathcal{N}$. For a production $r = (A, \alpha) \in \mathcal{P}$ and strings $\beta, \gamma \in \Sigma^*$, $\beta A \gamma \Rightarrow_{\mathcal{P}} \beta\alpha\gamma$ is a *derivation step* with r; $\Rightarrow^*_{\mathcal{P}}$ denotes the reflexive-transitive closure of this relation. The *language* of G is then defined as

$$\mathcal{L}(G) = \{w \in T^* \mid S \Rightarrow^*_{\mathcal{P}} w\}.$$

We denote a rule as $A \to \alpha$, and assume that the start symbol S occurs once on the left-hand side, and never on the right-hand side of a production. We also assume that G is *reduced*: for every production $p \in \mathcal{P}$, there is a word $w \in \mathcal{L}(G)$ such that $S \Rightarrow^*_{\mathcal{P}} \alpha \Rightarrow_p \beta \Rightarrow^*_{\mathcal{P}} w$.

A production $A \to \varepsilon$ is called *empty*, and a grammar G without empty productions is called *clean*.

Theorem 1. *For every context-free grammar G, there is a clean grammr G' generating the language $\mathcal{L}(G) \setminus \{\varepsilon\}$.*

Proof. Let $G = (\Sigma, \mathcal{N}, \mathcal{P}, S)$ be a context-free grammar. Construct the clean productions $\mathcal{P}'$ for $\mathcal{P}$ as follows.

1. Let $\mathcal{P}_0 := \mathcal{P}$. Set $i := 0$.
2. If $A \to \alpha B \beta \in \mathcal{P}_i$ with $B \to \varepsilon \in \mathcal{P}_i$, define $\mathcal{P}_{i+1} := \mathcal{P}_i \cup \{A \to \alpha\beta\}$.
3. Set $i := i + 1$, and repeat step 2 until $\mathcal{P}_{i+1} = \mathcal{P}_i$.
4. Define $\mathcal{P}' = \{S' \to S\} \cup \mathcal{P}_i \setminus \{A \to \varepsilon \mid A \in \mathcal{N}\}$ where S' does not occur in $\mathcal{P}$, and $G' = (\Sigma, \mathcal{N}, \mathcal{P}', S')$.

It is easy to see that $\mathcal{L}(G') = \mathcal{L}(G) \setminus \{\varepsilon\}$. □

3 *SLL(k)* and *SLR(k)* Grammars

We recall the definitions of $SLL(k)$ and $SLR(k)$ grammars, which use the notion of *first* and *follower* symbol sets.

Definition 2 (First and Follow Sets). For some $k > 0$,

- $\alpha|_k$ denotes the *k-prefix* of a word $\alpha \in \Sigma^*$, which consists the first k symbols of α, or equals α if w consists of les than k symbols;
- $L \cdot_k L' = \{w \cdot_k w' \mid w \in L, w' \in L'\}$ denotes the *k-prefix concatenation* of languages $L, L' \subseteq \Sigma^*$;
- the k *first* symbols $First_k(\alpha) = \{\beta|_k \mid \alpha \Rightarrow^*_{\mathcal{P}} \beta, \beta \in \Sigma^*\}$ are the set of k-prefixes generated from α;
- the k *followers* of a nonterminal A are given as

$$Follow_k(A) = \begin{cases} \bot^k & \text{if } A = S \\ \bigcup_{B \to \alpha A \beta \in \mathcal{P}} First_k(\beta) \cdot_k Follow_k(A) & \text{otherwise} \end{cases}$$

where we assume that the start symbol S is "followed" by a word of k terminal "end-of-text" symbols $\perp$ that do not occur in $\mathcal{P}$.[1]

A context-free grammar is $SLL(k)$ if for every nonterminal A, k symbols of lookahead allow to decide which productions for A will lead to a successful parse.

Definition 3 ($SLL(k)$ Grammar). A context-free grammar G as above is $SLL(k)$ if for every two productions $A \to \alpha$, $A \to \beta \in \mathcal{P}$ with $\alpha \neq \beta$,

$$\mathit{First}_k(\alpha) \cdot_k \mathit{Follow}_k(A) \cap \mathit{First}_k(\beta) \cdot_k \mathit{Follow}_k(A) = \varnothing. \tag{1}$$

A context-free grammar is $SLR(k)$ if it can be parsed with a deterministic bottom-up parser the actions of which are controled by a special finite state machine.

Definition 4 (Characteristic Finite State Machine). Let $G = (\Sigma, \mathcal{N}, \mathcal{P}, S)$ be a context-free grammar.

