Computing the Longest Common Prefix of a Context-free Language in Polynomial Time

Michael Luttenberger
Technische Universität München, Germany
luttenbe@in.tum.de

Raphaela Palenta
Technische Universität München, Germany
palenta@in.tum.de

Helmut Seidl
Technische Universität München, Germany
seidl@in.tum.de

Abstract

We present two structural results concerning the longest common prefixes of non-empty languages. First, we show that the longest common prefix of the language generated by a context-free grammar of size N equals the longest common prefix of the same grammar where the heights of the derivation trees are bounded by $4N$. Second, we show that each non-empty language L has a representative subset of at most three elements which behaves like L w.r.t. the longest common prefix as well as w.r.t. longest common prefixes of L after unions or concatenations with arbitrary other languages. From that, we conclude that the longest common prefix, and thus the longest common suffix, of a context-free language can be computed in polynomial time.

2012 ACM Subject Classification Mathematics of computing → Combinatorics on words, Theory of computation → Algebraic language theory, Theory of computation → Grammars and context-free languages

Keywords and phrases Longest Common Prefix, Context-free Languages, Combinatorics on Words

Digital Object Identifier 10.4230/LIPIcs.STACS.2018.48

Related Version An extended version of this article is available at `https://arxiv.org/abs/1702.06698`, [11].

1 Introduction

Let Σ denote an alphabet. On the set Σ^* of all words over Σ, the prefix relation provides us with a partial ordering $\sqsubseteq$ defined by $u \sqsubseteq v$ iff $uu' = v$ for some $u' \in \Sigma^*$. The *longest common prefix* (lcp for short) of a non-empty set $L \subseteq \Sigma^*$ then is given by the greatest lower bound $\bigsqcap L$ of L w.r.t. this ordering. For two words $u, v \in \Sigma^*$, we also denote this greatest lower bound as $u \sqcap v$. Our goal is to compute the lcp when the language L is context-free, i.e., generated by a context-free grammar (CFG) — we therefore assume wlog. that Σ contains at least two letters.

The computation of the lcp (sometimes also *maximum common prefix*) is well studied for finite languages, in particular in the setting of string matching based on suffix arrays (e.g., [6]) where the string is given explicitly. Very often, strings can be efficiently compressed using *straight-line programs (SLPs)* — essentially CFGs which produce exactly one word. Interestingly, many of the standard string operations can still be done efficiently also on

© Michael Luttenberger, Raphaela Palenta, and Helmut Seidl;
licensed under Creative Commons License CC-BY

35th Symposium on Theoretical Aspects of Computer Science (STACS 2018).
Editors: Rolf Niedermeier and Brigitte Vallée; Article No. 48; pp. 48:1–48:13
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany
SYMPOSIUM ON THEORETICAL ASPECTS OF COMPUTER SCIENCE

SLP-compressed strings (see, e.g., [10]). As the union of SLPs is a (acyclic) CFG, the question of computing the lcp of a context-free language naturally arises. CFGs also represent a popular formalism to specify sets of well-formed words. Assume that we are given a CFG for the legal outputs of a program. This CFG might be derived from the specification as well as from an abstract interpretation of the program. Then the lcp of this language represents a prefix which can be output already, before the program actually has been run. This kind of information is crucial for the construction of *normal forms*, e.g., of string producing processors such as linear tree-to-string transducers [1, 8]. For these devices, the normal forms have further interesting applications as they allow for simple algorithms to decide equivalence [2] and enable efficient learning [9].

Obviously, the lcp of the context-free language L is a prefix of the shortest word in L. Since the shortest word of a context-free language can be effectively computed, the lcp of L is also effectively computable. The shortest word generated from a context-free grammar G, however, may be of length exponential in the size of G. Therefore, it is an intriguing question whether or not the lcp can be efficiently computed. Here, we show that the longest common prefix can in fact be computed in polynomial time. As the words the algorithm computes with may be of exponential length, we have to resort to *compressed* representations of long words by means of SLPs [12]. We will rely on algorithms for basic computational problems for SLPs as presented, e.g., in [10].

Our method of computing $\bigsqcap L$ is based on two structural results. First we show in Section 3 that it suffices to consider the finite sublanguage of L consisting of those words, for which there is a derivation tree of height at most $4N$ — with N the number of nonterminals for a CFG of L.[1] This implies that (1) in the proof of our main result we can replace the grammar by an *acyclic* context-free grammar, and (2) the actual fixpoint iteration to compute the lcp will converge within at most $4N$ iterations. Second we show in Section 4 that for every non-empty language L there is a subset $L' \subseteq L$ of at most three elements which is *equivalent* to L w.r.t. the lcp after arbitrary concatenations with other words. This means that for every word w, the language $L'w$ has the same lcp as Lw.

We illustrate both results by examples. For the first result, i.e. the restriction to derivation trees of bounded height, consider the language

$$L := \{a^2b(a^2b)^i a^2b(a^2ba)^i a^2ba^2ba^3 \mid i \in \mathbb{N}_0\}$$

generated by the context-free grammar consisting of the following rules over the alphabet $\Sigma = \{a, b, c\}$ and the six nonterminals $\{S, X, A_2, A_1, X_2, X_1\}$:

$$\begin{array}{llll} S \to X_2A_2bA_2bA_2a & A_2 \to aA_1 & A_1 \to a & X \to A_2b \\ & X_2 \to aX_1 & X_1 \to abX & X \to X_2A_2ba \end{array}$$

It is easy to check that here the lcp is already determined by repeating the derivation of X to $aabXaaba$ at most two times, which corresponds to the sublanguage consisting of all

[1] To simplify the presentation we assume that the CFG is proper, i.e. we will rule out production rules of the form $A \to B$ and $A \to \varepsilon$ (with A, B nonterminals and ε the empty word).

words which have a derivation tree of height at most 9.

$$\begin{array}{rcll}
\bigsqcap L & = & aab\textcolor{blue}{aab}aabaabaa\;a & (i=0)\\
& \sqcap & aab\textcolor{red}{aab}\textcolor{blue}{aab}\textcolor{green}{aaba}a\;abaabaaa & (i=1)\\
& \sqcap & aab\textcolor{red}{aabaab}\textcolor{blue}{aab}\textcolor{green}{aa\;baaaba}aabaabaaa & (i=2)\\
& \sqcap & aab\textcolor{red}{aabaabaab}\textcolor{blue}{aa\;b}\textcolor{green}{aabaaabaaaba}aabaabaaa & (i=3)\\
& \sqcap & aab\textcolor{red}{aabaabaabaa\;b}\dots & (i\geq 4)\\
& = & aabaabaabaabaa &
\end{array}$$

We remark that the bound of $4N$, i.e. 24 for this example, on the height resp. the number of iterations needed to converge is a crude overapproximation based on the pigeon-hole principle which does not take into account the structure of the grammar. The actual computation of the lcp may thus terminate much earlier, in particular when taking the dependency of nonterminals into account as done in Example 18.

In order to compute the lcp recursively, we call two languages $L_1, L_2 \subseteq \Sigma^*$ *equivalent w.r.t. the* lcp if for all words $w \in \Sigma^*$ we have that $\bigsqcap(L_1 w) = \bigsqcap(L_2 w)$. In Section 4 we show that every language L can be reduced to a sublanguage L' consisting of at most three words so that L and L' are equivalent w.r.t. the lcp . In fact, this result can be motivated by considering the special case of a language of the form $L = \{u, uv_1\}$ (with $u, v_1 \in \Sigma^*$) where we have $\bigsqcap(Lw) = u(w \sqcap v_1^\omega)$ for any $w \in \Sigma^*$ (see also Section 4). From this observation one immediately obtains that for finite languages $L' = \{uv_1, uv_2, \dots, uv_k\}$ we have $\bigsqcap(L'w) = u(w \sqcap v_1^\omega \sqcap v_2^\omega \sqcap \dots \sqcap v_k^\omega)$ and that one only needs to keep those two uv_i, uv_j for which $v_i^\omega \sqcap v_j^\omega$ is minimal. The result then extends to arbitrary languages. E.g., in case of the language $L = a(ba)^*$ we only need the sublanguage $\{a, aba\}$ (with $\varepsilon^\omega \sqcap (ba)^\omega := (ba)^\omega$) as the words a and aba suffice to characterize both $\bigsqcap L = a$ and the period ba that generates all suffices. For comparison, in case of $L = abab + aba(ba)^*$ the lcp is aba, which can only be extended to at most $abab = aba(b^\omega \sqcap (ba)^\omega)$. We therefore need to remember $\{aba, abab, ababa\}$: the sublanguages $\{aba, abab\}$ resp. $\{aba, ababa\}$ preserve $\bigsqcap L = aba$ but can be extended by b^ω resp. $(ba)^\omega$; whereas $\{abab, ababa\}$ only captures the maximal extension of $\bigsqcap L$, but does not preserve $\bigsqcap L$ itself.

