TECHNISCHE UNIVERSITÄT MÜNCHEN
FAKULTÄT FÜR INFORMATIK
Lehrstuhl für Sprachen und Beschreibungsstrukturen

Linear Tree Transducers
From Equivalence to Balancedness

Raphaela Löbel

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:
Prof. Dr. Francisco Javier Esparza Estaun
Prüfer der Dissertation:
1. Prof. Dr. Helmut Seidl
2. Prof. Dr. Sebastian Maneth

Die Dissertation wurde am 06.07.2020 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 09.11.2020 angenommen.

Persevera!

Petrus Canisius

Abstract

This thesis studies two decision problems on linear tree transducers with serialized output – equivalence and balancedness. We show that equivalence of linear tree transducers with output in the free group is decidable in polynomial time. The decision procedure is based on an ordered form which guarantees that equivalent linear tree transducers process their input in the same order. This normal form is further analysed on two other types of linear tree transducers. First, if the output is interpreted as words in the free monoid, thus no inverses exist and the two words are only equivalent if they are literally the same. Second, the output is interpreted over the involutive monoid, thus the output alphabet consists of disjoint sets A and $\overline{\mathsf{A}} = \{\overline{a} \mid a \in \mathsf{A}\}$ where occurrences of the form $a\overline{a}$, $a \in \mathsf{A}$ reduce to the empty word.

For the second decision problem – balancedness – the output is again interpreted over the involutive monoid. A word is balanced if it reduces to the empty word, i.e., consists of properly nested opening (A) and closing letters ($\overline{\mathsf{A}}$). We consider 2-copy tree transducers (2-TWs) which call in their starting axiom two linear tree transducers on the same input. Thus, the processed transduction is not linear and the output language is not context-free. However, the output language of a linear tree transducer is context-free. We reduce balancedness of 2-TWs to the two questions whether a context-free grammar is well-formed and whether equivalence of linear tree transducers is decidable. A word is well-formed if it is a prefix of a balanced word or, equivalently, reduces to a word over opening letters only. We show that well-formedness of context-free grammars is decidable in polynomial time. Therefore, we provide a polynomial time algorithm to compute the longest common suffix of a context-free grammar *after reduction*. The algorithm performs a fixed-point iteration for that we (i) introduce a polynomial size representation for a language L to compute the longest common suffix of L, and (ii) show that the reduced longest common suffix of a context-free grammar G is the same as the reduced longest common suffix of the words with a derivation tree up to height $4N$ where N the number of nonterminals in G.

Zusammenfassung

Diese Arbeit untersucht zwei Entscheidungsprobleme über lineare Baumübersetzer – Äquivalenz und Balanciertheit. Wir zeigen, dass die Äquivalenz von linearen Baumübersetzern mit Ausgabe in der freien Gruppe in polynomieller Zeit entscheidbar ist. Der Entscheidungsalgorithmus basiert auf einer geordneten Normalform, die sicher stellt, dass äquivalente lineare Baumübersetzer ihre Eingabe in der gleichen Reihenfolge verarbeiten. Diese geordnete Normalform wird zusätzlich für zwei weitere Arten von linearen Baumübersetzern analysiert. Zum einen wenn die Ausgabe als Wörter im freien Monoid interpretiert wird und somit keine Inversen vorkommen und zwei Wörter ausschließlich äquivalent sind, wenn sie in jedem Buchstaben übereinstimmen. Zum anderen wenn die Ausgabe über einem Monoid mit Involution interpretier wird, d.h. das Ausgabealphabet besteht aus disjunkten Mengen A und $\overline{\mathsf{A}} = \{\overline{a} \mid a \in \mathsf{A}\}$ und alle Vorkommen der Form $a\overline{a}$, $a \in \mathsf{A}$ reduzieren sich zum leeren Wort.
Für das zweite Entscheidungsproblem – Balanciertheit – wird die Ausgabe wieder über einem Monoid mit Involution interpretiert. Ein Wort ist balanciert, wenn es sich zum leeren Wort reduzieren lässt, d.h. es besteht aus korrekt geklammerten öffnenden (A) und schließenden Buchstaben ($\overline{\mathsf{A}}$). Wir betrachten Baumübersetzer mit zwei Kopien (2-TWs), die in ihrer Startregel zwei lineare Baumübersetzer mit der gleichen Eingabe aufrufen. Somit ist die durchgeführte Übersetzung nicht linear und die Ausgabesprache ist nicht kontextfrei. Wir reduzieren Balanciertheit von 2-TWs zu den zwei Fragen, ob eine kontextfreie Grammatik wohlgeformt ist und ob Äquivalenz von linearen Baumübersetzern entscheidbar ist. Ein Wort ist wohlgeformt, wenn es ein Prefix eines balancierten Wortes ist oder, anders formuliert, sich zu einem Wort über ausschließlich öffnenden Buchstaben reduzieren lässt. Wir zeigen, dass Wohlgeformtheit von kontextfreien Grammatiken in polynomieller Zeit entscheidbar ist. Dafür verwenden wir einen polynomiellen Algorithmus zur Berechnung des längsten gemeinsamen Suffixes einer kontextfreien Grammatik *nach Reduktion*. Der Algorithmus führt eine Fixpunktiteration durch, für die wir (i) eine polynomiell große Repräsentation einer Sprache L zur Berechnung des längsten gemeinsamen Suffixes von L einführen und (ii) zeigen, dass der längste gemeinsame Suffix einer kontextfreien Grammatik G gleich dem längsten gemeinsamen Suffix aller Wörter, die einen Ableitungsbaum mit maximaler Höhe $4N$ haben, ist, wobei N die Anzahl der Nichtterminale in G ist.

List of Publications

This thesis includes the content of the following four publications:

[BP16] 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, Proceedings*, volume 9840 of *Lecture Notes in Computer Science*, pages 355–367. Springer, 2016

[LPS18] Michael Luttenberger, Raphaela Palenta, and Helmut Seidl. Computing the longest common prefix of a context-free language in polynomial time. In Rolf Niedermeier and Brigitte Vallée, editors, *35th Symposium on Theoretical Aspects of Computer Science, STACS 2018*, volume 96 of *LIPIcs*, pages 48:1–48:13. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018

[LLS20a] Raphaela Löbel, Michael Luttenberger, and Helmut Seidl. Equivalence of linear tree transducers with output in the free group. In Natasa Jonoska and Dmytro Savchuk, editors, *Developments in Language Theory - 24th International Conference, DLT 2020, Proceedings*, volume 12086 of *Lecture Notes in Computer Science*, pages 207–221. Springer, 2020

[LLS20b] Raphaela Löbel, Michael Luttenberger, and Helmut Seidl. On the balancedness of tree-to-word transducers. In Natasa Jonoska and Dmytro Savchuk, editors, *Developments in Language Theory - 24th International Conference, DLT 2020, Proceedings*, volume 12086 of *Lecture Notes in Computer Science*, pages 222–236. Springer, 2020

The publications [BP16, LPS18] are published under my former name Raphaela Palenta. Chapters 2 and 4 are based on [LLS20b] where Sections 2.3 and 2.4 subsume and improve the result presented in [LPS18]. Chapter 3 is based on [LLS20a] whereas Section 3.3 recaps the result shown in [BP16].

