On the transformation of LL(k)-linear grammars to LL(1)-linear*

Alexander Okhotin and Ilya Olkhovsky

St. Petersburg State University,
7/9 Universitetskaya nab., Saint Petersburg 199034, Russia
alexander.okhotin@spbu.ru, ilianolhin@gmail.com

Abstract. It is proved that every LL(k)-linear grammar can be transformed to an equivalent LL(1)-linear grammar. The transformation incurs a blow-up in the number of nonterminal symbols by a factor of $m^{2k-O(1)}$, where m is the size of the alphabet. A close lower bound is established: for certain LL(k)-linear grammars with n nonterminal symbols, every equivalent LL(1)-linear grammar must have at least $n \cdot (m-1)^{2k-O(\log k)}$ nonterminal symbols.

1 Introduction

The LL(k) parsing is one of the most well-known linear-time parsing techniques. In this method, a parse tree of an input string is reconstructed top-down, along with reading the string from left to right. A parser selects each rule by looking ahead by at most k symbol. The family of *LL(k) grammars*, to which this algorithm is applicable, was introduced and systematically studied in the papers by Knuth [5], Lewis and Stearns [7] and Rozenkrantz and Stearns [9]. In particular, Kurki-Suonio [6] and, independently, Rozenkrantz and Stearns [9], proved that LL($k+1$) grammars are more powerful than LL(k) grammars, and thus there is a strict hierarchy of languages defined by LL(k) grammars, with different k.

An important subclass of LL(k) grammars, the *LL(k)-linear grammars*, was first studied by Ibarra et al. [3] and by Holzer and Lange [2], who proved that all languages defined by these grammars belong to the complexity class NC^1. Learning algorithms for LL(1)-linear grammars and related subclasses were studied by de la Higuera and Oncina [1], and language-theoretic properties of these grammars have recently been investigated by Jirásková and Klíma [4].

Whether LL(k)-linear grammars form a hierarchy with respect to the length of the look-ahead k, remains unexplored. The first contribution of this paper is a proof that every language defined by an LL(k)-linear grammar, for some k, is defined by an LL(1)-linear grammar; therefore, in the case of LL(k)-linear grammars, the hierarchy with respect to k collapses. The proof is constructive: it is shown how to transform any given LL(k)-linear grammar to a LL(1)-linear grammar that defines the same language.

* Research supported by RFBR grant 18-31-00118.

Next, it is shown that the proposed tranformation is close to being optimal in terms of the number of nonterminal symbols. The transformation of an LL(k)-linear grammar to an LL(1)-linear grammar increases the number of nonterminal symbols by a factor of $m^{2k-O(1)}$, where m is the size of the alphabet. A lower bound of $(m-1)^{2k-O(\log k)}$ on this factor is established.

2 Definitions

Definition 1. *A (formal) grammar is a quadruple $G = (\Sigma, N, R, S)$, where Σ is the* alphabet *of the language being defined, N is the set of syntactic categories defined in the grammar, known as* nonterminal symbols*; R is a finite set of rules, each of the form $A \to \alpha$, with $A \in N$ and $\alpha \in (\Sigma \cup N)^*$, and $S \in N$ is a nonterminal symbol representing all well-formed sentences in the language, known as the* initial symbol.

Each rule $A \to X_1 \dots X_\ell$ in R states that each string representable as a concatenation of ℓ substrings of the form $X_1, \dots, X_\ell$, therefore has the property A. This is formalized as follows.

Definition 2. *Let $G = (\Sigma, N, R, S)$ be a grammar. A parse tree is a rooted tree with leaves labelled with symbols from Σ, and with internal nodes labelled with nonterminal symbols from N. For each node labelled with $A \in N$, with its successors labelled with $X_1, \dots, X_\ell$, there must be a rule $A \to X_1 \dots X_\ell$ in the grammar. All successors are ordered, and if w is the string of symbols in the leaves, and A is the nonterminal symbol in the root, this is said to be a tree of w from A.*

The language defined by the grammar, denoted by $L(G)$, is the set of all strings $w \in \Sigma^*$, for which there exists a parse tree from S.

A grammar is called *linear*, if each rule in R is of the form $A \to uBv$, with $u, v \in \Sigma^*$ and $B \in N$, or of the form $A \to w$, with $w \in \Sigma^*$. A parse tree for a linear grammar is a path labelled with nonterminal symbols, with each rule $A \to uBv$ spawning off the leaves u to the left and v to the right.

A *top-down parser* attempts to construct a parse tree of an input string, while reading it from left to right. At every point of its computation, the parser's memory configuration is a pair (α, v), where v is the unread portion of the input string uv. The parser tries to parse v as a concatenation $\alpha = X_1 \dots X_\ell$, where $\ell \geqslant 0$ and $X_1, \dots, X_\ell \in \Sigma \cup N$. This sequence of symbols is stored in a stack, with X_1 as the top of the stack.

At each point of the computation, the parser sees the top symbol of the stack and the first k symbols of the unread input—the *look-ahead string*—where $k \geqslant 1$ is a constant. If there is a nonterminal symbol $A \in N$ at the top of the stack, the parser determines a rule $A \to \alpha$ for this symbol, pops this symbol, and pushes the right-hand side of the rule onto the stack.

$$(A\beta, v) \xrightarrow{A \to \alpha} (\alpha\beta, v)$$

Denote by $\Sigma^{\leqslant k}$ the set of strings of length at most k. Let $x \in \Sigma^{\leqslant k}$ be the first k unread input symbols; if this string is shorter than k, this indicates the end of the input approaching. The rule is chosen by accessing a look-up table $T_k \colon N \times \Sigma^{\leqslant k} \to R \cup \{-\}$, which contains either a rule to apply, or a marker indicating a syntax error.

If the top symbol of the stack is a symbol $a \in \Sigma$, the parser checks that the unread portion of the input begins with the same symbol, and then pops this symbol from the stack and reads it from the input.

$$(a\beta, av) \xrightarrow{\text{READ } a} (\beta, v)$$

For a string $w \in \Sigma^*$, denote its first k symbols, with $k \geqslant 0$, by

$$\mathrm{First}_k(w) = \begin{cases} w, & \text{if } |w| \leqslant k \\ \text{first } k \text{ symbols of } w, & \text{if } |w| > k \end{cases}$$

This definition is extended to languages as $\mathrm{First}_k(L) = \{\, \mathrm{First}_k(w) \mid w \in L \,\}$.

Definition 3. *Let $G = (\Sigma, N, R, S)$ be a grammar. A string $v \in \Sigma^*$ is said to follow $X \in \Sigma \cup N$, if there is a parse tree of some string with a suffix v, containing a subtree with a root X, to the left of which there is a string of leaves v. For all $A \in N$, denote* $\mathrm{Follow}(A) = \{\, v \mid v \text{ follows } A \,\}$.