Then $I_G = \{A \to \alpha \centerdot \beta \mid A \to \alpha\beta \in \mathcal{P}\}$ is the set of *simple LR items* of G. An item $A \to \alpha \centerdot \beta$ is *initial* if $\alpha = \varepsilon$, and *reductive* if $\beta = \varepsilon$. For every set $I \subset I_G$ of items, the *closure* I^* is the least superset of I such that $B \to \centerdot\beta \in I^*$ whenever $A \to \alpha \centerdot B\alpha' \in I^*$ and $B \to \beta \in \mathcal{P}$.

The *characteristic finite state machine* $M = (\Sigma, Q, \Delta, q_0)$ (CFM[2], for short) of G consists of *states* Q that are represented by sets of items, with an *initial state* $q_0 \in Q$, a *transition relation* $\Delta \subseteq Q \times \Sigma \times Q$; it is constructed as follows.

1. The start state $q_0 \in Q$ is defined as the closure of the initial item $S \to \centerdot\alpha$ of the unique start rule $S \to \alpha \in \mathcal{P}$:

$$q_0 = \{S \to \centerdot\alpha\}^*.$$

2. Whenever some state $q \in Q$ contains an item $A \to \alpha \centerdot x\alpha'$ for some $x \in \Sigma$, Q contains a *successor-state* $q' = \{A \to \alpha x \centerdot \alpha' \mid A \to \alpha \centerdot x\alpha' \in q\}^*$, and Δ contains a transition (q, x, q').

Definition 5 ($SLR(k)$ Grammar). Let G be a context-free grammar with a CFM M as above.

In a state q of M, a reductive item $A \to \alpha\centerdot$ and some item $B \to \beta \centerdot \gamma$ *are in conflict* if

$$\mathit{Follow}_k(A) \cap (\mathit{First}_k(\gamma) \cdot_k \mathit{Follow}_k(B)) \neq \varnothing.$$

The conflict is called a *shift-reduce conflict* if $\gamma \neq \varepsilon$, and a *reduce-reduce conflict* otherwise. (The condition boils down to $\mathit{Follow}_k(A) \cap \mathit{Follow}_k(B) \neq \varnothing$ in the latter case.)

G is $SLR(k)$ if no state of M contains conflicting items.

[1] In the literature, first and follower sets consist only of terminal symbols. The inclusion of nonterminals into these sets is useful for Theorem 8, and does not change the notions of $SLL(k)$ and $SLR(k)$ grammars below.

[2] In the literature, the CFM if often called $LR(0)$ *automaton* as its consatruction does not involve any lookahead, in contrast to that of the CFM for $LR(k)$ gramars.

A special class of these grammars arises when there are no states with a potential for conflicts so that the lookahead is not needed at all.

Definition 6 ($LR(0)$ Grammar). A context-free grammar G is $LR(0)$ if reductive items do only occur as single kernel items of the states of their CFM. (Then, a reduction may never be in conflict with a shift or another reduction.)

4 Known Relations of $SLL(k)$ and $SLR(k)$ Grammars

Let us first recall the result for $LL(k)$ and $LR(k)$ languages and grammars mentioned in the introduction.

Theorem 2 ([7, Thm. 14]). *For all $k \geqslant 0$, every $LL(k)$ grammar is $LR(k)$.*

This result might raise the hope that similar relations hold for the subclasses of $SLL(k)$ and $SLR(k)$ grammars, but simple examples show that this is not the case.

Theorem 3 ([8, Thm. 8.60]). *For $k \geqslant 0$, the class of $SLL(k)$ grammars is incomparable with the class of $SLR(k)$ grammars.*

Proof (By counterexamples). On the one hand, the $SLL(1)$ grammar (with empty productions)

$$\begin{array}{llll} S \to S' & S' \to aA & S' \to bBd & A \to Be \quad A \to d \\ B \to b & B \to \varepsilon & C \to c & C \to \varepsilon \end{array}$$

is not $SLR(k)$ for any k.

On the other hand, a left-recursive grammar with rules like

$$S \to A \quad A \to Aa \quad A \to a$$

is not $LL(k)$, and thus not$SLL(k)$, for any k. □

The following results from the literature shed some light on the relation between $SLL(k)$ and $SLR(k)$ grammars.