Contents

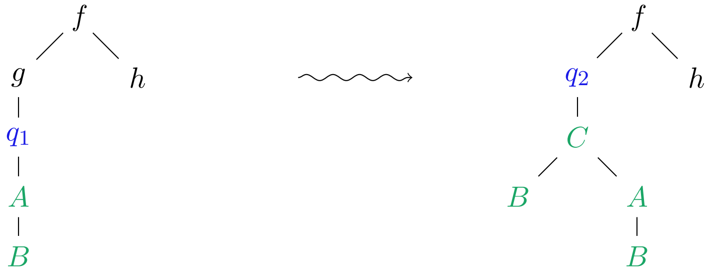
1 Introduction

Tree transformations are widespread in different applications, e.g. in functional programming [Wad90, Voi05], document processing [MN99, MBPS05, HMNI14] or automatic translation of natural languages [MGHK09, Mal17, BSQM13]. Tree transducers can be seen as a form of recursive programs that take as input ranked trees and produce as output trees or as serialization thereof words. The former model is called tree-to-tree transducer and the latter one tree-to-word transducer. We will only consider *deterministic* tree transducers as equivalence of non-deterministic transducers is undecidable [Gri68]. Early results on tree-to-tree transducers date back to the 1970s and 1980s where equivalence of bottom-up [Zac79] and top-down [Ési80] tree-to-tree transducers was shown to be decidable. A tree-to-tree transducer takes as input a tree and produces as output again a tree. It can either process over the input tree from bottom to top, i.e., from the leaves to the root node or from top to bottom, i.e., starting from the root node down to the leaves.

Example 1. *A rule of a bottom-up tree-to-tree transducer has the general form*

$$a(q_1(x_1), \dots, q_m(x_m)) \to q(t_X)$$

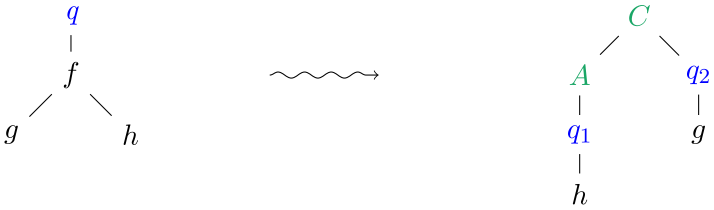
where a is an input symbol of rank m, $q, q_1, \dots, q_m$ are the states of the transducer, the variables $x_1, \dots, x_m$ represent subtrees that are already processed and t_X is the output tree that is produced and can contain the variables $x_1, \dots, x_m$ as placeholders for the already processed subtrees. For example, the rule $g(q_1(x_1)) \to q_2(C(B, x_1))$ applied on the tree $f(g, h)$ on the left yields the tree shown on the right. States are in blue color, their parameters in green color and the parameters of a state are depicted as its subtrees.



A rule of a top-down tree-to-tree transducer has the general form

$$q(f(x_1, \dots, x_m)) \to t_{Q(X)}$$

where q is a state, f is an input symbol of rank m, such that the variables $x_1, \dots, x_m$ represent the subtrees of the input tree and $t_{Q(X)}$ is an output tree that can contain recursive calls of the form $q_j(x_i)$ with q_j a state and variable x_i representing a subtree of the input tree. For example, the rule $q(f(x_1, x_2)) \to C(A(q_1(x_2)), q_2(x_1))$ applied on the tree on the left side yields the tree on the right side. States are in blue color, their parameter (the input tree) is depicted as subtree and the output is in green color.



In contrast to deterministic tree-to-tree transducers, equivalence of deterministic *tree-to-word* transducers was shown to be decidable in 2015 [SMK15] and was thus an open problem for at least 35 years [Eng80]. A tree-to-word transducer takes as input a tree and processes from top to bottom over this input tree and produces thereby a word.

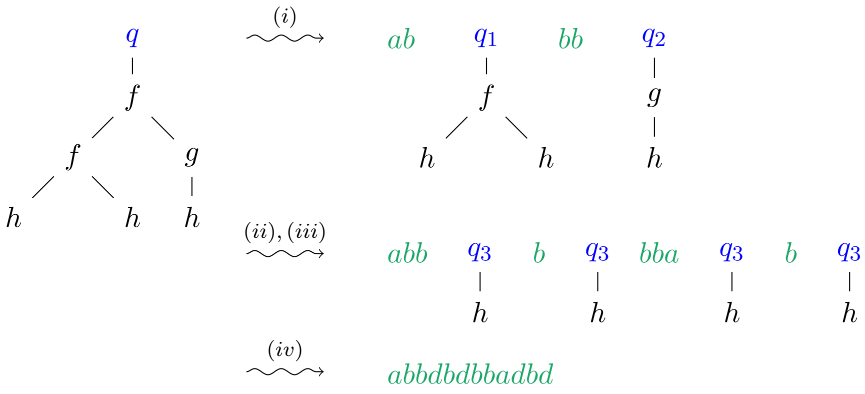
Example 2. *A rule of a tree-to-word transducer is in general of the form*

$$q(f(x_1, \dots, x_m)) \to u_0 q_1(x_{i_1}) u_1 \dots u_{l-1} q_l(x_{i_l}) u_l$$

where $q, q_1, \dots, q_l$ are states, f is an input symbol of rank m such that the variables $x_1, \dots, x_m$ represent the subtrees of the input tree, $u_0, \dots, u_l$ are output words and the variables x_{i_j} represent one of the m subtrees of the input. For example, the four rules

(i)	$q(f(x_1, x_2))$	$\to ab\, q_1(x_1)\, bb\, q_2(x_2)$
(ii)	$q_1(f(x_1, x_2))$	$\to b\, q_3(x_2)\, b\, q_3(x_1)$
(iii)	$q_2(g(x_1))$	$\to a\, q_3(x_1)\, b\, q_3(x_1)$
(iv)	$q_3(h)$	$\to d$

applied on the tree $f(f(h, h), g(h))$ on the left side lead to the following step-wise transductions where states are in blue color with their parameter (the input tree) depicted as subtree and the output words are in green color.