In order to compute the lcp of a given context-free language L we then (implicitly) unfold the given context-free grammar into an acyclic grammar, and compute for every nonterminal of the unfolded grammar an equivalent sublanguage of at most three words, each compressed by means of a SLP, instead of the actual language. From this finite representation of L we then can easily obtain its lcp. Altogether, we arrive at a polynomial time algorithm.

Missing proofs can be found in the extended version of this article available on arxiv [11].

2 Preliminaries

Σ denotes a (finite) alphabet. We assume that Σ contains at least two letters as any context-free language over a unary alphabet is regular. Σ^* is the set of all finite words over Σ with ε the empty word, Σ^ω the set of all (countably) infinite words over Σ. We use (ω-)rational expressions to denote words and languages, e.g. $w^* = \varepsilon + w + ww + \ldots = \sum_{i \in \mathbb{N}_0} w^i$ and $w^\omega = wwwwwwwwwww\ldots$.

By $\mathsf{C}_\Sigma = \{(u, v) \in \Sigma^* \times \Sigma^*\}$ we denote the set of all pairs of finite words over Σ. We define a multiplication on C_Σ by $(x, \bar{x})(y, \bar{y}) := (xy, \bar{y}\bar{x})$. For $(x, \bar{x}) \in \mathsf{C}_\Sigma$ and $w \in \Sigma^*$ set $(x, \bar{x})w = xw\bar{x}$. As in the case of words, we set $(x, \bar{x})^0 := (\varepsilon, \varepsilon)$, $(x, \bar{x})^{k+1} := (x, \bar{x})(x, \bar{x})^k$ and $(x, \bar{x})^* := \sum_{k \geq 0} (x, \bar{x})^k$ for all $x, \bar{x} \in \Sigma^*$ and $k \in \mathbb{N}_0$.

Note that we slightly deviate from standard notation when it comes to the prefix order (i.e. $u < w$) and the common prefix (i.e. $u \wedge v$) of two words in order to avoid the clash with the notation for conjunction ($\wedge$): For $u, v \in \Sigma^*$ we write $u \sqsubseteq v$ ($u \sqsubset v$) to denote that u is a (strict) prefix of v, i.e. $v = uw$ for some $w \in \Sigma^*$ ($w \in \Sigma^+$). For $L \subseteq \Sigma^*$ (with $L \neq \emptyset$) its *longest common prefix* (lcp) $\bigsqcap L$ is given by the greatest lower bound of L w.r.t. this ordering. We simply write $u \sqcap v$ for $\bigsqcap\{u, v\}$. Note that for any word $w \in L$ there is at least one word $\alpha \in L$ s.t. $\bigsqcap L = w \sqcap \alpha$; we call any such α a *witness (w.r.t. w).* Note that $\sqcap$ is commutative and associative; concatenation distributes from the left over the lcp (i.e. $u(v \sqcap w) = uv \sqcap uw$); and the lcp is monotonically decreasing on the union of languages, i.e. $\bigsqcap(L \cup L') = (\bigsqcap L) \sqcap (\bigsqcap L')$. The lcp of infinite words is defined analogously.

A word $p \in \Sigma^*$ is called a power of a word q if $p \in q^*$; then q is called a root of p; if $p \neq \varepsilon$ is its own shortest root, p it is called *primitive.* Two words u, v are *conjugates* if the is a factorization $u = pq$ and $v = qp$. We recall two well-known results:

▶ **Lemma 1** (Commutative Words, [3]). *Let $u, v \in \Sigma^*$ be two words. If $uv = vu$, then $u, v \in p^*$ for some primitive $p \in \Sigma^*$.*

▶ **Lemma 2** (Periodicity Lemma of Fine and Wilf, [5]). *Let $u, v \in \Sigma^+$ be two non-empty words. If $|u^\omega \sqcap v^\omega| \geq |u| + |v| - \gcd(|u|, |v|)$, then $uv = vu$.*

Combining these two lemmata yields the following result which is a useful tool in the proofs to follow (see also lemma 3.1 in [3] for a more general version of this result):

▶ **Corollary 3.** *Let $u, v \in \Sigma^*$ with $uv \neq vu$.*
Then $u^\omega \sqcap v^\omega = uv \sqcap vu$ with $|uv \sqcap vu| < |u| + |v| - \gcd(|u|, |v|)$.

Proof. Since the bound of the size of $|uv \sqcap vu|$ follows from Lemma 2 we only have to show that $uv \sqcap vu = u^\omega \sqcap v^\omega$. If $|u| = |v|$, then $uv \neq vu$ implies $u \neq v$ and $uv \sqcap vu = u \sqcap v = u^\omega \sqcap v^\omega$.

W.l.o.g. we assume that $|u| < |v|$. As $uv \neq vu$, we have $\varepsilon \neq u$. Let $v \sqcap u^\omega = u^k u' \sqsubset u^{k+1}$ with $v = u^k u' v'$ and $u = u' u''$. It follows that $uv \sqcap vu = uu^k u' v' \sqcap u^k u' v' u = u^k(uu'v' \sqcap u'v'u) = u^k u'(u'' u' v' \sqcap v' u' u'')$.

If $v' \neq \varepsilon$, we have $u'' u' v' \sqcap v' u' u'' = u'' \sqcap v' = \varepsilon$, and thus $uv \sqcap vu = u^k u' = v \sqcap u^\omega = v^\omega \sqcap u^\omega$.

So assume $v' = \varepsilon$, i.e. $v \sqsubset u^\omega$ with $k > 0$ as $|u| < |v|$. As $uv = u^k u' u'' u' \neq u^k u' u' u'' = vu$, also $u' u'' \neq u'' u'$. Hence $uv \sqcap vu = u^k u'(u'' u' \sqcap u' u'') = u^{k+1} u \sqcap vv = u^\omega \sqcap v^\omega$, which concludes the proof. ◀

Here is a short example for the last corollary:

▶ **Example 4.** Let $u = aab$, $v = aaba = ua$. Then $uv \sqcap vu = aabaaba \sqcap aabaaab = aabaa = va$ and $u^\omega \sqcap v^\omega = aabaabaabu^\omega \sqcap aabaaabav^\omega = aabaa$ with $|aabaa| = |u| + |v| - \gcd(|u|, |v|) - 1$. I.e. the bound is sharp. Note that this example also shows, that even if $uv \neq vu$ and $\varepsilon \neq u \sqsubset v$, we still can have $v \sqsubset uv \sqcap vu$.