Definition 4. *Let $k \geqslant 1$ and let $G = (\Sigma, N, R, S)$ be a grammar. An LL(k) table for G is a partial function $T_k \colon N \times \Sigma^{\leqslant k} \to R$ that satisfies the following condition: for all $A \in N$, $u, v \in \Sigma^*$ if, for every parse tree and for every subtree in that tree, if $A \in N$ is the label of its root and u is the string of its leaves, $A \to \alpha$ is the rule applied to A, and v follows A, then $T_k(A, \mathrm{First}_k(uv)) = A \to \alpha$.*

If such a table exists, then the grammar is said to be LL(k).

3 General plan of the transformation

The goal is to transform an arbitrary linear LL(k) grammar G to a linear LL(1) grammar G' that defines the same language. If there is a nonterminal symbol A in the original grammar, then choosing a rule for A requires knowing the next k symbols of the input. The general plan is to use a *buffer* for up to $k-1$ next input symbols, so that the parser reads them before having to choose a rule for A. In the new grammar, this buffer shall be attached to every nonterminal symbol, so that they are of the form ${}_uA$, with $A \in N$ and $u \in \Sigma^{\leqslant k-1}$. The goal is to have $L_{G'}({}_uA) = \{\, w \mid uw \in L_G(A) \,\}$.

Upon a closer inspection, there is a certain problem with this plan. If there is a rule $A \to s$ in the original grammar, with $s \in \Sigma^*$ and $|s| < k-1$, then, in order to choose a rule for A, an LL(1) parser needs to know *more symbols than there are in s and in its own 1-symbol lookahead.* If there are $k-1$ symbols in the buffer attached to A as a subscript, and this "short" rule $A \to s$ is to be applied, then what is this nonterminal symbol supposed to do with the surplus symbols in the buffer?

Example 1. The following grammar is linear LL(3).

$$S \to aabSa \mid a$$

In order to distinguish between these two rules, a hypothetical LL(1) parser buffers up to two first symbols using the following rules.

$$_{\varepsilon}S \to a_{a}S$$
$$_{a}S \to a_{aa}S$$

Once the parser has aa in the buffer and sees that the next symbol is b, it continues as planned.

$$_{aa}S \to b_{\varepsilon}Sa$$

However, if aa is in the buffer and the next symbol is a, then the parser realizes that the first symbol in its buffer should have been used in the rule $A \to a$, whereas the second a in its buffer should have been matched by the last symbol in some earlier rule $S \to aabSa$, and there seems to be no natural way to apply that rule retroactively.

The cause of this problem is a *short rule* that defines a substring of length less than $k-1$ in the middle of the input. Accordingly, the first step of the proposed transformation is to eliminate such rules.

4 Elimination of "short" rules

The first step in the transformation of a linear LL(k) grammar to a linear LL(1) grammar is the elimination of the so-called *short rules*, that is, rules of the form $A \to w$, with $w \in \Sigma^*$, $|w| < k-1$ and $\mathrm{Follow}(A) \neq \{\varepsilon\}$.

Lemma 1. *For every linear LL(k) grammar $G = (\Sigma, N, R, S)$ there exists a linear LL(k) grammar G' without short rules that defines the same language. The number of nonterminal symbols in G' is $|\Sigma^{\leqslant k-1}| \cdot |N|$.*

Proof. In the new grammar $G' = (\Sigma, N', R', S')$, nonterminals are of the form A_u, with $A \in N$ and $u \in \mathrm{Follow}_{k-1}(A)$. The goal is that every nonterminal A_u defines all strings defined by A in G, with a suffix u appended: $L_{G'}(A_u) = \{\, wu \mid w \in L_G(A) \,\}$.

For every nonterminal symbol A_u and for every rule for A in G, the new grammar has a rule defined as follows. For a rule $A \to w_1 B w_2 \in R$, let s denote the first $k-1$ symbols of $w_2 u$, so that $st = w_2 u$ with $|s| = \min(|w_2 u|, k-1)$. The corresponding rule in G' defers the string s to the nonterminal B, and appends the rest of the symbols in the end; these include all the remaining symbols of u.

$$A_u \to w_1 B_s t$$

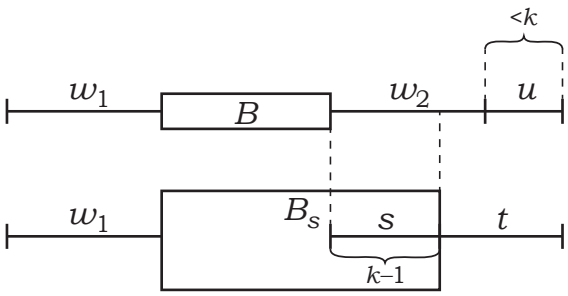

Fig. 1. How a rule $A \to w_1 B w_2$ in G is simulated by the rule $A_u \to w_1 B_s t$ in G'.

This is illustrated in Figure 1.

Once a rule $A \to s$ is reached, the corresponding rule in the new grammar appends the suffix to s.

$$A_u \to su$$

The correctness proof is comprised of several assertions: namely, that G' defines the desired language, has no short rules and is LL(k).

Claim. If a string w is defined by A_u in the new grammar, then $w = xu$ and A defines x in the original grammar.

Claim. If a string x is defined by A in the original grammar, then A_u defines xu in the new grammar.

Both claims are established by induction on the height of the respective parse trees. Together, the above two claims establish that $L_{G'}(A_u) = L_G(A)u$ for each nonterminal symbol A_u. From this, one can infer a similar correspondence between the languages defined by individual rules.

Claim. Let $A_u \to w_1 B_s t$ be a rule in G' obtained from a rule $A \to w_1 B w_2$ in G. Then $L_{G'}(w_1 B_s t) = L_G(w_1 B w_2)u$.

The proof that there are no short rules in the new grammar is given separately for $A_u \in N'$ with $|u| = k-1$ and with $|u| < k-1$.

Claim. For each nonterminal symbol $A_u \in N'$ with $|u| = k-1$, all strings defined by A_u are of length at least $k-1$.

This is proved by induction on the height of a parse tree of a string in A_u.

Next, assuming that $|u| < k-1$ and $y \in \mathrm{Follow}(A_u)$, one can infer that $y = \varepsilon$.

Claim. For each nonterminal symbol $A_u \in N'$ with $|u| < k-1$, the set $\mathrm{Follow}(A_u)$ equals $\{\varepsilon\}$.

This completes the proof that there are no short rules in the new grammar. The last claim is that the construction preserves the LL(k) property.

Claim. If the original grammar is LL(k), then so is the constructed grammar.

This completes the correctness proof for the short rule elimination construction. □

5 Reduction to one-symbol look-ahead

Once all short rules are eliminated, the following second construction reduces the length of the look-ahead strings to 1.