Different steps towards solving the equivalence of tree-to-word transducers have been made. A large class of tree-to-word transductions for which equivalence was shown to be decidable are those definable in monadic second-order (MSO) logic [EM06] that can be described by macro tree-to-word transducers with linear size increase [EM03]. Macro tree transducers use additional parameters to accumulate output values, i.e., the left-hand side of a rule has the general form $q(f(x_1, \ldots, x_m), y_1, \ldots, y_k)$ where the additional parameters y_i can be output as well. Without these additional parameters different restricted classes of tree-to-word transducers have been considered, e.g. those that are non-copying, i.e., in every rule of the transducer any subtree x_i of the input occurs at most once on the right-hand side of the rule. We call these tree-to-word transducers *linear* and *sequential* if additionally the order in which the subtrees are processed is the same as the order in the input tree. Thus, if $q_1(x_{i_1}), \ldots, q_l(x_{i_l})$ are the recursive calls on the right-hand side of a rule then $i_1 < i_2 < \ldots < i_l$ and l is at most the rank of the input symbol of the rule. For example, in Example 2 rule (i) fulfills the sequential property, rule (ii) is not sequential but linear and rule (iii) is not linear as the subtree x_1 is copied. Equivalence of sequential tree-to-word transducers was shown to be decidable in *polynomial time* [SLLN09]. Additionally, in [LLN+11] a normal form for sequential tree-to-word transducers was provided that allows for learning [LLN+14]. The key concept of this normal form is that the output is produced as *early* as possible and was already used for string transducers [Moh00, Cho03], i.e., transducers that take as input words and produce words as output, and tree-to-tree transducers [EMS09, FSM10]. Producing the output as early as possible means that at any point of the production the further output that will be produced relies on the further input. Thus, in case of tree-to-word transducer the words of the output language of a state has no common prefix or suffix. If there would be a common prefix or suffix this constant output could already be produced without knowing the further input.

Example 3. *The following rules of a tree-to-word transducer*

$$q(f(x_1)) \to ab\,q(x_1) \qquad q(g) \to a$$

process input trees of the form $f^n(g)$ to output words $(ab)^n a$. The output is not produced as early as possible since the output always starts with an a independently of the input. With other words the longest common prefix of the output language is a. In contrast, state q' with

$$q'(f(x_1)) \to ba\,q'(x_1) \qquad q'(g) \to \varepsilon$$

is earliest and $aq'(t)$ produces the same output as $q(t)$ for input trees of the form $f^n(g)$.

A state that produces the output as early as possible is called earliest and a transducer M is earliest if every state of M is earliest. Accordingly in [Boi16] an earliest normal form for *linear* tree-to-word transducers was provided, i.e., for non-copying but not necessarily order-preserving transducers. Equivalence of linear tree-to-word transducers in normal form can be decided in polynomial time [Boi16]. However, building an equivalent tree-to-word transducer that is in normal form takes exponential time. We lifted this normal form such that only the order of the processed subtrees is normalized and showed that with this *partial normal form* the equivalence of linear tree-to-word transducers is decidable in polynomial time [BP16]. The normal form is called partial as not all but only so called *ultimately periodic* states have to be earliest. A state is ultimately periodic if the output language is either of the form vp^* or p^*v; thus the language is periodic but might have additionally a constant prefix or suffix. In case of linear tree-to-word transductions the output languages can be described by a context-free grammar. Therefore, the longest common prefix (suffix) of a context-free grammar has to be computed if we test whether a state of a linear tree-to-word transducer is earliest. This raises the question whether there is an efficient algorithm to compute the longest common prefix (suffix) of a context-free language. Surprisingly, there were no results on that even so context-free grammars have already been introduced in the 1950s by Chomsky [Cho56] and intensively studied in the following decades [Flo62, HU69, HU79]. A context-free grammar encoding a single derivation tree and thus representing exactly one word is called straight-line program (SLP for short). SLPs and especially the compressed pattern matching on SLPs have been extensively studied [KRS95, PR99, Ryt03, Loh06, LL06] due to their use as compression of large strings that is important e.g. in large genome databases or XML processing. The longest common prefix (suffix) of two SLPs can be computed in polynomial time [Loh12]. We showed that the longest common prefix (suffix) of a context-free language that can represent *infinitely* many words, can be computed in polynomial time [LPS18].

We come back to the observation of the beginning that the decidability of equivalence of tree-to-tree transducers seemed to be easier to solve than in the case of tree-to-word transducers. One reason for that might be that tree-to-word transducers can produce their output in an *unstructured* way. Even if the output is structured, e.g. is a valid XML document with properly opening and closing tags, these corresponding opening and closing tags do not have to be produced together but can be output in many different ways. On the other hand the output does not need to be structured what leads to the question whether the output of a tree-to-word transducer is structured or as we will say *balanced*. Formally, we consider a language over opening letters A and corresponding closing letters $\overline{\mathsf{A}} = \{\overline{a} \mid a \in \mathsf{A}\}$ and want to decide whether every word in this language is balanced, i.e., for every opening letter there is a properly matching closing letter. For example, $ab\overline{b}\,\overline{a}$ is balanced while $ba\overline{b}\,\overline{a}$, $a\overline{a}\,\overline{b}$ and a are not balanced, $a, b \in \mathsf{A}, a \neq b$. In case of context-free grammars decidability of balancedness for a single pair of opening and closing letters was shown to be decidable by Knuth already in 1967 [Knu67]. In the beginning of the 21st century a *polynomial* time algorithm for deciding balancedness was presented [MT06]. Additionally the case of multiple pairs of opening and corresponding closing letters was shown to be decidable [BB02] and again a polynomial time algorithm was presented [TM07]. More recently it was proven that balancedness of MSO definable tree-to-word transducer for a single pair of opening and closing letters is decidable [MS18]. Whether this result can be extended to the case of multiple pairs of opening and closing letters is still an open problem. We present a polynomial time algorithm to decide balancedness of the output of 2-copy tree-to-word transducers (2-TWs for short) for multiple pairs of different kind of brackets. These transducers call in their start axiom two linear tree-to-word transducers on the given input tree and process therefore a *non-linear* transduction.