We briefly discuss properties of the lcp for very simple regular languages. These will be used several times in the proofs of Section 3 in order to bound the height of the derivation trees we need to consider:

▶ **Lemma 5.** *Let $y \neq \varepsilon$, then $w \sqcap yw = w \sqcap y^i w = \bigsqcap y^* w = w \sqcap y^\omega$ for all $i > 0$.*

Proof. Let $w \sqcap y^\omega = y^k y' \sqsubset y^{k+1}$ with $w = y^k y' w'$. Then for any $i > 0$ we have $w \sqcap y^i w = w \sqcap y^{k+i} y' w' = w \sqcap y^\omega$ where the last equality holds as $i > 0$ and $w \sqcap y^{k+1} = w \sqcap y^\omega \sqsubset y^{k+1}$. ◀

▶ **Lemma 6.** *If $w \not\sqsubseteq yw$, then $\bigsqcap y^* w = w \sqcap y^i w \sqsubset w$ for all $i > 0$.*

Proof. Since $w \not\sqsubseteq yw$, we have $w \neq \varepsilon$ and $y \neq \varepsilon$. By Lemma 5 we thus have $\bigsqcap y^* w = w \sqcap y^i w$ for any $i > 0$, in particular for $i = 1$. Define $w = y^k y' w'$ as in Lemma 5. As $w \not\sqsubseteq yw$, we have $w' \neq \varepsilon$ and thus $w \sqcap yw = y^k y' \sqsubset w$. ◀

We assume that the reader is familiar with context-free grammars (CFGs). We briefly introduce the notation we use for CFGs in the following. A context-free grammar G is given by a tuple $G = (\Sigma, V, P, S)$ where Σ is the alphabet of terminals, V is the set of nonterminals (also: variables), $P \subseteq V \times (V \cup \Sigma)^*$ is the set of production rules where a rule $p = (A, \gamma) \in P$ is also written as $A \to \gamma$, and S the axiom. The language generated by G is denoted by $L(G)$. G is *proper* if $A \to \varepsilon \notin P$ and $A \to B \notin P$ for all $A, B \in V$; G is in *Chomsky normal form (CNF)* if all rules are of the form $A \to a \in V \times \Sigma$ or $A \to BC \in V \to VV$. For every CFG G a proper CFG resp. a CFG in CNF G' can be constructed in time polynomial in the size of G such that $L(G) \setminus \{\varepsilon\} = L(G')$ [7]. As $\varepsilon \overset{?}{\in} L(G)$ is decidable in time polynomial in the size of G, and trivially $\bigsqcap L = \varepsilon$ if $\varepsilon \in L$, we will assume that $\varepsilon \notin L(G)$ and that G is proper from here on. For some proofs we assume in fact that G is in CNF but only in order to simplify notation.

3 LCP of a context-free language

Our main result in this section, Theorem 10, is that for every context-free language $L = L(G)$ generated by the given CFG G its lcp $\bigsqcap L$ is equal to the lcp of its *finite* sublanguage L' which contains only the words $w \in L$ which possess a derivation tree w.r.t. G whose height (considering only nonterminals) is at most four times the number of nonterminals of G. For the main result we require the following technical theorem (see the following example).

▶ **Theorem 7.** *Let $L = (x, \bar{x})[(y_1, \bar{y}_1) + \ldots + (y_l, \bar{y}_l)]^* w$ for $(x, \bar{x}), (y_1, \bar{y}_1), \ldots, (y_l, \bar{y}_l) \in \mathsf{C}_\Sigma$ and $w \in \Sigma^*$. Then:*

$$\bigsqcap L = \bigsqcap (x, \bar{x})[(y_1, \bar{y}_1)^{\leq 2} + \ldots + (y_k, \bar{y}_l)^{\leq 2}]w$$

Furthermore, if $\bigsqcap L = xw\bar{x} \sqcap xy^2 w\bar{y}^2\bar{x} \sqsubset xw\bar{x} \sqcap xyw\bar{y}\bar{x}$ for some $(y, \bar{y}) \in \{(y_1, \bar{y}_1), \ldots, (y_l, \bar{y}_l)\}$, then w.r.t. this y there exists some primitive $q \in \Sigma^*$ and some $k > 0$ such that

$$yw = wq^k \wedge q\bar{y} \neq \bar{y}q \wedge \bigsqcap L = xw\bar{x} \sqcap xywq\bar{y}\bar{x} \wedge xwq^k(\bar{y} \sqcap q^\omega) \sqsubseteq \bigsqcap L \sqsubset xwq^{k+1}(\bar{y} \sqcap q^\omega)$$

The proof of the main theorem of this section, Theorem 10, crucially depends on the observation that in the case $\bigsqcap L \sqsubset xw\bar{x} \sqcap xyw\bar{y}\bar{x}$, all the words y_i are powers of the same primitive word p with $pw = wq$ and all that is needed to obtain a witness is one additional power of p resp. its conjugate q (with $pw = wq$) to which Theorem 7 refers to. We give an example in order to clarify the statement of Theorem 7 in the case of $l = 2 \wedge y_1 y_2 = y_2 y_1$ which is central to Theorem 10:

▶ **Example 8.** We write $(y, \bar{y})$ for $(y_1, \bar{y}_1)$ and $(z, \bar{z})$ for $(y_2, \bar{y}_2)$, respectively. Let $(x, \bar{x}) = (\varepsilon, ababaaa) = (\varepsilon, qqaaa)$, $(y, \bar{y}) = (ab, abaab) = (q, qaab)$, $(z, \bar{z}) = (ab, abaac) = (q, qaac)$, and $w = \varepsilon$ with $q = ab = y = z$. We then have:

$$\begin{array}{lcl}
xw\bar{x} & = & ababaaa \\
xyw\bar{y}\bar{x} & = & ababaabababaaa \\
xzw\bar{z}\bar{x} & = & ababaacababaaa \\
xyyw\bar{y}\bar{y}\bar{x} & = & abababaababaabababaaa \\
xyzw\bar{z}\bar{y}\bar{x} & = & abababaacabaabababaaa \\
xzyw\bar{y}\bar{z}\bar{x} & = & abababaababaacababaaa \\
xzzw\bar{z}\bar{z}\bar{x} & = & abababaacabaacababaaa \\
x(y+z)^{\geq 3}\ldots & = & ababab\ldots \\
xywq\bar{y}\bar{x} & = & abababaabababaaa \\
xzwq\bar{z}\bar{x} & = & abababaacababaaa \\
\hline
\bigsqcap L & = & ababa
\end{array}$$

So in this example, any word except for $xyw\bar{y}\bar{x}$ and $xzw\bar{z}\bar{x}$ is a witness for the lcp w.r.t. $xw\bar{x}$. W.r.t. the proof of Theorem 10 it is important that also in general we can pick a witness which either is derived using only $(y,\bar{y})$ or $(z,\bar{z})$ but not both, and that we need to use $(y,\bar{y})$ resp. $(z,\bar{z})$ at most twice in order to get one additional copy of the conjugate q of the primitive root of both y and z.

To give an impression of the proof of Theorem 7 we show the case $l = 1$. The complete proof of Theorem 7 can be found in the appendix of [11].

▶ **Lemma 9.** *Let $L = (x,\bar{x})(y,\bar{y})^*w$. Then: $\bigsqcap L = \bigsqcap(x,\bar{x})(y,\bar{y})^{\leq 2}w$.*
If $\bigsqcap L \sqsubset xw\bar{x} \sqcap xyw\bar{y}\bar{x}$, then there is some primitive q and some $k > 0$ s.t.

$$yw = wq^k \wedge q\bar{y} \neq \bar{y}q \wedge \bigsqcap L = xw\bar{x} \sqcap xywq\bar{y}\bar{x} \wedge xwq^k(\bar{y} \sqcap q^\omega) \sqsubseteq \bigsqcap L \sqsubset xwq^{k+1}(\bar{y} \sqcap q^\omega)$$

Proof. Recall that for any $z \in L$ there is some *witness* $z' \in L$ s.t. $\bigsqcap L = z \sqcap z'$. Our main goal is to show that w.r.t. $xw\bar{x}$ we find a witness within $\{xy^iw\bar{y}^i\bar{x} \mid i = 0,1,2\}$. What makes the proof technically more involved is that for Theorem 10 we need a stronger characterization of the case when $xyyw\bar{y}\bar{y}\bar{x}$ is the only witness in this set.