Lemma 2. *For every LL(k) linear grammar $G = (\Sigma, N, R, S)$ without short rules, there exists and can be effectively constructed an LL(1) linear grammar $G' = (\Sigma, N', R', {}_{\varepsilon}S)$, with $N' = \{ {}_{u}A \mid A \in N,\ u \in \Sigma^{\leqslant k-1} \}$, that describes the same language.*

Proof. In the new grammar G', nonterminal symbols are of the form ${}_{u}A$, with $A \in N$ and $u \in \Sigma^*$. The left subscript u of a nonterminal ${}_{u}A$ is a buffer for up to $k-1$ last symbols read by a parser. The goal is to have $L_{G'}({}_{u}A) = \{ w \mid uw \in L_G(A) \}$.

While the buffer is underfull, the parser reads extra symbols and appends them to the buffer. As soon as the buffer is filled, the parser sees a nonterminal symbol ${}_{u}A$ with $u \in \Sigma^{k-1}$, as well as a one-symbol look-ahead. Altogether, the parser has all k symbols needed to determine a rule to apply to A, which is given in the entry $T(A, u)$ in the LL(k) table for G. The buffer is updated along with simulating this rule: the first symbols of the rule for A are removed from the buffer, and all symbols remaining in the buffer are attached to the next nonterminal symbol, which is of the form ${}_{v}B$.

The initial symbol of the new grammar, ${}_{\varepsilon}S$, is S with an empty buffer.

There are three types of rules in the grammar G'. First, there are rules for filling the buffer. For each nonterminal ${}_{u}A$ with $|u| < k-1$, and for each symbol $a \in \Sigma$, there is a rule that appends this symbol to the buffer.

$$ {}_{u}A \to a\, {}_{ua}A $$

Second, there are rules for simulating the corresponding rules in G. For each ${}_{u}A \in N'$ and $a \in \Sigma$ with $|u| = k-1$ and with $T(A, ua)$ defined, the new grammar contains one rule defined as follows. If $T(A, ua) = A \to sBt$, then one of u, s is a prefix of the other; there are two cases, depending on which string is longer.

$$ {}_{u}A \to s'\, {}_{\varepsilon}Bt \qquad (s = us',\ \text{for } s' \in \Sigma^*) $$
$$ {}_{u}A \to {}_{v}Bt \qquad (u = sv,\ \text{for } v \in \Sigma^+) $$

If $T(A, ua) = A \to s$, then $s = ux$, and a is the first symbol of x if $x \neq \varepsilon$. Then the new grammar contains the following rule.

$$ {}_uA \to x $$

At last, there are rules for the case when the end of the input string is visible. Namely, for each ${}_uA \in N'$ with $|u| \leqslant k-1$ and with $T(A, u)$ defined, the grammar G' contains a null rule.

$$ {}_uA \to \varepsilon $$

The resulting grammar G' is linear. The proof that it is $LL(1)$ grammar and defines the same language as G is given in a series of claims.

Claim. If $x \in L_{G'}({}_uA)$, then $ux \in L_G(A)$.

Claim. If $ux \in L_G(A)$, then $x \in L_{G'}({}_uA)$.

In each direction, the proof is by induction on the height of the respective parse trees.

Claim. For each $u \in \Sigma^{\leqslant k-1}$, if $y \in \mathrm{Follow}({}_uA)$, then $y \in \mathrm{Follow}(A)$.

Here the proof considers a parse tree and a subtree followed by y, and is carried out by induction on the depth of that subtree in the tree.

Claim. The grammar G' is LL(1).

The proof naturally relies on the LL(k) property of G. □

Lemma 1 and Lemma 2 together imply the desired result.

Theorem 1. *For every linear LL(k) grammar $G = (\Sigma, N, R, S)$ there exists a linear LL(1) grammar with $|N| \cdot |\Sigma^{\leqslant k-1}|^2$ nonterminal symbols that describes the same language.*

6 Lower bound

The above construction, applied to an LL(k)-linear grammar with a set of nonterminal symbols N, produces a new LL(1)-linear grammar with as many as $|N| \cdot |\Sigma^{\leqslant k-1}|^2$ nonterminal symbols. The next result is that almost as many nonterminal symbols are in the worst case necessary.

Theorem 2. *For every $m \geqslant 3$, $k \geqslant 4$ and $n \geqslant 1$, there exists a language described by an LL(k)-linear grammar G over an m-symbol alphabet, with n nonterminal symbols, so that every LL(1)-linear grammar for the same language has at least $n \cdot (m-1)^{2k-3-\log_{m-1} k}$ nonterminal symbols.*

Let $\Sigma \cup \{\#\}$, where $\# \notin \Sigma$ is a special symbol, be an m-symbol alphabet. The grammar shall have rules of the form $A \to xAf(x)$, with $x \in \Sigma^{k-1}\#$, for some function $f\colon \Sigma^{k-1}\# \to \Sigma$, as well as a rule $A \to \varepsilon$. The function shall be defined in a way that in order to detect where the rule $A \to \varepsilon$ should be applied, an LL(1)-linear parser would have to buffer almost $2k$ symbols.

Consider the last C arguments of f. Once the parser reads the first $k-C$ symbols $c_1 \dots c_{k-C}$ of a presumed block $x \in \Sigma^{k-1}\#$, and has C further symbols to read, $d_{k-C+1} \dots d_{k-1}\#$, in order to compute f on this block, it needs to remember a function $g\colon \Sigma^{C-1} \to \Sigma$ defined by $g(d_{k-C+1} \dots d_{k-1}) = f(c_1 \dots c_{k-C} d_{k-C+1} \dots d_{k-1}\#)$. The goal is to choose C and f, so that the number of these functions g is as large as possible.

Lemma 3. *For $C = \log_{|\Sigma|} k + 1$, there exists a function $f\colon \Sigma^{k-1}\# \to \Sigma$, for which all functions with the first $k-C$ arguments substituted, $g_w(w') = f(ww')$, for $w \in \Sigma^{k-C}$, are pairwise distinct.*

Proof. There are $|\Sigma|^{|\Sigma|^{C-1}}$ distinct functions $g\colon \Sigma^{C-1}\# \to \Sigma$. In order to reach different functions g_w upon reading distinct leading prefixes $w \in |\Sigma|^{k-C}$, the number of leading prefixes should not exceed the number of functions.

$$|\Sigma|^{k-C} \leqslant |\Sigma|^{|\Sigma|^{C-1}}$$

This inequality holds true, because $C = \log_{|\Sigma|} k + 1$ implies $k - C \leq |\Sigma|^{C-1}$.