Theorem 4 ([7], Thm. 2). *For an $LL(k)$ grammar, an equivalent $SLL(k)$ grammar can be effecively constructed.*

Theorem 5 ([7], Thm. 3). *For an $LL(k)$ grammar G, a clean $LL(k+1)$ grammar generating $\mathcal{L}(G) \setminus \{\varepsilon\}$ can be effecively constructed.*

This theorem shows that empty productions are not necessary for $LL(k)$ grammars, and by Theorem 4, this also holds for $SLL(k)$ grammars.

The construction of their CFMs makes obvious that every $SLR(k)$ grammar is $LR(k)$, but not vice versa.

Theorem 6 ([8, Thm. 6.59]). *Every $SLR(k)$ grammar is $LR(k)$, but not vice versa.*

Figure 1 depicts the relation between $LL(k)$, $SLL(k)$, $LR(k)$, and $SLR(k)$ grammars established by these results. More significantly, they disprove a relation between $SLL(k)$ and $SLR(k)$ grammars, even for "extended" k, let alone a relation between these classes for $k = 1$. Such a relation shall be established in the next section.

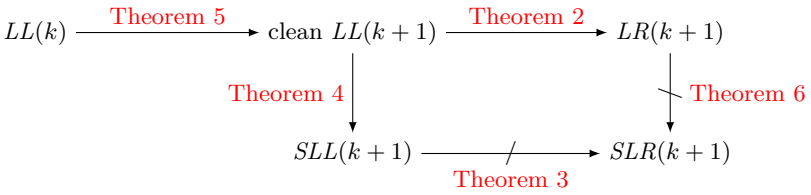

Fig. 1. Known relations between $SLL(k)$ and $SLR(k)$ grammars

5 New Relations between $SLL(1)$ and $SLR(1)$ Grammars

We use the observation that empty productions are relevant for the power of $SLL(k)$ to show a special relation for the case $k = 1$.

First show a property of CFMs of $SLL(k)$ grammars.

Lemma 1. *For an $SLL(1)$ grammar G, every state of its CFM has a single kernel item.*

Proof. G has exactly one start production $S \to \alpha$, which forms the only kernel item $S \to \centerdot\alpha$ of the start state q_0 of its CFM M.

Now consider an arbitrary state q with a single kernel item. By showing that every successor state of q has a single kernel item too, we prove the claim for all states.

Wlog., the kernel item of q has the form $A \to \alpha \centerdot \alpha'$. We inspect the form of α'.

1. If $\alpha' = \varepsilon$, q does not have a successor state.
2. If $\alpha' = b\beta$, q does not have closure items, and a single transition under b, to a successor state with the unique kernel item $A \to \alpha b \centerdot \beta$.
3. If $\alpha' = B\beta$, q has a transition under B to a state q' with kernel item $A \to \alpha B \centerdot \beta$. Assume that q contains a closure item of the form $B' \to \centerdot B\beta'$ so that q' contains another kernel item $B' \to B \centerdot \beta'$. The definition of closures implies that $B \Rightarrow_{\mathcal{P}}^{*} B'\beta \Rightarrow_{\mathcal{P}} B\beta'\beta$. But then, B is left-recursive, contradicting the assumption that G is $SLL(1)$.
   Now we consider the transitions introduced for an arbitrary closure item $C \to \centerdot\gamma$.
   (a) If $\gamma = \varepsilon$, the item does not introduce a transition.

(b) If $\gamma = d\delta$, q has a transition under d to a successor state q' with the kernel item $C \to d\,.\,\delta$. Assume that q contains another closure item $C' \to \,.\,d\delta'$ so that q' contains another kernel item $C' \to d\,.\,\delta'$. By definition of closures, $B \Rightarrow_{\mathcal{P}}^{*} C\beta \Rightarrow_{\mathcal{P}}^{*} c\delta\beta$, and $B \Rightarrow_{\mathcal{P}}^{*} C'\beta' \Rightarrow_{\mathcal{P}}^{*} c\delta'\beta'$. Then $c \in \mathit{First}_1(C) \cap \mathit{First}_1(D)$ by definition of first sets, contradicting the assumption that G is $\mathit{SLL}(1)$.

(c) If $\gamma = D\delta$, there is a transition under D to a state q' with a kernel item $C \to D\,.\,\delta$. Assume that q contains another closure item $C' \to \,.\,D\delta'$ so that q' contains another kernel item $C' \to D\,.\,\delta'$. By definition of closures, $B \Rightarrow_{\mathcal{P}}^{*} D\beta \Rightarrow_{\mathcal{P}}^{*} D\delta\beta$, and $B \Rightarrow_{\mathcal{P}}^{*} D\beta' \Rightarrow_{\mathcal{P}}^{*} D\delta'\beta'$. Then $\mathit{First}_1(D) \subseteq \mathit{First}_1(C) \cap \mathit{First}_1(D)$ by definition of first sets, contradicting the assumption that G is $\mathit{SLL}(1)$. □

Theorem 7. *A clean $\mathit{SLL}(1)$ grammar is $\mathit{LR}(0)$.*

Proof. Consider some state of the CFM, which has a unique kernel item that has the form $A \to \alpha\,.\,\alpha'$.