If $y = \varepsilon \vee \bar{y} = \varepsilon$, then L is actually regular and Lemma 5 already tells us that $xyw\bar{y}\bar{x}$ is a witness (w.r.t. $xw\bar{x}$). So wlog. $y \neq \varepsilon \neq \bar{y}$. If $w \not\sqsubseteq yw$, then $\bigsqcap y^*w = w \sqcap yw \sqsubset w$ by Lemma 6 and thus $\bigsqcap L = x(w \sqcap yw)$, i.e. $xyw\bar{y}\bar{x}$ is again a witness.

From now on we assume that $w \sqsubseteq yw$. Then there is some conjugate μ of y defined by $w\mu = yw$, and xw is a prefix of $\bigsqcap L$ as $xy^iw\bar{y}^i\bar{x} = xw\mu^i\bar{y}^i\bar{x}$. Wlog. we therefore assume $xw = \varepsilon$ from now on so that L becomes $\{y^i\bar{y}^i\bar{x} \mid i \in \mathbb{N}_0\}$.

Let q be the primitive root of y s.t. $y = q^k$ for a suitable $k > 0$ (as $y \neq \varepsilon$). By choosing $j > |\bar{x}| \,/\, |y|$ we obtain $\bigsqcap L \sqsubseteq \bar{x} \sqcap y^j\bar{y}^j\bar{x} = \bar{x} \sqcap q^{kj} \sqsubset q^\omega$, i.e. $\bigsqcap L \sqsubset q^\omega$. We therefore factorize $\bar{x}$ and $\bar{y}$ w.r.t. q^ω: Let $\bar{x} = q^nq'\bar{x}'$ with $\bar{x} \sqcap q^\omega = q^nq' \sqsubset q^{n+1}$; and let $\bar{y} = q^{k'}\hat{q}\bar{y}'$ with $\bar{y} \sqcap q^\omega = q^{k'}\hat{q} \sqsubset q^{k'+1}$. The words of L have thus the form $y^i\bar{y}^i\bar{x} = q^{ik}\left(q^{k'}\hat{q}\bar{y}'\right)^i q^nq'\bar{x}'$.

If q (resp. y) and $\bar{y}$ commute, then $\bar{y} = q^{k'}$ by Lemma 1 (as q is primitive) for some suitable $k' \in \mathbb{N}$. Then $L = (y\bar{y})^*\bar{x} = (q^{k+k'})^*q^nq'\bar{x}'$ with $\bigsqcap L = q^nq'$, and $y\bar{y}\bar{x}$ is again a witness w.r.t. $\bar{x}$. We thus also assume $q\bar{y} \neq \bar{y}q$ from here on.

If $q^nq' \sqsubseteq q^{k+k'}\hat{q}$, then $\bigsqcap L \sqsubseteq q^nq'$ and $qy\bar{y}\bar{x}$ is a witness w.r.t. $\bar{x}$: by choice of n we have $\bar{x} \sqcap q^\omega = \bar{x} \sqcap q^{n+1}$, by $q^nq' \sqsubseteq q^{k+k'}\hat{q}$ we also have $q^{n+1} \sqsubseteq q^{k+k'+1}$; from this we obtain $\bar{x} \sqcap qy\bar{y}\bar{x} = \bar{x} \sqcap q^{k+k'+1}\hat{q}\bar{x} = \bar{x} \sqcap q^{n+1} = q^nq'$. Thus, also $yy\bar{y}\bar{y}\bar{x}$ is a witness w.r.t $\bar{x}$. Assume now that $q^{k+k'}\hat{q} \sqsubset q^nq'$ and thus $q^{k+k'}\hat{q} \sqsubseteq \bigsqcap L$. If $\bigsqcap L = q^{k+k'}\hat{q}$, then $\bar{x} \sqcap y\bar{y}\bar{x} = q^{k+k'}\hat{q}$ has to hold, i.e. $y\bar{y}\bar{x}$ has to be a witness. Thus assume $q^{k+k'}\hat{q} \sqsubset \bigsqcap L$. If $\bar{y}' \neq \varepsilon$, then, as $q^{k+k'}\hat{q} \sqsubset q^nq'$, we have that $q^nq' \sqcap q^{k+k'}\hat{q}\bar{y}' = q^{k+k'}\hat{q}$ so that $y\bar{y}\bar{x}$ is again a witness. Hence assume $\bar{y}' = \varepsilon$ resp. $\bar{y} = q^{k'}\hat{q}$ for the remaining. As q and $\bar{y}$ do not commute, also q and $\hat{q}$ do not commute implying $q\hat{q} \sqsubset \hat{q}\,q \sqsubset q\hat{q}$. Thus

$$\begin{aligned} q^{k+k'}\hat{q} \sqsubset \bigsqcap L \sqsubseteq y\bar{y}\bar{x} \sqcap yy\bar{y}\bar{y}\bar{x} \quad &= \quad q^{k+k'}(\hat{q}q^nq'\bar{x}' \sqcap q^k\hat{q}\bar{y}\bar{x}) \\ &\overset{n \geq k > 0 \wedge \hat{q} \sqsubset q}{=} \quad q^{k+k'}(\hat{q}q \sqcap q\hat{q}) \sqsubset q^{k+k'}q\hat{q} \end{aligned}$$

That is either $y\bar{y}\bar{x}$ or $yy\bar{y}\bar{y}\bar{x}$ has to be a witness w.r.t. $\bar{x}$ as $\bigsqcap L \sqsubset q^\omega$ and as we can extend $q^{k+k'}\hat{q}$ by at most $|q| - 1$ symbols, i.e. we need at most one additional copy of q which is again given by $yyw\bar{y}\bar{y}\bar{x}$ as $k > 0$. In particular, we have again that, if $yy\bar{y}\bar{y}\bar{x}$ is a witness, then so is $qy\bar{y}\bar{x}$. ◀

Using Theorem 7, we now can show that we only need to consider a finite sublanguage of L instead of L itself:

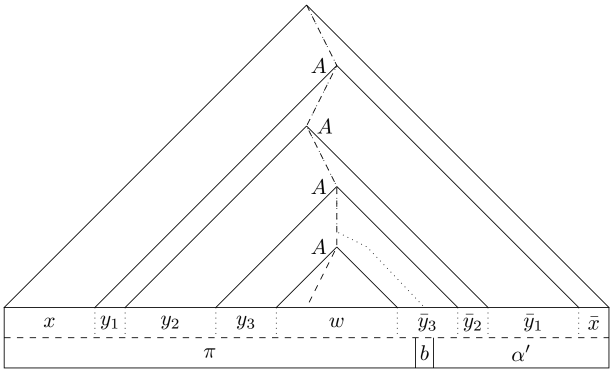

Figure 1 Factorization of a witness $\alpha = (x,\bar{x})(y_1,\bar{y}_1)(y_2,\bar{y}_2)(y_3,\bar{y}_3)w = \pi b\alpha'$ w.r.t. a nonterminal A occurring at least four times a long the dashed path in a derivation tree of α leading to a letter either within the lcp $\pi = \bigsqcap L$ or to the lcp-defining letter b (the leaf of the dotted path).

▶ **Theorem 10.** *Let $L = L(G)$ be given by a proper CFG $G = (\Sigma, V, P, S)$. Let $\hat{L} \subseteq L$ be the finite language of all words of L for which there is a derivation tree w.r.t. G of height*[2] *at most $4N$ with $N = |V|$. Then: $\bigsqcap L = \bigsqcap \hat{L}$.*

Proof. Let N be the number of nonterminals of G. Let $\sigma \in L$ be a shortest word, and $\alpha \in L$ a shortest word with $\bigsqcap L = \sigma \sqcap \alpha$. Set $\pi := \bigsqcap L$.

We claim that there is at least one such α (for any fixed σ) that has an derivation tree w.r.t. G of height less than $4N$.If $\sigma = \alpha$, we are done as σ has a derivation tree of height less than N. So assume $\sigma \neq \alpha$ s.t. $\sigma = \pi a\sigma'$ and $\alpha = \pi b\alpha'$ with $a \neq b$ and $a, b \in \Sigma$. Then fix any derivation tree t of α w.r.t. G.