Then, strings $w \in \Sigma^{k-C}$ can be injectively mapped to functions $g_w\colon \Sigma^{C-1}\# \to \Sigma$, so that the desired function f is defined as $f(ws) = g_w(s)$, for all $w \in \Sigma^{k-C}$ and $s \in \Sigma^{C-1}\#$. □

With f fixed, the grammar $G = (\Sigma \cup \{\#\}, N, R, S)$ is defined to have $N = \{A_1, \dots, A_n\}$ and $S = A_1$, with the following rules. Each nonterminal symbol has the rules that match x on the left to $f(x)$ on the right.

$$A_i \to xA_i f(x) \qquad (A_i \in N,\ x \in \Sigma^{k-1}\#)$$

In the middle of the string, there are two possibilities. First, there can be nothing, which shall be detected only k symbols later, when the marker (#) is not found in its usual place.

$$A_i \to \varepsilon \qquad (A_i \in N)$$

The other possibility is that the number of the nonterminal symbol is explicitly written in the middle. This is detected immediately; the reason for having this case is that it forces a parser to remember this number.

$$A_i \to b\#^i \qquad (A_i \in N)$$

Finally, a parser changes the number of the nonterminal symbol by reading an explicit command.

$$A_i \to \#A_{i+1} \qquad (A_i \in N \setminus \{A_n\})$$

In particular, $L(G)$ contains strings of the following form.

$$\begin{aligned} &\#^{i-1}x_1 \dots x_n f(x_n) \dots f(x_1), && \text{with } i, n \geqslant 0,\ x_1, \dots, x_n \in \Sigma^{k-1}\# \\ &\#^{i-1}x_1 \dots x_n b \#^i f(x_n) \dots f(x_1), && \text{with } i, n \geqslant 1,\ x_1, \dots, x_n \in \Sigma^{k-1}\# \end{aligned}$$

This is the LL(k)-linear grammar promised in Theorem 2. It remains to prove that every LL(1)-linear grammar G' for the same language must have at least as many nonterminal symbols as stated in the theorem.

Consider the stack contents of an LL(1) parser for G' after reading a string of the following form, for some $i \in \{1, \dots, n\}$, $x_1, \dots, x_k \in \Sigma^{k-1}\#$, and $a_1, \dots, a_{k-C} \in \Sigma$.

$$u = \#^{i-1}x_1 \dots x_k a_1 \dots a_{k-C}$$

Since the grammar is linear, the stack contains some symbols Av, with $A \in N'$ and $v \in (\Sigma \cup \{\#\})^*$. The general plan of the proof is to show that, for a certain large set of strings u of the above form, the corresponding nonterminal symbols A must be pairwise distinct.

The first claim is that most of the information the parser stores at this point is encoded in the nonterminal symbol.

Lemma 4. *Let $G' = (\Sigma \cup \{\#\}, N', R', S')$ be any LL(1) grammar that describes the same language as G, and let Av be the stack contents after reading a string of the form $u = \#^{i-1}x_1 \dots x_k a_1 \dots a_{k-C}$, with $i \in \{1, \dots, n\}$, $x_1, \dots, x_k \in \Sigma^{k-1}\#$, and $a_1, \dots, a_{k-C} \in \Sigma$. Then, $|v| \leqslant 2$.*

Proof. Let $a_{k-C+1}, \dots, a_{k-1} \in \Sigma$ be some symbols following u, and consider the following two blocks.

$$\begin{aligned} x_{k+1} &= a_1 \dots a_{k-1}\# \\ x_{k+2} &= f(x_{k+1}) f(x_k) \dots f(x_3)\# \end{aligned}$$

Consider the following two strings obtained by extending u with the former block and with both blocks, respectively.

$$\begin{aligned} w_1 &= \#^{i-1}x_1 \dots x_k x_{k+1} f(x_{k+1}) f(x_k) \dots f(x_3) f(x_2) f(x_1) \\ w_2 &= \#^{i-1}x_1 \dots x_k x_{k+1} x_{k+2} f(x_{k+2}) f(x_{k+1}) f(x_k) \dots f(x_3) f(x_2) f(x_1) \end{aligned}$$

Both strings begin with u and are in $L(G)$, and have the following prefix.

$$w = \#^{i-1}x_1 \dots x_k x_{k+1} f(x_{k+1}) f(x_k) \dots f(x_4)$$

The longest common prefix of w_1 and w_2 is $wf(x_3)$. Let $A'v'$ be the stack contents after reading w. At this point, the parser sees the symbol $f(x_3)$. There are at least two strings in $L(G)$ that begin with $wf(x_3)$—namely, w_1 and w_2—and therefore the right-hand side of the rule in the LL(1)-table for $(A', f(x_3))$ begins with a nonterminal symbol. The sequence of rules applied until $f(x_3)$ is finally read is

then of the following form, for some $B_1, \ldots, B_\ell \in N'$, $t_1, \ldots, t_\ell \in (\Sigma \cup \{\#\})^*$ and $\gamma \in (\Sigma \cup \{\#\} \cup N')^*$.

$$\begin{aligned} A' &\to B_1 t_1 \\ B_1 &\to B_2 t_2 \\ &\vdots \\ B_{\ell-1} &\to B_\ell t_\ell \\ B_\ell &\to f(x_3)\gamma \end{aligned}$$

While the parser applies these rules, it cannot pop any of the symbols of v at the bottom of the stack, and hence the resulting stack contents after reading $wf(x_3)$ are $\gamma v''$, where $v'' \in (\Sigma \cup \{\#\})^*$ and v is a suffix of v''.

For the parser to accept the string $w_1 = wf(x_3)f(x_2)f(x_1)$, after reading $wf(x_3)$, it must have at most two symbols on the stack: that is, $|v''| = 2$. Since v is a suffix of v'', this proves the lemma. □

Now consider any *two* distinct strings of the same general form as in Lemma 4. It is claimed that the parser must have different nonterminal symbols in the stack after reading these strings.

Lemma 5. *Let $G' = (\Sigma \cup \{\#\}, N', R', S')$ be any LL(1) grammar for the same language as G, and consider two strings of the following form.*

$$u_1 = \#^{i-1} x_1 \ldots x_k a_1 \ldots a_{k-C} \quad (i \in \{1, \ldots, n\},\ x_1, \ldots, x_k \in \Sigma^{k-1}\#,\ a_1, \ldots, a_{k-C} \in \Sigma)$$
$$u_2 = \#^{j-1} y_1 \ldots y_k b_1 \ldots b_{k-C} \quad (j \in \{1, \ldots, n\},\ y_1, \ldots, y_k \in \Sigma^{k-1}\#,\ b_1, \ldots, b_{k-C} \in \Sigma)$$

Assume that either $i \neq j$, or $f(x_3) \ldots f(x_k) \neq f(y_3) \ldots f(y_k)$, or $a_1 \ldots a_{k-C} \neq b_1 \ldots b_{k-C}$. Let Av and Bv' be the parser's stack contents after reading these strings. Then, $A \neq B$.