Example 4. *We consider* 2-TW M *with axiom* $q(x_1)\,q'(x_1)$ *and the following rules*

$$\begin{array}{rlrl} q(f(x_1)) & \to a\,q(x_1)\,b \qquad & q'(f(x_1)) & \to \overline{b}\;q'(x_1)\,\overline{a} \\ q(h) & \to \varepsilon & q'(h) & \to \varepsilon \end{array}$$

M *transforms input trees of the form* $f^n(h)$ *to output words* $a^n b^n \overline{b}^{\,n} \overline{a}^{\,n}$*. Note that the output language is* not *context-free.*

We reduce balancedness of 2-TWs to two other decision problems: (a) the well-formedness of context-free grammars and (b) the equivalence of linear tree-to-word transducers over the free group. A context-free grammar is *well-formed* if every word produced by the grammar is a prefix of a balanced word. For example, $ab\overline{b}$ and $ab\overline{b}\,a$ are well-formed, while $a\overline{b}$, $a\overline{a}\,\overline{b}$ are not well-formed. If L_1, L_2 are the (context-free) output languages of two linear tree-to-word transducer called in the axiom of a balanced 2-TW, then L_1 and $\overline{L_2}$ have to be well-formed. The inverted language

$\overline{L_2}$ is obtained from L_2 by reversing every word and replacing an opening by the corresponding closing letter and vice versa, e.g. $\overline{\overline{a}\,\overline{b}\,a\overline{a}} = a\overline{a}\,ba$. In the same lines a context-free grammar or the transduction of a linear tree-to-word transducer T can be inverted, called $\overline{T}$. Then we observe that the output of a transduction $T_1(t)T_2(t)$ with T_1 and $\overline{T_2}$ producing well-formed output only, is balanced if $T_1(t)$ and $\overline{T_2}\,(t)$ are the same after removing matching opening and closing brackets. We show that with the prerequisite that T_1 and $\overline{T_2}$ are well-formed we can test T_1 and $\overline{T_2}$ for equivalence over the free group to obtain a decision procedure for balancedness of 2-TWs. To arrive at a polynomial time decision procedure equivalence of linear tree-to-word transducers has to be decidable in polynomial time. We show that the polynomial time algorithm for equivalence of linear tree-to-word transducers where word equivalence is considered [BP16] can be extended to the free group. In fact, due to the inverses provided in the free group the algorithm becomes easier and cleaner as well as the proofs for the correctness. The observation that results can be lifted from words to the free group and the argumentation in fact becomes more direct and therefore even shorter was already made for other results. E.g. Plandowski extended his result of the existence of polynomial size test sets for context-free languages from words [KPR92] to the free group [Pla94].

Well-formedness and Longest Common Suffix. We consider a well-formed word $w \in (\mathsf{A} \cup \overline{\mathsf{A}}\,)^*$, i.e., there is a word $v \in (\mathsf{A} \cup \overline{\mathsf{A}}\,)^*$ such that wv is balanced. We denote with $\rho(w)$ the *reduced* word we obtain from w by removing all properly matching opening and closing brackets. Then, w is well-formed if $\rho(w) \in \mathsf{A}^*$. We show that deciding well-formedness of a context-free grammar G relies on the computation of the *reduced* longest common suffix of G. Similar to the idea in [TM07] to decide balancedness of a context-free grammar we observe that any sentential form $\alpha X \beta$, $\alpha, \beta \in (\mathsf{A} \cup \overline{\mathsf{A}}\,)^*$ derived from a well-formed context-free grammar leads to the following constraints for the words produced by nonterminal X. If $\rho(\alpha) = w$, $\rho(\beta) = \overline{u}\,v$, $u, v, w \in \mathsf{A}^*$ then for all $\gamma \in (\mathsf{A} \cup \overline{\mathsf{A}}\,)^*$ derivable from X with $\rho(\gamma) = \overline{u_\gamma}\,v_\gamma$, (a) $w = w' u_\gamma$ and (b) $v_\gamma = v'_\gamma u$ has to hold. These conditions can be checked if we compute the longest common suffix of the reduced language of every nonterminal in G. We presented a polynomial time algorithm to compute the longest common prefix (suffix) of a context-free grammar over a general alphabet Σ in [LPS18]. There no reduction had to be taken into account such that we can not directly apply this result here. In fact, it needs completely new comprehensive proofs to obtain a polynomial time algorithm for the computation of the *reduced* longest common suffix. The underlying ideas of the proofs, however, stayed the same. As the longest common suffix is a suffix of the shortest word of the language the size of the shortest word is an upper bound. But the size of the shortest word of a context-free grammar can be exponential in the size of the grammar.

Example 5. *Let $n \in \mathbb{N}$ be fixed. The context-free grammar G has nonterminals $A_1, \dots A_n$, axiom A_n and the rules*

$$\begin{aligned} &A_n \to A_{n-1}A_{n-1} \\ &\dots \\ &A_1 \to A_0A_0 \\ &A_0 \to a \end{aligned}$$

Every nonterminal A_i produces the word a^{2^i}, e.g. for $n = 3$ the context-free grammar G has the derivation $A_3 \to^* A_2A_2 \to^* A_1A_1\ A_1A_1 \to^* A_0A_0\ A_0A_0\ A_0A_0\ A_0A_0 \to^* a^{2^3}$.

Every nonterminal in the above example is an SLP and represents a word that is exponential in the size of the SLP. The challenge in computing the longest common suffix of two SLPs *efficiently* is to find algorithms that directly work on the grammar – the compressed word – and do not decompress the word as this leads in general to a exponential runtime. An SLP for a shortest word w of a context-free grammar can be computed in polynomial time. However, the question remains how we can determine a second word v of a given CFG G that together with a fixed shortest word defines the longest common suffix (lcs) of the reduced language L of G, i.e., $\mathsf{lcs}(\rho(w), \rho(v)) = \mathsf{lcs}(\rho(L))$. If we consider the longest common suffix of a language after reduction we say reduced longest common suffix. We show that it is enough to consider the words of a CFG G with a derivation tree of height at most $4N$ to compute the reduced longest common suffix of G where N is the number of nonterminals in G. The proof is a combinatorical argumentation combined with a pumping argument. We investigate how often a pumping tree can be repeated/pumped such that the resulting words could change (shorten) the reduced longest common suffix.

Example 6. *Consider the context-free language $L = aab(a)^n(ab)^n$ with $n \geq 0$. Then L contains for $n = 0, 1, 2, 3, \dots$ the words*

$$\begin{aligned} aab &\qquad n = 0 \\ aabaab &\qquad n = 1 \\ aabaaabab &\qquad n = 2 \\ aabaaaababab &\qquad n = 3 \\ \dots & \end{aligned}$$

The word $aab(a)^2(ab)^2$ together with the shortest word aab define the longest common suffix of language L, i.e., $\mathsf{lcs}(L) = \mathsf{lcs}(aab, aabaaabab) = ab$. Enumerating more words $aab(a)^i(ab)^i$ with $i \geq 3$ containing the pumping tree more than two times do not change the longest common suffix of L.

In the above Example 6 the language L contains just one pumping tree while in general a context-free grammar can contain many different pumping trees. We therefore additionally have to reason about the impact of the combination of different pumping trees on the reduced longest common suffix of a language.