In fact, we will show the stronger claim that any path from the root of t to any letter of πb has length at most $3N$ (i.e. all the paths leading to the separating letter b or a letter left of it, see Figure 1); note that any path that leads to a letter right of b (i.e. into α') has to enter a subtree of height less than N as soon as it leaves the path leading to b because of the minimality of α. Hence, if all the paths leading to b or a letter left of b have length less than $3N$, the longest path in the derivation tree must have length at most $4N$.

So assume for the sake of contradiction that there is a path leading to a letter within πb that has at least length $3N$ i.e. consists of at least $3N + 1$ nonterminals. Then there is one nonterminal A that occurs at least four times leading to a factorization

$$\alpha = (x,\bar{x})(y_1,\bar{y}_1)(y_2,\bar{y}_2)(y_3,\bar{y}_3)w$$

Note that $x\bar{x} \neq \varepsilon$, $y_i\bar{y}_i \neq \varepsilon$ $(i = 1, 2, 3)$, and $w \neq \varepsilon$ as G is proper. As this path ends at b or left of it, we have $xy_1y_2y_3 \sqsubseteq \pi$. With $(x,\bar{x})(y_i,\bar{y}_i)(y_j,\bar{y}_j)w \in L$ for any $i, j \in \{1, 2, 3\}$ we thus obtain that $xy_iy_j \sqsubseteq \pi$ and $xy_jy_i \sqsubseteq \pi$ and thus $y_iy_j = y_jy_i$ for all $i, j \in \{1, 2, 3\}$. So $y_i = p^{k_i}$ for the same primitive p using Lemma 1.

Let $L' = (x,\bar{x})[(y_1,\bar{y}_1) + (y_2,\bar{y}_2) + (y_3,\bar{y}_3)]^*w$ so that $\{xw\bar{x}, \alpha\} \subseteq L'$. By construction $L' \subseteq L$ and thus $\bigsqcap L \sqsubseteq \bigsqcap L' \sqsubseteq xw\bar{x} \sqcap \alpha$. As $xw\bar{x}$ is shorter than α, it cannot be a witness, so $\pi a \sqsubseteq xw\bar{x}$ and $\pi = xw\bar{x} \sqcap \alpha$. Hence

$$\bigsqcap L = \sigma \sqcap \alpha = \pi = xw\bar{x} \sqcap \alpha \sqsupseteq \bigsqcap L' \sqsupseteq \bigsqcap L \quad \text{i.e.} \quad \bigsqcap L = \bigsqcap L'$$

[2] We measure the height of a derivation tree only w.r.t. nonterminals along a path from the root to a leaf.

It therefore suffices to consider L' in the following; in particular, α has to be a witness w.r.t. $xw\bar{x}$ of minimal length, too. (From here on, *witness* will always be w.r.t. $xw\bar{x}$.) By virtue of Theorem 7 we have $\bigsqcap L' = \bigsqcap(x,\bar{x})[(y_1,\bar{y}_1)^{\leq 2} + (y_2,\bar{y}_2)^{\leq 2} + (y_3,\bar{y}_3)^{\leq 2}]w$. Note that $\bigsqcap L' \sqsubset xw\bar{x} \sqcap xy_iw\bar{y}_i\bar{x}$ for any $i = 1,2,3$ as $|xy_iw\bar{y}_i\bar{x}| < |\alpha|$ and thus $xy_iw\bar{y}_i\bar{x}$ cannot be a witness by minimality of α. So for some $I \in \{1,2,3\}$

$$\bigsqcap L' = xw\bar{x} \sqcap xy_Iy_Iw\bar{y}_I\bar{y}_I\bar{x} \sqsubseteq \alpha$$

i.e. $xy_Iy_Iw\bar{y}_I\bar{y}_I\bar{x}$ has to be also a witness. Set $(y,\bar{y}) := (y_I,\bar{y}_I)$ and $L'' = (x,\bar{x})(y,\bar{y})^*w$ so that $L'' \subseteq L' \subseteq L$ and $\bigsqcap L = \bigsqcap L' = \bigsqcap L''$ as

$$xw\bar{x} \sqcap xyyw\bar{y}\bar{y}\bar{x} = \bigsqcap L \sqsubseteq \bigsqcap L' \sqsubseteq \bigsqcap L'' \sqsubseteq xw\bar{x} \sqcap xyyw\bar{y}\bar{y}\bar{x} \sqsubset xyw\bar{y}\bar{x}$$

As $xyw\bar{y}\bar{x}$ is not a witness, Theorem 7 tells us that there is some q satisfying

$$yw = wq^k \wedge q\bar{y} \neq \bar{y}q \wedge \bigsqcap L = \bigsqcap L'' = xw\bar{x} \sqcap xywq\bar{y}\bar{x} \wedge xwq^k(\bar{y} \sqcap q^\omega) \sqsubseteq \bigsqcap L \sqsubset xwq^{k+1}(\bar{y} \sqcap q^\omega)$$

From this, we obtain: **1.** As we already know that $y_i = p^{k_i}$ (as they commute), it follows that p and q are conjugates with $pw = qw$ s.t. $y_iw = wq^{k_i}$. **2.** As $xwq^k \sqsubseteq \bigsqcap L \sqsubset xwq^\omega$, we find some $m \geq 0$ and $\dot{q} \sqsubset q$ s.t. $\pi = \bigsqcap L = xwq^kq^m\dot{q}$ and, thus, $\pi a = xwq^kq^m\dot{q}a \sqsubseteq xw\bar{x}$ and $\pi b = xwq^kq^m\dot{q}b \sqsubseteq xyyw\bar{y}\bar{y}\bar{x}$. (Here, b might change, yet it cannot become a as $xyyw\bar{y}\bar{y}\bar{x}$ is a witness.) Additionally, from $\pi = xw\bar{x} \sqcap xyyw\bar{y}\bar{y}\bar{x} \sqsubset xwq^{k+1}(\bar{y} \sqcap q^\omega)$ we obtain $\pi c \sqsubseteq xwq^{k+1}(\bar{y} \sqcap q^\omega)$, i.e. $q^m\dot{q}c \sqsubseteq q\bar{y} \sqcap q^\omega \sqsubset q^\omega$ and thus $\dot{q}c \sqsubseteq q$. Hence, any word with prefix $xwq^{k+1}(\bar{y} \sqcap q^\omega)$ is a witness.

If there was at least one $j \in \{1,2,3\} \setminus \{I\}$ with $k_j > 0$ s.t. $y_j = p^{k_j} \neq \varepsilon$, then $(x,\bar{x})(y_j,\bar{y}_j)(y,\bar{y})w$ would be a witness shorter than α as y_j would give us at least one copy of q:

$$\begin{array}{rcll} (x,\bar{x})(y_j,\bar{y}_j)(y,\bar{y})w & = & xy_jyw\bar{y}\bar{y}_j\bar{x} & \\ & \sqsupseteq & xwq^{k+k_j}\bar{y} & (\text{as } yw = wq^k \text{ and } y_jw = wq^{k_j}) \\ & \sqsupseteq & xwq^{k+k_j}(\bar{y} \sqcap q^\omega) & \\ & \sqsupseteq & xwq^{k+1}(\bar{y} \sqcap q^\omega) & (\text{as } k_j > 0 \text{ and } q^{k+1}(\bar{y} \sqcap q^\omega) \sqsubset q^\omega) \end{array}$$