Proof. Suppose, for the sake of a contradiction, that $A = B$, and accordingly the stack contents after reading u_1 and u_2 are Av and Av', respectively.

Denote $\widetilde{a} = a_1 \ldots a_{k-C}$ and $\widetilde{b} = b_1 \ldots b_{k-C}$. If $\widetilde{a} \neq \widetilde{b}$, then, by the construction of f, there is a string $z = c_1 \ldots c_{C-1}\#$ of length C, with $c_1, \ldots, c_{C-1} \in \Sigma$, that satisfies $f(\widetilde{a}z) \neq f(\widetilde{b}z)$; if $\widetilde{a} = \widetilde{b}$, then let $z = c_1 \ldots c_{C-1}\#$ be any string with $c_1, \ldots, c_{C-1} \in \Sigma$.

The following two strings are in $L(G)$.

$$u_1 z f(\widetilde{a}z) f(x_k) \ldots f(x_1)$$
$$u_1 z b \#^i f(\widetilde{a}z) f(x_k) \ldots f(x_1)$$

Since the stack contains Av after reading their common prefix u_1, both remaining suffixes must be in $L_{G'}(Av)$. Denote the substrings in $L_{G'}(A)$ by w_1 and w_2, respectively.

$$w_1 = z f(\widetilde{a}z) f(x_k) \ldots f(x_{|v|+1})$$
$$w_2 = z b \#^i f(\widetilde{a}z) f(x_k) \ldots f(x_{|v|+1})$$

(note that $|v| \leqslant 2$ by Lemma 4)

By the assumption, u_1 and u_2 must differ either in the number of sharp signs in the beginning ($i \neq j$), or in one of the images of the last $k-2$ complete blocks ($f(x_3)\dots f(x_k) \neq f(y_3)\dots f(y_k)$), or in one of the symbols in the last incomplete block ($\widetilde{a} \neq \widetilde{b}$), There are accordingly three cases to consider.

- Let $i \neq j$. The parser's stack contents after reading u_2 is Av', and therefore the string u_2w_2v' must be accepted.

  $$u_2w_2v' = \#^{j-1}y_1\dots y_kb_1\dots b_{k-C}zb\#^if(\widetilde{a}z)f(x_k)\dots f(x_{|v|+1})v'$$

  However, the mismatch between the prefix $\#^{j-1}$ and the substring $b\#^i$ in the middle means that the string is not in $L(G)$. This is a contradiction.
- In the case when $f(x_3)\dots f(x_k) \neq f(y_3)\dots f(y_k)$, after reading u_2, the parser has Av' in its stack, and thus must accept u_2w_1v'.

  $$u_2w_1v' = \#^{j-1}y_1\dots y_kb_1\dots b_{k-C}zf(\widetilde{a}z)f(x_k)\dots f(x_{|v|+1})v'$$

  However, since $f(x_3)\dots f(x_k) \neq f(y_3)\dots f(y_k)$, this string is not in $L(G)$ and cannot be accepted, contradiction.
- Assume that $\widetilde{a} \neq \widetilde{b}$. Then, by the construction, $f(\widetilde{a}z) \neq f(\widetilde{b}z)$. The rest is like in the previous case: the parser accepts u_2w_1v', which is not in $L(G)$. □

Proof (of Theorem 2). For all $i \in \{1,\dots,n\}$, $d_3,\dots,d_k \in \Sigma$ and $a_1,\dots,a_{k-C} \in \Sigma$, let $x_1,\dots,x_k \in \Sigma^{k-1}\#$, with $f(x_j) = d_j$ for all $j \in \{3,\dots,n\}$. Then the corresponding string $u_{i;d_3,\dots,d_k;a_1,\dots,a_{k-C}}$ is defined as follows.

$$u_{i;d_3,\dots,d_k;a_1,\dots,a_{k-C}} = \#^{i-1}x_1\dots x_ka_1\dots a_{k-C}$$

By Lemma 5, upon reading different strings of this form, the LL(1)-linear parser must have pairwise distinct nonterminal symbols in its stack. Therefore, there must be at least as many nonterminal symbols as there are such strings, that is, $n \cdot (m-1)^{2k-C-2}$, as claimed. □

7 Conclusion

The collapse of the hierarchy of LL(k)-linear languages establishes the *LL-linear languages* as a robust language family that deserves future investigation.

In particular, the succinctness tradeoff between LL(k)-linear grammars with different k has been determined only with respect to the number of nonterminal symbols. It would be interesting to know whether the elimination of lookahead similarly affects the total length of description. The witness languages constructed in this paper do not establish any lower bounds on that, and more research is accordingly needed to settle this question.

Another suggested line of research is investigating LL(k)-linear conjunctive grammars and LL(k)-linear Boolean grammars [8], and determining whether a lookahead hierarchy exists for those grammar families.

References

1. C. de la Higuera, J. Oncina, "Inferring Deterministic Linear Languages", *Computational Learning Theory* (COLT 2002, Sydney, Australia, July 8–10, 2002), LNCS 2375, 185–200.
2. M. Holzer, K.-J. Lange, "On the complexities of linear LL(1) and LR(1) grammars", *Fundamentals of Computation Theory* (FCT 1993, Hungary, August 23–27, 1993), LNCS 710, 299–308.
3. O. H. Ibarra, T. Jiang, B. Ravikumar, "Some subclasses of context-free languages in NC^1", *Information Processing Letters*, 29:3 (1988), 111–117.
4. G. Jirásková, O. Klíma, "Deterministic biautomata and subclasses of deterministic linear languages", *Language and Automata Theory and Applications—13th International Conference* (LATA 2019, St. Petersburg, Russia, March 26–29, 2019), LNCS 11417, 315–327.
5. D. E. Knuth, "Top-down syntax analysis", *Acta Informatica*, 1 (1971), 79–110.
6. R. Kurki-Suonio, "Notes on top-down languages", *BIT Numerical Mathematics*, 9:3 (1969), 225–238.
7. P. M. Lewis II, R. E. Stearns, "Syntax-directed transduction", *Journal of the ACM*, 15:3 (1968), 465–488.
8. A. Okhotin, "Expressive power of LL(k) Boolean grammars", *Theoretical Computer Science*, 412:39 (2011), 5132–5155.
9. D. J. Rosenkrantz, R. E. Stearns, "Properties of deterministic top-down grammars", *Information and Control*, 17 (1970), 226–256.

Appendix

A Details for the proof of Lemma 1 on page 4

Claim. If a string w is defined by A_u in the new grammar, then $w = xu$ and A defines x in the original grammar.

Proof. Induction on the height of a parse tree of a string w as A_u.

Base case: w is obtained by a rule $A_u \to w$. By construction, the rule is of the form $A_u \to xu$, obtained from a rule $A \to x$ in R, and $w = xu$. Then, $x \in L_G(A)$.