Equivalence. A normal form for linear tree-to-word transducers was shown in [Boi16]. The construction of this normal form is exponential in the size of the original transducer. We showed that this normal form can be adapted to obtain a polynomial time equivalence test for linear tree-to-word transducers [BP16]. We further extend this result by considering linear tree transducers with outputs in the free group. In fact, we show that with the same underlying ideas as in [BP16] the equivalence of tree transducers with output in the free group is decidable in polynomial time. If two linear tree transducers process their input in the same order then a context-free grammar simulating the parallel runs of the two transducers can be constructed, cf. [SLLN09], and equivalence can be tested in polynomial time by the morphism equivalence result from Plandowski [Pla94]. We therefore define an *order* for linear tree transducers such that transducers that are equivalent and ordered process their input in the same order. The following example illustrates the key observation for the characterization of the output languages in equivalent transductions that process their input in different orders.

Example 7. *We consider the following two linear tree-to-word transducers T_1, T_2*

$$T_1 : \; q_0(g(x_1,x_2)) \to a\, q_1(x_1)\, q_2(x_2) b \qquad T_2 : \; q_0'(g(x_1,x_2)) \to q_2'(x_2)\, a\, q_1'(x_1) b$$

$$\begin{array}{rcl@{\qquad}rcl} q_1(f(x)) & \to & ba\, q_1(x) & q_1'(f(x)) & \to & ba\, q_1'(x) \\ q_1(h) & \to & b & q_1'(h) & \to & \varepsilon \\ q_2(f(x)) & \to & q_2(x)\, ba & q_2'(f(x)) & \to & q_2'(x)\, ab \\ q_2(h) & \to & a & q_2'(h) & \to & ab \end{array}$$

both performing equivalent transductions from input trees of the form $g(f^k(h), f^j(h))$ to the output language $abab(ab)^{k+j}$. The output language of T_1 is $a\, L_1\, L_2\, b$ with $L_1 = (ba)^k\, b$ and $L_2 = a\,(ba)^j$ the output languages of q_1, q_2, respectively. The output language of T_2 is $L_2'\, a\, L_1' b$ with $L_2' = ab\,(ab)^j$ and $L_1' = (ba)^k$ the output languages of q_2', q_1', respectively. The order of the recursive calls on the subtrees x_1, x_2 in the rules of the states q_0 , q_0' differs but the transduction is the same as the output languages of q_1 , q_1', q_2 , q_2' are all periodic and the periods are conjugates of each other.

Based on the characterization of the output languages in an unordered part of the transduction we show how to order the recursive calls on the subtrees. In fact, this rewriting to obtain an ordered linear tree transducer is easier and cleaner in case inverses provided by the free group can be used compared to the procedure needed in the word case [BP16]. We recap the ordering of linear tree-to-word transducers from [BP16] based on the new results to explicitly show the differences in the rewriting if inverses are used or not.

Balancedness. For context-free languages the question of balancedness is the question whether the language is contained in the Dyck language given by the grammar

$$S \to \varepsilon \mid [S]S$$

While the general question whether a context-free grammar is a subset of another context-free grammar is undecidable, the question whether a context-free language is a Dyck language is decidable [Knu67, BB02]. Balancedness was naturally extended to the case of multiple pairs of different brackets which we consider here. We show that balancedness of the output language of a 2-copy tree-to-word transducer (2-TW) is decidable in polynomial time. Assume that $T_1(x_0)T_2(x_0)$ is the axiom of a 2-TW M. Thus, T_1, T_2 are the two linear tree-to-word transducers called on the input tree in the axiom and the output is evaluated in the free involutive monoid. In Section 4.2 we reduce balancedness of M to the following two decision problems, (a) well-formedness of the output languages of $T_1, \overline{T_2}$ and (b) equivalence of T_1 and $\overline{T_2}$ over the free group. The transducer $\overline{T_2}$ produces for an input tree t the inverted output of T_2 on t – a word is inverted by reversing the word and replacing opening by corresponding closing letters and vice versa. It is notable that in the free group other rewriting rules apply than in the rewriting system we consider for balancedness. For balancedness the order of corresponding opening and closing letters are important, e.g. $a\overline{a}$ reduces to ε while $\overline{a}\,a$ do not reduce any further and is not balanced; in the free group aa^{-1} as well as $a^{-1}a$ are equivalent to the empty word. However, as we test the output languages of T_1 and $\overline{T_2}$ for well-formedness no patterns of the form $\overline{a}\,a$ can occur in the output of T_1 and $\overline{T_2}$ and inverted letters can therefore be interpreted as inverses in the free group. Additionally, we present in Section 4.3 an alternative approach to decide equivalence of linear tree-to-word transducers over the involutive monoid. The key observation thereby is that all inverted letters occurring in the rules of (ultimately) periodic states can be removed by a rewriting similar to the ordering used for linear tree-to-word transducers.

2 Well-formedness of Context-free Grammars

We consider context-free grammars over an involutive monoid; therefore let A be an alphabet opening letters and $\overline{\mathsf{A}} = \{\overline{a} \mid a \in \mathsf{A}\}$ be the alphabet of corresponding closing letters. Then a word $w \in (\mathsf{A} \cup \overline{\mathsf{A}})^*$ is called *reduced* if it contains no occurrences aa^-, $a \in \mathsf{A}$ of matching opening and closing letters. Every word $w \in (\mathsf{A} \cup \overline{\mathsf{A}})^*$ reduces to a unique reduced word independently of the order in which matching opening and closing letters are canceled. The reduced word of w is denoted by $\rho(w)$, e.g. $\rho(acb\overline{b}\,\overline{c}) = a$. We call words that reduce to the empty word ε *balanced* and words that are prefixes of a balanced word *well-formed*, i.e., w is well-formed if $\rho(w) \in \mathsf{A}^*$. For example $a\overline{a}\,ba\overline{a}\,\overline{b}$ is balanced and $a\overline{a}\,b$ is well-formed as $\rho(a\overline{a}\,b) = a \in \mathsf{A}^*$. We extend the notations accordingly to languages, i.e., a language is balanced if all contained words are balanced and a language is well-formed if all contained words are well-formed. While the question whether the language produced by a context-free grammar is balanced was already introduced by Knuth in 1967 [Knu67] and investigated further [MT06, BB02, TM07], the tightly connected question of well-formedness was not discussed. Moreover, we could not find any straightforward modifications to the existing decision algorithms for balancedness to obtain a decision algorithm for well-formedness. In Section 2.3 we show that well-formedness can be reduced to the computation of the *reduced* longest common suffix of a context-free grammar. I.e., we consider the longest common suffix *after reduction* of the produced words. Therefore the polynomial algorithm to compute the longest common suffix of a context-free grammar without considering any reduction [LPS18] cannot be applied. The underlying ideas of the proofs are very similar, however, completely new proofs for the case with reduction were needed. We present a fixed point algorithm to compute the reduced longest common suffix of a context-free grammar that terminates in polynomial time. Two important components are needed to obtain this algorithm. The first component is a polynomial size representation of arbitrary languages for the computation of the longest common suffix. We show in Section 2.2 that for the computation of the longest common suffix we can represent any language by a language consisting of at most three words. More precisely, we represent every language by the so called lcs-summary consisting of the lcs of the language and the lcs-extension. The lcs-extension is the maximal word by that the lcs of the language can be extended if another language is concatenated from the left and might be unbounded in the length. For example, the language $L_1 = (ab)^*a$ has the longest common suffix a and if a language $L_1' \subseteq (ab)^*$ is concatenated from the left the lcs can be extended by any factor $(ab)^i$; the language $L_2 = \{a, aba, baba\}$ can at most be extended by bab if bab is a suffix of the longest common suffix of the language concatenated from the left, e.g. $\mathsf{lcs}(\{abab\}L_2) = \mathsf{lcs}(\{ababa, abababa, ababbaba\}) = baba$. The second component is a bound on the height of the derivation trees of a context-free grammar that need to be considered for the computation of the longest common suffix. We show in Section 2.3 that for a given context-free grammar G the reduced longest common suffix of the words with a derivation tree of height at most $4N$ is the same as for all words produced by G. The proof is based on the analysis of simple linear well-formed grammars of the form