So for all remaining $j \in \{1,2,3\} \setminus \{I\}$ we have $y_j = \varepsilon$ and thus $\bar{y}_j \neq \varepsilon$ as G is proper and thus $y_j\bar{y}_j \neq \varepsilon$. By Lemma 5 $\bigsqcap xw\bar{y}_j^*\bar{x} = xw\bar{x} \sqcap xw\bar{y}_j\bar{x}$, hence $\pi a \sqsubseteq xw\bar{y}_j^*\bar{x}$, i.e. $q^{k+m}\dot{q}a \sqsubseteq \bar{y}_j^\omega$. If $q^m\dot{q}b \sqsubseteq \bar{y}_j$ for some $j \in \{1,2,3\} \setminus \{I\}$ (recall $\dot{q}b \sqsubseteq q$), then as $a \neq b$

$$xw\bar{x} \sqcap (x,\bar{x})(y,\bar{y})(y_j,\bar{y}_j)w \overset{(\text{as } y_j = \varepsilon)}{=} xw(\bar{x} \sqcap q^k\bar{y}_j\bar{y}\bar{x}) = xw(q^{k+m}\dot{q}a \sqcap q^{k+m}\dot{q}b) = \pi$$

i.e. $xyy_jw\bar{y}_j\bar{y}\bar{x}$ would be a shorter witness than α. Hence $\bar{y}_j \sqsubseteq q^m\dot{q} \sqsubset q^{k+m}\dot{q}a$ for both $j \in \{1,2,3\} \setminus \{I\}$. Thus:

$$\left|q^\omega \sqcap \bar{y}_j^\omega\right| \geq \left|q^{k+m}\dot{q}\right| \geq |q| + |q^m\dot{q}| > |q| + |\bar{y}_j| - \gcd(|q|, |\bar{y}_j|)$$

By the periodicity lemma of Fine and Wilf (Lemma 2) this implies $\bar{y}_j = q^{k'_j}$ for some $k'_j > 0$ (as q primitive), and, subsequently as the final contradiction, that $xy_Iy_jw\bar{y}_j\bar{y}_I\bar{x}$ would be a shorter witness. ◀

4 Small Equivalent Subsets of Languages

In this section we formally introduce a notion of equivalence of languages w.r.t. longest common prefixes. The first main result of this section is that every non-empty language has an equivalent subset consisting of at most three elements. In case of acyclic context-free languages, such a subset can be computed in polynomial time. In combination with Theorem 10, we can lift the restriction on acyclicity. This enables us to ultimately conclude that the longest common prefix of a context-free language can be computed in polynomial time.

▶ **Definition 11.** Two languages L, L' are *equivalent w.r.t the* lcp *(short: $L \equiv L'$)* iff $\bigsqcap(Lw) = \bigsqcap(L'w)$ for all words $w \in \Sigma^*$.

We observe that L is equivalent to L' w.r.t. the lcp also after union or concatenation from the left or right with arbitrary other languages. Formally, this amounts to the following properties:

▶ **Lemma 12.** *For all non-empty languages $L, L', \hat{L}$ with $L \equiv L'$ we have:*

1. $\bigsqcap(L\hat{L}) = \bigsqcap(L'\hat{L})$
2. $\bigsqcap(\hat{L}L) = \bigsqcap(\hat{L}L')$
3. $\bigsqcap(L \cup \hat{L}) = \bigsqcap(L' \cup \hat{L})$

Proof. The argument is as follows:

1. $\bigsqcap(L\hat{L}) \;=\; \bigsqcap_{w\in\hat{L}}(\bigsqcap(Lw)) \;=\; \bigsqcap_{w\in\hat{L}}(\bigsqcap(L'w)) \;=\; \bigsqcap(L'\hat{L})$;
2. $\bigsqcap(\hat{L}L) \;=\; \bigsqcap(\hat{L}(\bigsqcap L)) \;=\; \bigsqcap(\hat{L}(\bigsqcap L')) \;=\; \bigsqcap(\hat{L}L')$;
3. $\bigsqcap(L \cup \hat{L}) \;=\; \bigsqcap L \sqcap \bigsqcap \hat{L} \;=\; \bigsqcap L' \sqcap \bigsqcap \hat{L} \;=\; \bigsqcap(L' \cup \hat{L})$. ◀

The next lemma gives us an explicit formula for $\bigsqcap(Lw)$ for the special case of the two-element language $L = \{u, uv\}$.

▶ **Lemma 13.** *Assume that $u, v \in \Sigma^*$ with $v \neq \epsilon$. For all words $w \in \Sigma^*$, $\bigsqcap(\{u, uv\}w) = u(w \sqcap v^\omega)$ holds.*

Proof. $\bigsqcap(\{u, uv\}w) = uw \sqcap uvw$. If w and v are incomparable or w is a prefix of v, $w \sqcap vw = w \sqcap v = w \sqcap v^\omega$, and the claim follows. Thus, it remains to consider the case that $v \sqsubseteq w$. Then $w = v^i w'$ for some i so that v is no longer a prefix of w'. Then $\bigsqcap(\{u, uv\}w) = \bigsqcap(\{u, uv\}v^i w') = uv^i(w' \sqcap vw') = uv^i(w' \sqcap v^\omega) = u(w \sqcap v^\omega)$. ◀

The explicit formula from Lemma 13 can be used to identify small equivalent sublanguages.

▶ **Theorem 14.** *For every non-empty language $L \subseteq \Sigma^*$ there is a language $L' \subseteq L$ consisting of at most three words such that $L \equiv L'$.*

Proof. If L is a singleton language, we choose $L' = L$. So assume that L contains at least two words with lcp u. If the lcp u of L is not contained in L then we choose L' as consisting of the two minimal words w_1, w_2 so that $u = w_1 \sqcap w_2$. It remains to consider the case where the lcp u of L is contained in L. Then we have for each word $w \in \Sigma^*$,

$$\begin{array}{rcll} \bigsqcap(Lw) & = & \bigsqcap(\{uv \mid uv \in L\}w) & \\ & = & \bigsqcap\{\bigsqcap(\{u, uv\}w) \mid uv \in L, v \neq \epsilon\} & \\ & = & \bigsqcap\{u(w \sqcap v^\omega) \mid uv \in L, v \neq \epsilon\} & \text{(Lemma 13)} \\ & = & u(w \sqcap \bigsqcap\{v^\omega \mid uv \in L, v \neq \epsilon\}) & \end{array} \tag{1}$$

If L is ultimately periodic, then all words in L are of the form uv_0^i for some $v_0 \in \Sigma^+$ and $i \geq 0$, and $(v_0^i)^\omega = v_0^\omega$. Thus, $\bigsqcap(Lw) = u(w \sqcap v^\omega)$ for any $uv \in L$ with $v \neq \epsilon$. Hence, $L \equiv L' = \{u, uv\}$ for any such v.

If L is not ultimately periodic, then we choose words $uv_1, uv_2 \in L$ so that the lcp of v_1^ω and v_2^ω has minimal length. Then

$$\begin{aligned} \bigsqcap(\{u, uv_1, uv_2\}w) &= u(w \sqcap v_1^\omega \sqcap v_2^\omega) \\ &= u(w \sqcap \bigsqcap\{v^\omega \mid uv \in L, v \neq \epsilon\}) \end{aligned}$$

by the minimality of $v_1^\omega \sqcap v_2 \omega$. Therefore, $L \equiv L' = \{u, uv_1, uv_2\}$. ◀

Since for any non-empty words w_1, w_2 given by SLPs, an SLP for $w_1^\omega \sqcap w_2^\omega = w_1w_2 \sqcap w_2w_1$ (if $w_1 \neq w_2$) can be computed in polynomial time[3], we have:

▶ **Corollary 15.** *For every non-empty finite $L \subseteq \Sigma^*$ consisting of words each of which is represented by an SLP, a subset $L' \subseteq L$ consisting of at most three words can be calculated in polynomial time such that $L \equiv L'$.*

Proof. The proof distinguishes the same cases as in the proof of Theorem 14 and relies on polynomial algorithms on SLPs [10]. If L contains at most three words we are done. Since the words in L are given as SLPs, we can calculate (a SLP for) the lcp u of the words in L. Next, we determine whether u is in L. This can again be checked in polynomial time. If this is not the case, then we can select two words $w_1, w_2 \in L$ so that $u = w_1 \sqcap w_2$ giving us $L' = \{w_1, w_2\}$ in polynomial time. So, now assume that u is in L. Next, we check whether or not L is ultimately periodic, i.e., whether for any non-empty words v_1, v_2 with $uv_1, uv_2 \in L$, $v_1^\omega = v_2^\omega$. By Lemma 2 this is the case iff $v_1v_2 = v_2v_1$. The latter can be checked in polynomial time as concatenation and equality of SLPs can be calculated in polynomial time. If this is the case, then we obtain $L' = \{u, uv\}$ for some $uv \in L$ with $v \neq \epsilon$ in polynomial time.