Induction step. Assume that w is defined by a rule $A_u \to w_1 B_s t$, which was obtained from a rule $A \to w_1 B w_2$, with $w_2 u = st$. Then, $w = w_1 y t$, where $y \in L_{G'}(B_s)$, and the height of a parse tree of y as B_s is less than that of the parse tree of w. Then, by the induction hypothesis, $y = zs$, for some $z \in L_G(B)$. Altogether, $w = w_1 zst = w_1 z w_2 u$, and therefore $w \in L_G(A)u$ by the rule $A \to w_1 B w_2$.

□

Claim. If a string x is defined by A in the original grammar, then A_u defines xu in the new grammar.

Proof. Induction on the height of a parse tree of a string x as A.

Base case: x is obtained by a rule $A \to x$. By construction, grammar G' contains the rule $A_u \to xu$, thus $xu \in L_{G'}(A_u)$.

Induction step. Assume that x is defined by a rule $A \to w_1 B w_2$. Then $w = w_1 y w_2$ for some $y \in L_G(B)$. The height of a parse tree of y as B is less than that of the parse tree of w. Then, by the induction hypothesis, $ys \in L_{G'}(B_s)$, where s is the first $k-1$ symbols of the string $w_2 u$. The grammar G' has a rule $A_u \to w_1 B_s t$, obtained from the rule $A \to w_1 B w_2$ in G, and thus $w_1 yst \in L_{G'}(A_u)$. Since $w_1 yst = w_1 y w_2 u = wu$, the string wu is in $L_{G'}(A_u)$.

□

Claim. Let $A_u \to w_1 B_s t$ be a rule in G' obtained from a rule $A \to w_1 B w_2$ in G. Then $L_{G'}(w_1 B_s t) = L_G(w_1 B w_2)u$.

Proof. Indeed, by the construction, $st = w_2 u$, and $L_{G'}(B_s) = L_G(B)s$ by the previous claim. Altogether, $L_{G'}(w_1 B_s t) = w_1 L_G(B) st = L_G(w_1 B w_2)u$. □

Claim. If there is a rule $A_u \to w_1 B_s t$ in G', then $|s| \geqslant |u|$, and if furthermore $|t| > 0$, then $|s| = k-1$.

Proof. By the construction of G', the rule $A_u \to w_1 B_s t$ was obtained from some rule $A \to w_1 B w_2$, with $|s| = \min(|w_2 u|, k-1)$ and $st = w_2 u$. Then, $|s| = \min(|w_2 u|, k-1) \geqslant \min(|u|, k-1) = |u|$. If furthermore $|t| > 0$, then $|s| < |w_2 u|$, and therefore, since $|s| = \min(|w_2 u|, k-1)$ and $|s| \neq |w_2 u|$, it follows that $|s| = k-1$. □

Claim. For each nonterminal symbol $B_s \in N'$ if $y \in \text{Follow}(B_s)$ then $sy \in \text{Follow}(B)$. If furthermore $|s| < k-1$, then $y = \varepsilon$.

Proof. Let y be a string from $\text{Follow}(B_s)$, that is, there exists a parse tree containing a subtree with the root B_s, and with the string y as the suffix to the right of this subtree.

The proof is carried out by induction on the depth of the subtree within the tree.

Base: B_s is the root of whole parse tree, that is $B_s = S_\varepsilon$. Since there are no symbols to the right of the whole tree, $y = \varepsilon$ and thus $sy = \varepsilon \in \text{Follow}(S)$.

Induction step: let A_u be the ancestor of the subtree. Let $A_u \to w_1 B_s t$ be the rule applied to A_u, and let $A \to w_1 B w_2$ be the rule in G, with $w_2 u = st$, from which it was obtained. Let y' denote the suffix to the right of the subtree with root labeled as A_u, so that $y = ty'$ and $y' \in \text{Follow}(A_u)$. As the subtree B_s is deeper than the subtree A_u, the induction hypothesis can be applied to A_u and $y' \in \text{Follow}(A_u)$, and it asserts that $uy' \in \text{Follow}(A)$. Since $w_2 u = st$, the string $w_2 uy'$ is in $\text{Follow}(B)$. Furthermore, since $w_2 uy' = sty' = sy$, it follows that $sy \in \text{Follow}(B)$.

Now assume $|s| < k-1$. The previous claim states that $|s| \geqslant |u|$ and that if furthermore $|t| > 0$, then $|s| = k-1$. This implies that $|u| < k-1$ and $t = \varepsilon$. Now, the induction hypothesis further asserts that if $|u| < k-1$, then $y' = \varepsilon$, and therefore $y = ty' = \varepsilon$.

□

Claim. For each nonterminal symbol $A_u \in N'$ with $|u| = k-1$, all strings defined by A_u are of length at least $k-1$.

Proof. Let $w \in L_{G'}(A_u)$. The proof is carried out by induction on the height of a parse tree of w as A_u.

Base: w is obtained by rule $A_u \to w$. By the construction of the grammar G', the rule $A_u \to w$ was obtained from a rule $A \to w'$, where $w'u = w$. Then the length of w is $|w| = |w'u| \geqslant k-1$.

Induction step. Assume that w is defined by a rule $A_u \to w_1 B_s t$, which was obtained from a rule $A \to w_1 B w_2$, with $w_2 u = st$. Then $w = w_1 yt$ for some $y \in L_{G'}(B_s)$.

The height of a parse tree of y as B_s is less than that of the parse tree of w. Then, by the induction hypothesis, if $|s| = k-1$, then $|y| \geqslant k-1$, and therefore $|w| = |w_1 yt| \geqslant k-1$.

By construction, $|s| = \min(|w_2 u|, k-1)$ and since $|u| = k-1$ we get $|s| = k-1$, so $|w| \geqslant k-1$. □

Claim. If the original grammar is $\text{LL}(k)$, then so is the constructed grammar.

Proof. Let A_u be a nonterminal symbol in G', with the following rules.

$$\begin{aligned} A_u &\to \alpha_1 \\ A_u &\to \alpha_2 \\ &\vdots \\ A_u &\to \alpha_n \end{aligned}$$

By definition the grammar G' has the LL(k) property if and only if, for each A_u, the sets $\mathrm{First}_k(\alpha_j \mathrm{Follow}(A_u))$ are pairwise disjoint.

By construction each rule α_j is uniquely defined from a rule γ_j of grammar G. Since grammar G is $LL(k)$, the sets $\mathrm{First}_k(\gamma_j \mathrm{Follow}(A))$ are pairwise disjoint. So to prove that G' is $LL(k)$ it is sufficient to show that $\mathrm{First}_k(\alpha_j \mathrm{Follow}(A_u)) \subseteq \mathrm{First}_k(\gamma_j \mathrm{Follow}(A))$ for each rule α_j.