$$S \to \alpha\, X\, \beta \qquad X \to \sigma_1\, X\, \tau_1 \mid \sigma_2\, X\, \tau_2 \mid \gamma$$

with $\sigma_i, \tau_i, \gamma \in (\Sigma \cup \overline{\Sigma}\,)^*$. In Section 2.4 we show that the reduced longest common suffix of the language produced by such a grammar is the same as the reduced longest common suffix of $\alpha\gamma\beta$ and either $\alpha\sigma_i\gamma\tau_i\beta$ or $\alpha\sigma_i\sigma_j\gamma\tau_j\tau_i\beta$ with $i \in \{1,2\}$ and $j \in \{1,2\}$ independently chosen from i.

Outline We state basic notation and definitions in Section 2.1. In Section 2.2 we present a polynomial size representation of languages called lcs-summary that we use in the computation of the longest common suffix. The lcs-summary of a language L consists of the longest common suffix of L and the maximal extension of the longest common suffix. The maximal extension of the longest common suffix v of a language L is the maximal word w for that it exists a language L' such that vw is the longest common suffix of $L'L$. We show that the lcs-summary of the concatenation or union of two languages L_1, L_2 can be computed if only the lcs-summary of L_1, L_2 is known. The main result of Chapter 2, i.e., how to decide well-formedness of a context-free grammar is presented in Section 2.3. The comprehensive combinatorical results that are needed to compute the reduced longest common suffix of a context-free grammar can be found in Section 2.4.

2.1 Preliminaries

As usual, $\mathbb{N}$ ($\mathbb{N}_0$) denotes the natural numbers (including 0). The power set of a set S is denoted by 2^S. Σ denotes some generic (nonempty) alphabet, $\Sigma^* = \bigcup_{k \in \mathbb{N}_0} \Sigma^k$ and $\Sigma^\omega\{(w_i)_{i\in\mathbb{N}_0} \mid w_i \in \Sigma\}$ denote the set of all finite words and the set of all infinite words, respectively. Then $\Sigma^\infty = \Sigma^* \cup \Sigma^\omega$ is the set of all countable words. We denote the empty word by ε. Concatenation of words and languages is defined as always. For a finite word $w = w_0 \dots w_l$, its reverse w^R is defined by $w^R = w_l \dots w_1 w_0$ with

$\varepsilon^R = \varepsilon$. A is used to denote an alphabet of *opening brackets* with $\overline{\mathsf{A}} = \{\overline{a} \mid a \in \mathsf{A}\}$ the derived alphabet of *closing brackets*, and $\mathsf{B} := \mathsf{A} \cup \overline{\mathsf{A}}$ the resulting alphabet of *opening and closing brackets.*

Longest common prefix and suffix We first define the *longest common prefix* of a language, and then reduce the definition of the *longest common suffix* to it by means of the reverse. We write $\overset{p}{\sqsubseteq}$ to denote the prefix relation on Σ^∞, i.e. we have $u \overset{p}{\sqsubseteq} w$ if either

- $u, w \in \Sigma^*$ and there exists $v \in \Sigma^*$ s.t. $w = uv$, or
- $u \in \Sigma^*$ and $w \in \Sigma^\omega$ and there exists $v \in \Sigma^\omega$ s.t. $w = uv$, or
- $u, w \in \Sigma^\omega$ and $u = w$.

We extend Σ^∞ by a greatest element $\top \notin \Sigma^\infty$ w.r.t. $\overset{p}{\sqsubseteq}$ s.t. $u \overset{p}{\sqsubseteq} \top$ for all $u \in \Sigma^\infty_\top := \Sigma^\infty \cup \{\top\}$. Then every set $L \subseteq \Sigma^\infty_\top$ has an infimum w.r.t. $\overset{p}{\sqsubseteq}$ which is called the *longest common prefix* of L, abbreviated by $\mathsf{lcp}(L)$. Alternatively, we can use the following equivalent set-theoretic definition of the lcp from which many of its properties become obvious. We embed $\Sigma^* \cup \Sigma^\omega$ into 2^{Σ^ω} by treating words as sets of infinite paths through the infinite $|\Sigma|$-ary tree. A finite word $u \in \Sigma^*$ is identified with the set of all infinite paths $u\Sigma^\omega := \{uw \in \Sigma^\omega\}$ that start with u resp. the subtree rooted at u of Σ^ω. An infinite word $w \in \Sigma^\omega$ is identified with $\{w\}$. Thus, for $u \in \Sigma^\infty$, we set $h(u) := u\Sigma^\omega$, if $u \in \Sigma^*$, and $h(u) := \{u\}$ otherwise. We then have $h(u) = \{w \in \Sigma^\omega \mid u \overset{p}{\sqsubseteq} w\}$, i.e. we can alternatively define $\overset{p}{\sqsubseteq}$ by means of $u \overset{p}{\sqsubseteq} v :\Leftrightarrow h(u) \supseteq h(v)$. Thus, $\top$ becomes the greatest element w.r.t. $\overset{p}{\sqsubseteq}$ by setting $h(\top) := \emptyset$ and we can identify ε^ω with $\emptyset$. Therefore we define $\varepsilon^\omega := \top$. Extend h to languages $L \subseteq \Sigma^\infty$ by means of $\hat{h}(L) := \bigcup_{w \in L} h(w)$. The $\mathsf{lcp}(L)$ is then the unique word in $\Sigma^\infty_\top$ satisfying