It remains to consider the case where the lcp u is contained in L and L is not ultimately periodic. Then we need to determine words uv_1 and uv_2 in L with $v_1 \neq \epsilon \neq v_2$ such that $v_1^\omega \sqcap v_2^\omega$ has minimal length. Since $v_1^\omega \sqcap v_2^\omega = v_1v_2 \sqcap v_2v_1$ (see Corollary 3), such a pair can be computed in polynomial time as well. Therefore, $L' = \{u, uv_1, uv_2\}$ can be computed in polynomial time. ◀

The following lemma explains that equivalence of two non-empty languages of cardinalities at most 3 can be decided in polynomial time.

▶ **Lemma 16.** *Let $L_1, L_2 \subseteq \Sigma^*$ denote non-empty languages consisting of at most three words each, which are all given by SLPs. Then $L_1 \overset{?}{\equiv} L_2$ can be decided in polynomial time.*

Proof. If one of the two languages contains just a single word, then $L_1 \equiv L_2$ iff $L_1 = L_2$ — which can be decided in polynomial time. Otherwise, we first compute $\bigsqcap L_1$ and $\bigsqcap L_2$. If these differ, then by definition L_1 cannot be equivalent to L_2. Therefore assume now that $u = \bigsqcap L_1 = \bigsqcap L_2$ is the common lcp .

Obviously, L_i and $L_i \cup \{u\}$ are equivalent w.r.t. the lcp ($i = 1, 2$). Thus, for testing equality, we may add u to L_1 resp. L_2, if it is missing, and reduce L_1 resp. L_2 subsequently to languages of at most three words.

[3] Lohrey [10] gives an overview over the classical algorithms for SLPs. The fully compressed pattern matching problem for SLPs is in PTIME [10, Theorem 12], i.e. we can test whether one SLP is a factor of another SLP. Especially we can test whether one SLP is a prefix of another SLP. As we can build an SLP for any prefix of an SLP in polynomial time we can use a binary search to compute the lcp of two SLPs in polynomial time.

From Equation 1 follows that $L_1 \equiv L_2$ if $\bigsqcap\{v_1^\omega \mid uv_1 \in L_1, v_1 \neq \epsilon\} = \bigsqcap\{v_2^\omega \mid uv_2 \in L_2, v_2 \neq \epsilon\}$. This is the case if either $v_1^\omega = v_2^\omega$ for all $uv_1 \in L_1$ and $uv_2 \in L_2$ or for $uv_i, uv_i' \in L_i$, $v_i \neq \epsilon \neq v_i'$ with $w_i = v_i^\omega \sqcap v_i'^\omega$ is minimal for L_i $(i = 1, 2)$, $w_1 = w_2$ holds.

In the first case $v_1^\omega = v_2^\omega$ for all $uv_1 \in L_1$ and $uv_2 \in L_2$ can be checked in polynomial time according to the periodicity lemma of Fine and Wilf (cf. Corollary 3). In the second case w_1, w_2 can be computed and compared in polynomial time as all words are given as SLPs. Thus, we ultimately arrive at a polynomial time decision procedure. ◀

▶ Remark. Note that in light of the equivalence test, we can choose distinct letters $a, b \in \Sigma$, and equivalently replace the language $L_1 = \{uv_1, uv_2\}$ with $L_1' = \{ua, ub\}$ whenever $v_1 \neq \epsilon \neq v_2$ and $v_1 \sqcap v_2 = \epsilon$, and the language $L_2 = \{u, uv_1, uv_2\}$ by the language $L_2' = \{u, uwa, uwb\}$ whenever $w = v_1v_2 \sqcap v_2v_1 \neq v_1v_2$ holds. This reduced representation allows for an easier computation.

Now we have all pre-requisites to prove the main theorem of our paper.

▶ **Theorem 17.** *Assume that G is a proper context-free grammar with $L = L(G)$ non-empty. Then the longest common prefix of L can be calculated in polynomial time.*

Proof. Assume w.l.o.g. that G is a CFG in Chomsky normal form as this simplifies the notation. For the actual fixed-point iteration this is not required. Then we calculate $\bigsqcap L(G)$ as follows. We build (implicitly, see the following remark) an acyclic CFG $\hat{G}$ in polynomial time such that $L(\hat{G})$ consists of all words of $L(G)$ for which there is a derivation tree of height at most $4N$ where N is the number of nonterminals in G. To this end, we tag the variables with a counter that bounds the height of the derivation trees. In more detail, for every rewriting rule $A \to BC$ of G and every $i \in \{1, \dots, 4N\}$ we add to $\hat{G}$ the rule $A^{(i)} \to B^{(i-1)}C^{(i-1)}$, and for every rule $A \to a$ of G and every $i \in \{0, 1, \dots, 4N\}$ we add the rule $A^{(i)} \to a$ to $\hat{G}$. In a derivation tree w.r.t. $\hat{G}$ every path starting at some node labeled by $A^{(i)}$ has thus length at most i as i has strictly decreases when moving down to towards the leaves, hence, a node labeled by $A^{(i)}$ can only be the root of a (sub-)tree of height at most i. Further, every derivation tree of $\hat{G}$ becomes a derivation tree of G by simply replacing $A^{(i)}$ by A. As every rule of G is copied at most $4N + 1$ times with N the number of nonterminals of G, the size of $\hat{G}$ grows at most quadratically with the size of G. In particular, $\hat{G}$ is still proper and in CNF. For more details, see e.g. section 3 in [4].

By Theorem 10, we know that $\bigsqcap L(G) = \bigsqcap L(\hat{G})$. By construction, $\hat{G}$ is also in Chomsky normal form. For i from 0 to (at most) $4N$ (with N still the number of variables of the original grammar G – as $\hat{G}$ is acyclic we only need to compute $[A^{(i)}]$ once when proceeding bottom-up), we then compute in every iteration for every nonterminal $A^{(i)}$ (for the currently value of i) first the language

$$[A^{(i)}]' := \{a \in \Sigma^* \mid A^{(0)} \to a \in P\} \cup \bigcup_{A \to BC \in G} [B^{(i-1)}] \cdot [C^{(i-1)}]$$

By induction on i, we may assume that the languages $[B^{(i-1)}], [C^{(i-1)}]$ (a) have already been computed, (b) consist of at most three words, and (c) every word is given as an SLP. Note that the cardinality of every language $[A^{(i)}]'$ is polynomial in the size of G. By virtue of Corollary 15, we therefore can reduce $[A^{(i)}]'$ in polynomial time to a language $[A^{(i)}] \subseteq [A^{(i)}]'$ with $[A^{(i)}] \equiv [A^{(i)}]'$ and $\left|[A^{(i)}]\right| \leq 3$. By construction, we then have

$$[A^{(i)}] \quad \equiv \quad \{w \in \Sigma^* \mid A^{(i)} \Rightarrow^* w\}$$

Since $\hat{G}$ has polynomially many nonterminals only, the overall algorithm runs in polynomial time. ◀

▶ Remark. Note that we can drop the assumption that the grammars G and likewise $\hat{G}$ are in Chomsky normal form if the right-hand sides of all rules have bounded lengths. Then the cardinality of the languages $[A^{(i)}]'$ are still polynomial. Further, instead of spelling out the grammar $\hat{G}$ explicitly, we may perform a round robin fixpoint iteration where in every round we first compute

$$[A]' := \bigcup_{A \to w_1 B_1 w_2 B_2 \ldots w_k B_k w_{k+1}} \{w_1\} \cdot [B_1] \cdot \{w_2\} \cdot [B_2] \cdots \{w_k\} \cdot [B_k] \cdot \{w_{k+1}\}$$

with initially $[A] := \{w \in \Sigma^* \mid A \to w \in G\}$, then updating $[A]$ so that $[A] \subseteq [A]'$ with $[A] \equiv [A]'$ and $|[A]| \leq 3$. Theorem 10 guarantees that the lcp is attained after at most $4N$ iterations. Using standard approaches like work lists, we only need to recompute $[A]$ if there is some rule $A \to \gamma B \delta$ in G and $[B]$ has changed since the last recomputation of $[A]$. As shown in Lemma 16 we can easily check if $[B] \not\equiv [B]'$ in every round and accordingly insert A into the work list.