By $L_{G'}(\alpha_j) = L_G(\gamma_j)u$ for each rule α_j and by the previous claim $u\mathrm{Follow}(A_u) \subseteq \mathrm{Follow}(A)$.

Therefore

$$\mathrm{First}_k(\alpha_j \mathrm{Follow}(A_u)) = \mathrm{First}_k(\gamma_j u \mathrm{Follow}(A_u)) \subseteq \mathrm{First}_k(\gamma_j \mathrm{Follow}(A))$$

end the proof is complete.

□

B Details for the proof of Lemma 2 on page 6

The type of each rule ${}_uA \to \alpha$ can be determined from its general form as follows. If α contains a nonterminal ${}_vB$ and $|v| > |u|$, then this is a rule for appending a new symbol to the buffer. If $T(A, u)$ is defined, this is the case when the end of the input string is visible, and the rule must be ${}_uA \to \varepsilon$. In all other cases, the rule ${}_uA \to \alpha$ is obtained from one of the rules in R; the original rule can be uniquely reconstructed by processing $u\alpha$ as follows: $us'_\varepsilon Bt$ becomes sBt; $u_v Bt = sv_v B$ becomes sBt by cancelling v and the buffer of B; ux remains as it is.

Claim. If $x \in L_{G'}({}_uA)$, then $ux \in L_G(A)$.

Induction on the length of the derivation of x in G'.

Base case: x is defined by a rule ${}_uA \to x$. By construction there are two types of rules in G' with no nonterminals in the right side: rules obtained from those in grammar G and rules corresponging to the case when the end of the input string is visible, that is $T(A, u)$.

In the first case the rule ${}_uA \to x$ was obtained from some rule $A \to ux$ of grammar G thus $ux \in L_G(A)$.

Now consider the second case. Then by construction $x = \varepsilon$. Since $|u| < k$ and the end of the string is visible $u = x't$ for some $t \in \mathrm{Follow}(A)$ and

$x' \in L_G(A)$. Therefore $|x'| \leq |u| < k$ and since grammar G has no short rules, $t = \varepsilon$.

Then $u = x'$, so $ux = u = x' \in L_G(A)$.

Induction step. Assume x is defined by a rule ${}_uA \to \gamma$ where γ contains a nonterminal symbol. Then rule γ may be obtained from a rule in grammar G or be a rule appending new symbol to the buffer:

If γ was obtained from a rule sBt of grammar G, there are two cases depending on which of the strings u and s is longer:

- If $|u| > |s|$ then $\gamma = {}_vBt$ with $u = sv$. Then $x = yt$ for some $y \in L_{G'}({}_vB)$. Height of a parse tree of y as ${}_vB$ is less than that of the parse tree of x. Then by induction hypothesis $vy \in L_G(B)$. Hence $ux = uyt = svyt \in L_G(sBt) \in L_G(A)$.
- If $|u| \leqslant |s|$, then $\gamma = s'{}_\varepsilon Bt$ with $us' = s$. Then $x = s'yt$ with $y \in L_{G'}({}_\varepsilon B)$.
  By the induction hypothesis, $y \in L_G(B)$. Hence, $ux = us'yt = syt \in L_G(sBt) \in L_G(A)$.

Now assume γ is a rule appending new symbol to the buffer. Then $\gamma = a_{ua}A$ for some $a \in \Sigma$. Then $x = ay$ with $y \in L_{G'}({}_{ua}A)$.

Hence by induction hypothesis $uay = ux \in L_G(A)$.

Claim. Let ${}_uA \to \gamma$ be a rule obtained from a rule $A \to \alpha$. Then, if $x \in L_{G'}(\gamma)$, then $ux \in L_G(\alpha)$.

If $\alpha \in \Sigma^*$ then by construction $\gamma = x$ and $\alpha = ux$. Now assume $\alpha = sBt$ for some $s, t \in \Sigma^*$ and $B \in N$.

If $|u| < |s|$ then $\gamma = s'{}_\varepsilon Bt$ with $us' = s$. Then $x = s'yt$ for some $y \in L_{G'}({}_\varepsilon B)$. As it was proved if $y \in L_{G'}({}_\varepsilon B)$ then $y \in L_G(B)$ so $ux = us'yt = syt \in L_G(sBt)$.

If $|u| \geqslant |s|$ then $\gamma = {}_vBt$ with $u = sv$. Then $x = yt$ for some $y \in L_{G'}({}_vB)$. As it was proved if $y \in L_{G'}({}_vB)$ then $vy \in L_G({}_vB)$. Thus $ux = svyt \in L_G(sBt)$.

Claim. If $ux \in L_G(A)$, then $x \in L_{G'}({}_uA)$.

Proof. The proof is given separately for strings ux of length less than k with $T(A, ux)$ defined and for all other strings.

Firstly assume that $|ux| < k$ and $T(A, ux)$ is defined.

Then by construction grammar G' contains the rule ${}_{ux}A \to \varepsilon$.

Let $x = x_1 \dots x_m$. Since u is a prefix of ux, the buffer of ${}_uA$ can be increased up to ux by the following rules:

$$
\begin{aligned}
{}_uA &\to x_1\ {}_{ux_1}A \\
{}_{ux_1}A &\to x_2\ {}_{ux_1x_2}A \\
{}_{ux_1\dots x_{m-1}}A &\to x_m\ {}_{ux}A
\end{aligned}
$$

The chain of rules above together with the rule ${}_{ux}A \to \varepsilon$ construct the derivation of x as ${}_uA$.

Therefore $x \in L_{G'}({}_uA)$.

Now assume that $|ux| \geq k$ or $|ux| = k-1$ but $T(A, ux)$ is not defined.

If $|ux| = k-1$ then Follow$(A) \neq \{epsilon\}$ and there exists a symbol $c \in \text{Follow}_1(A)$.

Let $n = k - |u| - 1$ and symbol a denote either x_{n+1} if $|ux| \geq k$ or c if $|ux| = k-1$.

Let u' denote the string $ux_1 \dots x_n$.

Then $|u'| = k-1$ and $T(A, u'a) = \gamma$, where γ is the rule, defining ux as A.

As in the previous case, the buffer of ${}_uA$ can be increased up to u' by the rules:

$$
\begin{aligned}
{}_uA &\to x_1\ {}_{ux_1}A \\
{}_{ux_1}A &\to x_2\ {}_{ux_1x_2}A \\
{}_{ux_1\dots x_{n-1}}A &\to x_n\ {}_{u'}A
\end{aligned}
$$

Then since $T(A, u'a) = \gamma$, grammar G' contains a rule ${}_{u'}A \to \alpha$, obtained from the rule $A \to \gamma$ of grammar G.

It will be proved that α defines the string $x_{n+1} \dots x_m$ and thus the chain of rules above together with the rule ${}_{u'}A \to \alpha$ construct the derivation of $x_1 \dots x_m = x$ as ${}_uA$.