$$h(\mathsf{lcp}(L)) = \hat{h}(\{\mathsf{lcp}(L)\}) = \bigcap \{z\Sigma^\omega \mid \hat{h}(L) \subseteq z\Sigma^\omega\}$$

and thus is the infimum w.r.t. $\overset{p}{\sqsubseteq}$. We give an example in Figure 2.1. We define $\varepsilon^\omega := \top$, $\top^R := \top$, and $\top w := \top =: w\top$ for all $w \in \Sigma^\infty_\top$. Note, that for all $w \in \Sigma^\infty_\top$, $\mathsf{lcp}(\varepsilon^\omega, w) = w$ as $h(\varepsilon^\omega) = h(\top) = \emptyset \subseteq h(w)$. Additionally we have for all $w \in \Sigma^\infty_\top$, $\mathsf{lcp}(\varepsilon, w) = \varepsilon$ as $h(w) \subseteq \Sigma^\omega = h(\varepsilon)$.

In Section 2.3 we will need to study the *longest common suffix (lcs)* of a language L. For $L \subseteq \Sigma^*$, we can simply set $\mathsf{lcs}(L) := \mathsf{lcp}(L^R)^R$, but also for certain infinite words the lcs can be defined. Recall that for $u, w \in \Sigma^*$ and $w \neq \varepsilon$ the ω-regular expression uw^ω denotes the unique infinite word $uwww\ldots$ in $\bigcap_{k \in \mathbb{N}_0} uw^k\Sigma^\omega$; such a word is also called *ultimately periodic.* For the lcs we will use the expression $w^{\overleftarrow{\omega}}u$ to denote the "reverse" of $(u^R)(w^R)^\omega$, i.e. the infinite word $\ldots wwwu$ that ends

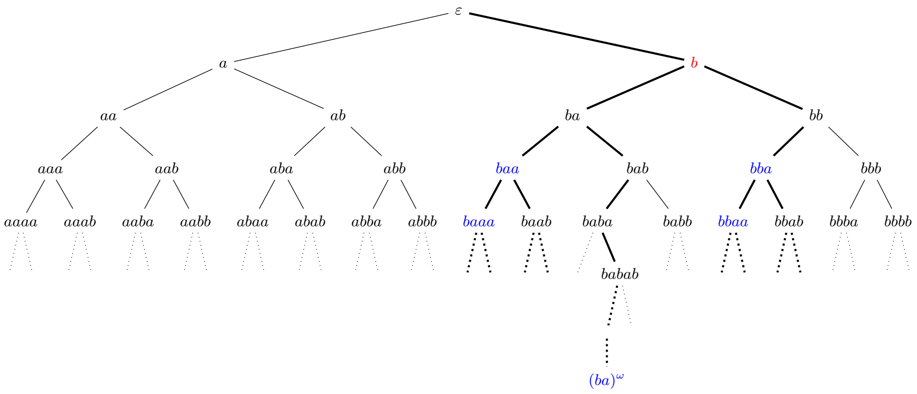


Figure 2.1: Given language $L = \{ba(ba)^\omega, baa, bba, baaa, bbaa\}$. The paths corresponding to the words contained in L are marked bold. The longest common prefix of L is then the branching point of these paths, i.e., b.

on the suffix u with infinitely many copies of w left of u; these words are used to abbreviate the fact that we can generate a word $w^k u$ for unbounded $k \in \mathbb{N}_0$.

Definition 1 (Ultimately left-periodic words, longest common suffix)**.**

- *Let Σ^{up} denote the set of all expressions of the form $w^{\mho}u$ with $u \in \Sigma^*$ and $w \in \Sigma^+$. Σ^{up} is called the set of* ultimately left-periodic words*. Define the reverse of an expression $w^{\mho}u \in \Sigma^{\mathsf{up}}$ by means of $(w^{\mho}u)^R := u^R(w^R)^\omega$. Accordingly, set $(uw^\omega)^R := (w^R)^{\mho}u^R$ for $u \in \Sigma^*$, $w \in \Sigma^+$.*

- *The* suffix order *on $\Sigma^* \cup \Sigma^{\mathsf{up}} \cup \{\top\}$ is defined by $u \overset{s}{\sqsubseteq} v :\Leftrightarrow u^R \overset{p}{\sqsubseteq} v^R$.*

- *The* longest common suffix (lcs) *of a language $L \subseteq \Sigma^* \cup \Sigma^{\mathsf{up}}$ is $\mathsf{lcs}(L) := \mathsf{lcp}(L^R)^R$.*

Note that a term like $\mathsf{lcs}(x^{\mho}, w)$ is always supposed to be read as $\mathsf{lcs}(\ldots xxxxx, w) = \mathsf{lcp}((x^R)^\omega, w)^R$ and to evaluate to a word in $\{\top\} \cup \Sigma^* \cup \Sigma^{\mathsf{up}}$. For instance, we have

$$\begin{aligned} &\mathsf{lcs}((bba)^{\mho}, (ba)^{\mho}a) = \mathsf{lcp}((abb)^\omega, a(ab)^\omega)^R = a, && \text{and} \\ &\mathsf{lcs}((ab)^{\mho}, (ba)^{\mho}b) = \mathsf{lcp}((ba)^\omega, b(ab)^\omega)^R = (ab)^{\mho}. \end{aligned}$$

Additionally, by definition $\varepsilon^{\mho} = \top$ s.t. $\mathsf{lcs}(x^{\mho}, \varepsilon^{\mho}) = \mathsf{lcs}(x^{\mho}, \top) = \mathsf{lcs}(x^{\mho}) = x^{\mho}$ while $\mathsf{lcs}(\varepsilon, x) = \varepsilon$ for all $x \in \{\top\} \cup \Sigma^* \cup \Sigma^{\mathsf{up}}$.

As usual, we write $u \overset{s}{\sqsubset} v$ if $u \overset{s}{\sqsubseteq} v$, but $u \neq v$ and $u \overset{p}{\sqsubset} v$ if $u \overset{p}{\sqsubseteq} v$, but $u \neq v$. As the lcp is the infimum w.r.t. $\overset{p}{\sqsubseteq}$, we also have for $x, y, z \in \{\top\} \cup \Sigma^* \cup \Sigma^{\mathsf{up}}$ and $L, L' \subseteq \{\top\} \cup \Sigma^* \cup \Sigma^{\mathsf{up}}$ that

- $\mathsf{lcs}(x, y) = \mathsf{lcs}(y, x)$,
- $\mathsf{lcs}(x, \mathsf{lcs}(y, z)) = \mathsf{lcs}(x, y, z)$,
- $\mathsf{lcs}(L) \overset{s}{\sqsubseteq} \mathsf{lcs}(L')$ for $L \supseteq L'$, and
- $\mathsf{lcs}(Lx) = \mathsf{lcs}(L)x$ for $x \in \{\top\} \cup \Sigma^*$.