- If $\alpha' = \varepsilon$, the item is reductive, has no closure items, and thus no potential conflicts.
- If $\alpha' \neq \varepsilon$, the item may have closure items, but only the closure item of an empty production would be reductive, contradicting our assumption that G is clean. So there is no potential conflict either.
- The assumption also excludes the case that $\alpha = \alpha' = \varepsilon$, so we are done. □

Corollary 1. *For an $\mathit{SLL}(1)$ grammar G, all conflicts in their CFM arise from items of empty productions in the closures of states.*

Proof. By Theorem 7, conflicts may only arise from items of empty productions $A \to \varepsilon$. If this would be a kernel item, it has no closure items that could cause conflicts. (By construction of the CFM, this is the case only if $A = S$ and implies $\mathcal{L}(G) = \{\varepsilon\}$ due to our assumptions on G). However, items of the form $A \to \alpha\,.\,B\alpha'$ do have closure items, among them might be items of empty productions, which could raise conflicts. □

Only empty productions of $\mathit{SLL}(1)$ grammars may cause conflicts in their CFMs. Our main theorem shows that this is not the case for cleaned versions of $\mathit{SLL}(1)$ grammars.

Theorem 8. *The cleaned version of an $\mathit{SLL}(1)$ grammar is $\mathit{SLR}(1)$.*

Proof. We inspect the occurrences of initial items in the CFM M of G, the transitions induced by them, and show that the corresponding states in the CFM $\tilde{M}$ of the cleaned version $\tilde{G}$ of G are free of $\mathit{SLR}(1)$ conflicts.

Let $A \to \,.\,x_1 \dots x_k$ be an item in some state q of M (with $x_i \in \Sigma$, $1 \leqslant i \leqslant k$, $k \geqslant 0$). Then q has, for every $y \in \mathit{First}_1(x_1)$, a transitions under y to a state q_y, and a sequence of transitions under x_2 to x_k from q_{x_1} to states q_2 and further on to q_k. If $k = 0$, the item induces no transitions. Such an item, of an empty production, cannot occur in $\tilde{M}$. So we only have to distinguish two cases.