The proof is carried out by induction on the length of the derivation of ux as A.

Base: ux is defined by a single rule $A \to ux$ Then $\gamma = T(A, u'a) = A \to ux$ and by construction $\alpha = x_{n+1} \dots x_m$. Therefore $x \in L_{G'}({}_uA)$.

Induction step: assume ux is defined by a rule $A \to sBt$]

Then $ux = syt$ for some string $y \in L_G(B)$ and the rule ${}_{u'}A \to \alpha$ is obtained from the rule $A \to sBt$ of grammar G.

There are two cases depending on which of the strings u' and s is longer.

If u' is shorter than s then $\alpha = s'{}_\varepsilon Bt$ with $u's' = s$.

Height of a parse tree of y as B is less than that of the parse tree of ux. Then by induction hypothesis $y \in L_{G'}({}_\varepsilon B)$.

Therefore $s'yt \in L_{G'}({}_{u'}A)$. Note that $u's'yt = syt = ux = u'x_{n+1} \dots x_m$.

Thus $s'yt = x_{n+1} \dots x_m$ and $x_{n+1} \dots x_m \in L_{G'}({}_{u'}A)$ as desired.

If s is shorter than u' then $\alpha = {}_vBt$ with $sv = u'$

To apply the induction hypothesis for ${}_vB$ it is necessary to prove that v is a prefix y.

Since grammar G has no short rules, either $|y| \geq k-1$ or $t = \varepsilon$.

If $t = \varepsilon$ then $ux = sy$ and since $sv = u'$ is a prefix of ux, v is a prefix of y.

If $|y| = k-1$ then $|sy| \geq k-1$. Since u' is a prefix of $ux = syt$ and $|u'| < k$ it is obtained that $u' = sv$ is a prefix of sy, thus again v is a prefix of y.

Therefore $y = vy'$ for some $y' \in \Sigma^*$ and by induction hypothesis $y' \in L_{G'}({}_vB)$.

Hence $y't \in L_{G'}({}_{u'}A)$.

Note that $u'y't = svy't = syt = ux = u'x_{n+1}\dots x_m$. Therefore $y't = x_{n+1}\dots x_m$ and thus $x_{n+1}\dots x_m \in L_{G'}({}_{u'}A)$ as desired.

□

Claim. For each $u \in \Sigma^{\leqslant k-1}$, if $y \in \mathrm{Follow}({}_uA)$, then $y \in \mathrm{Follow}(A)$.

Since $y \in \mathrm{Follow}({}_uA)$, there exists a subtree with root ${}_uA$ with the string y as the suffix to the right of this subtree. The proof that $y \in \mathrm{Follow}(A)$ is carried out by induction on the height of the subtree.

Base: ${}_uA$ is the root of whole parse tree, that is ${}_uA = {}_\varepsilon S$ In this case $y = \varepsilon$ since there are no symbols to the right of the whole tree. Therefore $y = \varepsilon \in \mathrm{Follow}(S)$

Induction step: let ${}_vB$ be ancestor of the subtree Let γ be the rule applied to ${}_vB$, and let y' denote the symbols to the right of the subtree with root ${}_vB$, so that $y' \in \mathrm{Follow}({}_vB)$.

The rule γ may either be obtained from a rule of grammar G or it may be a rule that appends new symbols to the buffer.

In the first case, γ is obtained from some rule $B \to w_1Aw_2$. The height of the subtree with root ${}_vB$ is less than that of the subtree with root ${}_uA$.

Then, by the induction hypothesis, $y' \in \mathrm{Follow}({}_vB)$ entails $y' \in \mathrm{Follow}(B)$. Therefore, the string $y = w_2y'$ is in $\mathrm{Follow}(A)$.

In the second case, since γ is a rule for appending symbols to the buffer, $B = A$ and $u = va$ for some symbol $a \in \Sigma$, so actually $\gamma = {}_vA \to a_uA$. Thus, $y' = y$, and so $y \in \mathrm{Follow}({}_vA)$.

The height of the subtree with root ${}_vA$ is less than that of the subtree with root ${}_uA$. By the induction hypothesis $y \in \mathrm{Follow}({}_vA)$, and therefore $y \in \mathrm{Follow}(A)$.

Claim. The grammar G' is LL(1).

The grammar G' has $LL(1)$ property if and only if for any ${}_uA \in N'$ and two distinct rules ${}_uA \to \gamma_1$ and ${}_uA \to \gamma_2$ the sets $\mathrm{First}_1(\gamma_1\mathrm{Follow}({}_uA))$ and $\mathrm{First}_1(\gamma_2\mathrm{Follow}({}_uA))$ do not intersect.

To prove that there are will be considered two cases depending on whether $|u| < k-1$ or $|u| = k-1$.

If $|u| < k-1$, each of the strings γ_1 and γ_2 is whether a rule increasing buffer or a rule ${}_uA \to \varepsilon$ if $T(A,u)$ is defined.

If one of the rules, say γ_1, is a rule ${}_uA \to \varepsilon$ then $u \in L_G(A)$.

Then since $|u| < k-1$ and grammar G has no short rules $\mathrm{Follow}(A) = \{\varepsilon\}$. As it was proved $\mathrm{Follow}({}_uA) \subseteq \mathrm{Follow}(A)$ thus $\mathrm{Follow}({}_uA) = \{\varepsilon\}$. Therefore $\mathrm{First}_1(\gamma_1\mathrm{Follow}({}_uA)) = \{\varepsilon\}$.

Since rules γ_1 and γ_2 differ, γ_2 is a rule increasing buffer so $\gamma_2 = A \to a_{ua}A$ for some $a \in \Sigma$.

Therefore $\mathrm{First}_1(\gamma_1\mathrm{Follow}({}_uA)) \cap \mathrm{First}_1(\gamma_2\mathrm{Follow}({}_uA)) = \varnothing$ since $\varepsilon \notin L_{G'}(a_{ua}A)$.

If $|u| = k-1$ then γ_1 and γ_2 were obtained from grammar G rules α_1 and α_2 correspondingly.

Let $a \in \mathrm{First}_1(\gamma_1 \mathrm{Follow}({}_u A)) \cap \mathrm{First}_1(\gamma_2 \mathrm{Follow}({}_u A))$.

As it was proved $\mathrm{Follow}({}_u A) \subseteq \mathrm{Follow}(A)$ thus $ua \in \mathrm{First}_k(\alpha_1 \mathrm{Follow}(A)) \cap \mathrm{First}_k(\alpha_2 \mathrm{Follow}(A))$.

But that contradicts $LL(k)$ property of grammar G so $\mathrm{First}_1(\gamma_1 \mathrm{Follow}({}_u A)) \cap \mathrm{First}_1(\gamma_1 \mathrm{Follow}({}_u A)) = \varnothing$ and the proof is finished.