We demonstrate this simplified version of the algorithm described in Theorem 17 by an example.

▶ **Example 18.** Consider the following grammar G with the following rules:

$$S \to Aababaac \mid ababaac \qquad A \to ab\,A\,abaab \mid ab\,A\,abaac \mid ababaab \mid ababaac$$

The round robin fixpoint iteration would proceed by iteratively evaluating the equations

$$\begin{array}{lcl} [A]' & := & \{abwabaab, abwabaac, ababaab, ababaac \mid w \in [A]\} \\ [S]' & := & \{wababaac, ababaac \mid w \in [A]\} \end{array}$$

and recomputing the languages $[A]$ and $[S]$ so that $[A] \equiv [A]'$ and $[S] \equiv [S]'$ and both $[A]$ and $[S]$ consist of at most three words where we further reduce the words of $[A]$ and $[S]$ as described in the remark following Lemma 16. As $[A]$ does not depend on $[S]$, we can postpone the computation of $[S]$ until after $[A]$ has converged. In the first round, we have:

$$[A]' = \{ababaab, ababaac\}$$

and thus update $[A]$ to $[A] := \{(ab)^2aab, (ab)^2aac\}$. For the second round, we obtain

$$\begin{aligned} [A]' = ab\{(ab)^2aab, (ab)^2aac\}abaab \cup ab\{(ab)^2aab, (ab)^2aac\}abaac \cup \{(ab)^2aab, (ab)^2aac\} \\ \equiv \{(ab)^3a(ab)^2aab, (ab)^2aab\} \equiv \{(ab)^3, (ab)^2aa\} =: [A] \end{aligned}$$

which is already the fixpoint as an additional iteration would show. Therefore we obtain

$$\begin{array}{rl} [S]' = & \{(ab)^3, (ab)^2aa\}(ab)^2aac \cup \{(ab)^2aac\} \\ \equiv & \{(ab)^3(ab)^2aac, (ab)^2aac\} \equiv \{(ab)^3, (ab)^2aa\} =: [S] \end{array}$$

So $\bigsqcap L = (ab)^3 \sqcap (ab)^2aa = (ab)^2a$.

5 Conclusion

We have shown that the longest common prefix of a non-empty context-free language can be computed in polynomial time. This result was based on two structural results, namely, that it suffices to consider words with derivation trees of bounded height, and second that each non-empty language is equivalent to a sublanguage consisting of at most three elements. For the actual algorithm, we relied on succinct representations of long words by means of SLPs. It remains as an intriguing open question whether the presented method can be generalized to more expressive grammar formalisms.

References

1 Adrien Boiret. Normal form on linear tree-to-word transducers. In Adrian-Horia Dediu, Jan Janousek, Carlos Martín-Vide, and Bianca Truthe, editors, *Language and Automata Theory and Applications - 10th International Conference, LATA 2016, Prague, Czech Republic, March 14-18, 2016, Proceedings*, volume 9618 of *Lecture Notes in Computer Science*, pages 439–451. Springer, 2016. doi:10.1007/978-3-319-30000-9_34.

2 Adrien Boiret and Raphaela Palenta. Deciding equivalence of linear tree-to-word transducers in polynomial time. In Srecko Brlek and Christophe Reutenauer, editors, *Developments in Language Theory - 20th International Conference, DLT 2016, Montréal, Canada, July 25-28, 2016, Proceedings*, volume 9840 of *Lecture Notes in Computer Science*, pages 355–367. Springer, 2016. doi:10.1007/978-3-662-53132-7_29.

3 Christian Choffrut and Juhani Karhumäki. *Combinatorics of Words*, pages 329–438. Springer Berlin Heidelberg, 1997. doi:10.1007/978-3-642-59136-5_6.

4 Javier Esparza and Michael Luttenberger. Solving fixed-point equations by derivation tree analysis. In Andrea Corradini, Bartek Klin, and Corina Cîrstea, editors, *Algebra and Coalgebra in Computer Science - 4th International Conference, CALCO 2011, Winchester, UK, August 30 - September 2, 2011. Proceedings*, volume 6859 of *Lecture Notes in Computer Science*, pages 19–35. Springer, 2011. doi:10.1007/978-3-642-22944-2_2.

5 N. J. Fine and H. S. Wilf. Uniqueness theorems for periodic functions. *Proceedings of the American Mathematical Society*, 16(1):109–114, 1965. URL: http://www.jstor.org/stable/2034009.

6 Toru Kasai, Gunho Lee, Hiroki Arimura, Setsuo Arikawa, and Kunsoo Park. Linear-time longest-common-prefix computation in suffix arrays and its applications. In *Combinatorial Pattern Matching, 12th Annual Symposium, CPM 2001 Jerusalem, Israel, July 1-4, 2001 Proceedings*, pages 181–192, 2001.

7 Martin Lange and Hans Leiß. To CNF or not to cnf? an efficient yet presentable version of the CYK algorithm. *Informatica Didactica*, 8, 2009. URL: http://www.informatica-didactica.de/cmsmadesimple/index.php?page=LangeLeiss2009.

8 Grégoire Laurence, Aurélien Lemay, Joachim Niehren, Slawek Staworko, and Marc Tommasi. Normalization of sequential top-down tree-to-word transducers. In Adrian-Horia Dediu, Shunsuke Inenaga, and Carlos Martín-Vide, editors, *Language and Automata Theory and Applications - 5th International Conference, LATA 2011, Tarragona, Spain, May 26-31, 2011. Proceedings*, volume 6638 of *Lecture Notes in Computer Science*, pages 354–365. Springer, 2011. doi:10.1007/978-3-642-21254-3_28.

9 Grégoire Laurence, Aurélien Lemay, Joachim Niehren, Slawek Staworko, and Marc Tommasi. Learning sequential tree-to-word transducers. In Adrian-Horia Dediu, Carlos Martín-Vide, José Luis Sierra-Rodríguez, and Bianca Truthe, editors, *Language and Automata Theory and Applications - 8th International Conference, LATA 2014, Madrid, Spain, March 10-14, 2014. Proceedings*, volume 8370 of *Lecture Notes in Computer Science*, pages 490–502. Springer, 2014. doi:10.1007/978-3-319-04921-2_40.

10 Markus Lohrey. Algorithmics on slp-compressed strings: A survey. *Groups Complexity Cryptology*, 4(2):241–299, 2012. doi:10.1515/gcc-2012-0016.

11 Michael Luttenberger, Raphaela Palenta, and Helmut Seidl. Computing the longest common prefix of a context-free language in polynomial time. *CoRR*, abs/1702.06698, 2017. arXiv:1702.06698.

12 Wojciech Plandowski. Testing equivalence of morphisms on context-free languages. In Jan van Leeuwen, editor, *Algorithms - ESA '94, Second Annual European Symposium, Utrecht, The Netherlands, September 26-28, 1994, Proceedings*, volume 855 of *Lecture Notes in Computer Science*, pages 460–470. Springer, 1994. doi:10.1007/BFb0049431.