1. If no nonterminal of the x_i has an empty production, the item induces corrsponding transitions to corrsponding states $\tilde{q}_y$ and $\tilde{q}_i$ in $\tilde{M}$ (for $y \in \mathit{First}_1(x_1)$ and $2 \leqslant i \leqslant k$). Since these states do not contain items for empty productions, these states are free of conflicts.
2. Otherwise, assume that n nonterminals of the x_i have empty productions ($1 \leqslant n \leqslant k$). Then the corresponding state $\tilde{q}$ contains 2^n items, for the productions introduced by empty removal. Now let $x_i = B$ be the first nonterminal with an empty production (where $1 \leqslant i \leqslant k+1-n$). Then $\tilde{q}$ contains 2^n items starting with $\alpha = x_1 \dots x_{i-1}$; in one half of them, B does follow, in the other half, it does not.
   If x_1 is a terminal, it does not induce closure items, neither to q, nor to $\tilde{q}$. If x_i is a nonterminal, it induces the same closure items, both in q and in $\tilde{q}$ (x_i does not have an empty production that would not exist in $\tilde{q}$), and induces the same transitions for these closure items, to the same successor states. In q, the transition under x_1 leads to a state $q_{x_1} = q_2$ with a unique kernel item, whereas it leads to a state $\tilde{q}_2$ with 2^n kernel items in $\tilde{q}$.
   For the transitions under x_2 to x_{i-1}, the situation is as for x_1. Then we reach the state q_i that has a unique kernel item $A \to \alpha \,.\, B\alpha'$ in M, and a corresponding state $\tilde{q}_i$ with $m = 2^{n-1}$ pairs of kernel items $A \to \alpha \,.\, B\alpha'_j$, $A \to \alpha \,.\, \alpha'_j$ for $1 \leqslant j \leqslant m$. For every $z \in \mathit{First}_1(B)$, q_i has a transition under z to a state q_z, with a unique kernel item. For $z = B$, the kernel item of q_B is $A \to \alpha B \,.\, \alpha'$. In $\tilde{q}_i$, every $z \in \mathit{First}_1(B)$ has a transition under z to a state $\tilde{q}_z$. If $z \neq B$, q_z has a unique kernel item. For $z = B$, of $\tilde{q}_B$ has $m = 2^{n-1}$ kernel items $A \to \alpha B \,.\, \alpha'_j$ for $1 \leqslant j \leqslant m$. We consider the form of α'.
   (a) If $\alpha' = \varepsilon$, q_B is a reductive state, $n = m = 1$, and the two kernel items of $\tilde{q}_i$ are then $A \to \alpha \,.\, B$ and $A \to \alpha\,.$. Then $\mathit{First}_1(\varepsilon) \cdot_1 \mathit{Follow}_1(A) = \mathit{Follow}_1(A) \subseteq \mathit{Follow}_1(B)$. Since G is $SLL(1)$, the empty production for B in G implies that $\mathit{Follow}_1(B) \cap \mathit{First}_1(B) = \varnothing$ so that there is no shift-reduce conflict in $\tilde{q}_i$.
   (b) If $\alpha' = x_{i+1} \dots x_k$ with $i < k$, q_B has transitions, for every $z' \in \mathit{First}_1(x_{i+1})$, to successor states $q_{z'}$ and $\tilde{q}_i$ has further transitions, for every $z' \in \mathit{First}_1(x_{i+1})$, to successor states $\tilde{q}_{z'}$. Then $\mathit{First}_1(x_{i+1}) \subseteq \mathit{Follow}_1(B)$. Since G is $SLL(1)$, the empty production for B in G implies that $\mathit{Follow}_1(B) \cap \mathit{First}_1(B) = \varnothing$ so that the transitions induced by x_{i+1} go to states distinct to those induced by B. Hence the successor states in $\tilde{M}$ are distinct from one another, and have single kernel items.

Now consider the start states of the machines$M$ and $\tilde{M}$: The kernel item of q_0 ist $S \to \,.\alpha$. If $\alpha = \varepsilon$, G defines the languge $\{\varepsilon\}$ and $\tilde{G}$ defines the empty language. Otherwise $\tilde{q}_0$ has the kernel item $S' \to \,.S$. This state does not contain reductive items, and has no conflicts.

Thus $\tilde{M}$ is free of conflicts. □

6 Conclusions

In this note, we have established that the cleaned version of an $SLL(1)$ grammar is $SLR(1)$-parseable. To the best of our knowledge, this less important relation

has not been published before. Unfortunately, the proof of this result gives no clue how it can be genereralized to grammars with a lookahead of $k > 1$ symbols. The result does, however, help to establish a relation between subclasses of hyperedge replacement graph grammars [4] that generate languages of string graphs, and allow predictive top down parsing [2] and predictive shift-reduce parsing [3]. We even hope that the proof of our result might give a clue to extend the result to grammars that generate languages of "real" graphs, not only those that represent strings.

References

1. F. L. DeRemer. Simple LR(k) grammars. *Comm. of the ACM*, 14(7):453–460, 1971.
2. F. Drewes, B. Hoffmann, and M. Minas. Predictive top-down parsing for hyperedge replacement grammars. In F. Parisi-Presicce and B. Westfechtel, editors, *Graph Transformation - 8th International Conference, ICGT 2015. Proceedings*, volume 9151 of *Lecture Notes in Computer Science*, pages 19–34. Springer, 2015.
3. F. Drewes, B. Hoffmann, and M. Minas. Predictive shift-reduce parsing for hyperedge replacement grammars. In *International Conference for Graph Transformation 2017*, March 2017. Submitted.
4. A. Habel. *Hyperedge Replacement: Grammars and Languages*. Number 643 in Lecture Notes in Computer Science. Springer, 1992.
5. D. E. Knuth. On the translation of languages from left to right. *Information and Control*, 8(6):607 – 639, 1965.
6. P. M. Lewis II and R. E. Stearns. Syntax-directed transduction. *J. ACM*, 15(3):465–488, 1968.
7. D. J. Rosenkrantz and R. E. Stearns. Properties of deterministic top-down grammars. *Information and Control*, 17(3):226–256, 1970.
8. S. Sippu and E. Soisalon-Soininen. *Parsing Theory II: LR(k) and LL(k) Parsing*, volume 20 of *EATCS Monographs in Theoretical Computer Science*. Springer, 1990.