We prove the following technical properties of the lcs which allow to simplify the computation of the lcs, in particular in the case of ultimately periodic words.

Lemma 1. *Let $u, v, w, x, y, z \in \Sigma^*$.*

1. *Let $w \neq \varepsilon$, then*

$$\begin{aligned} u \overset{s}{\sqsubseteq} w^\omega \quad &\textit{iff} \quad u \overset{s}{\sqsubset} uw \\ &\textit{iff} \quad \exists w', w'', k\colon w = w'w'' \wedge w'' \overset{s}{\sqsubset} w \wedge u = w''w^k \end{aligned}$$

2. $$\begin{aligned} u \overset{s}{\sqsubseteq} w^\omega \quad &\textit{iff} \quad u \overset{s}{\sqsubseteq} uw \\ &\textit{iff} \quad \exists v\colon uw = vu \wedge v^\omega u = w^\omega \end{aligned}$$

3. *Let $u, w \neq \varepsilon$, then*

$$u^\omega v = w^\omega \quad \textit{iff} \quad \exists p, q \in \Sigma^*: pv = vq \wedge u \in p^* \wedge w \in q^*$$

4. $$\begin{aligned} \mathsf{lcs}(w, wu) &= \mathsf{lcs}(w, wu^k) \quad (\forall k \geq 1) \\ &= \mathsf{lcs}(w, u^\omega) \end{aligned}$$

5. $$\mathsf{lcs}(w^\omega, u^\omega) = \begin{cases} (wu)^\omega & \textit{if } wu = uw \\ \mathsf{lcs}(uw, u^\omega) = \mathsf{lcs}(wu, w^\omega) = \mathsf{lcs}(wu, uw) & \textit{if } wu \neq uw \end{cases}$$

6. $$\mathsf{lcs}(u^\omega, (wu)^\omega) = \mathsf{lcs}(u^\omega, w^\omega)u$$

7. $$\mathsf{lcs}(u^\omega, w^\omega, (wu)^\omega) = \mathsf{lcs}(u^\omega, w^\omega)$$

Proof. 1. $u \overset{s}{\sqsubseteq} w^\omega \overset{s}{\sqsubset} \top$ iff $u \overset{s}{\sqsubset} uw$ iff $\exists w', w'', k\colon w = w'w'' \wedge w'' \overset{s}{\sqsubset} w \wedge u = w''w^k$ with $w \neq \varepsilon$

Assume that there are w', w'', k such that $w = w'w''$, $w'' \overset{s}{\sqsubset} w$ and $u = w''w^k$. Then $u = w''w^k \overset{s}{\sqsubset} w^{k+1} \overset{s}{\sqsubset} w''w^k w = uw$.

Assume $u \overset{s}{\sqsubset} uw$. Then, we show by induction on $|u|$ that $u \overset{s}{\sqsubseteq} w^\omega$. If $|u| \leq |w|$, then $w = w'u$ and therefore $u \overset{s}{\sqsubseteq} w \overset{s}{\sqsubset} w^\omega$. Thus, assume $|u| > |w|$ such that $u = u'w$. Then, $u' \overset{s}{\sqsubset} u'w$ as $u'w = u \overset{s}{\sqsubset} uw = u'ww$. Hence, by induction $u' \overset{s}{\sqsubseteq} w^\omega \overset{s}{\sqsubset} \top$ and thus $u' \overset{s}{\sqsubseteq} w^\omega w = w^\omega$.

Assume $u \overset{s}{\sqsubseteq} w^\omega$. Then, there is $k \in \mathbb{N}_0$ such that $w^k \overset{s}{\sqsubseteq} u \overset{s}{\sqsubset} w^{k+1}$ as $w \neq \varepsilon$. Thus, $u = w''w^k$ for some factorization $w = w'w''$ with $w'' \overset{s}{\sqsubset} w$.

2. $u \overset{s}{\sqsubseteq} w^\omega$ iff $u \overset{s}{\sqsubseteq} uw$ iff $\exists v\colon uw = vu \wedge w^\omega = v^\omega u$

   Assume $w = \varepsilon$. Then $u \overset{s}{\sqsubseteq} w^\omega = \top$ and $u \overset{s}{\sqsubseteq} uw = u$ and $\exists v\colon u = uw = wv = v \wedge w^\omega = \top = v^\omega u$ follow directly. Thus, $w \neq \varepsilon$ from now on.

   Assume $u \overset{s}{\sqsubseteq} w^\omega \overset{s}{\sqsubset} \top$. Then $\exists w', w'', m\colon w = w'w'' \wedge w'' \overset{s}{\sqsubset} w \wedge u = w''w^m$ by preceding result. Let $v = w''w'$ such that $|v| = |w| > 0$. Then $uw = w''w^m w = (w''w')w''w^m = vu$. Hence $\forall k\colon w^k \overset{s}{\sqsubseteq} v^k u \overset{s}{\sqsubseteq} v^\omega v^k u = v^\omega u \wedge v^k u = uw^k \overset{s}{\sqsubseteq} w^\omega w^k = w^\omega$.

   Assume $\exists v\colon uw = vu$. Then $u \overset{s}{\sqsubset} uw = vu$ and therefore $u \overset{s}{\sqsubseteq} w^\omega \overset{s}{\sqsubset} \top$.

3. $u^\omega v = w^\omega$ iff $\exists p, q \in \Sigma^*\colon pv = vq \wedge u \in p^* \wedge w \in q^*$

   Assume $u^\omega = w^\omega$. Let p be the primitive root of u, and q that of w. Then, $u^\omega = p^\omega$ and $w^\omega = q^\omega$ s.t. $v \overset{s}{\sqsubseteq} w^\omega = q^\omega \overset{s}{\sqsubset} \top$. Thus, $\exists \hat{q}\colon vq = \hat{q}v$ s.t. $p^\omega v = q^\omega = \hat{q}^\omega v$. Therefore, $p^\omega = \hat{q}^\omega$ and thus $p = \hat{q}$ as both are primitive.

   Assume $\exists p, q \in \Sigma^*\colon pv = vq \wedge u \in p^* \wedge w \in q^*$ Let $u = p^i$, then $u^\omega v = (p^i)^\omega v = v(q^i)^\omega = q^\omega = w^\omega$.

4. $\mathsf{lcs}(w, wu) = \mathsf{lcs}(w, wu^k) = \mathsf{lcs}(w, u^\omega)$ for all $k \geq 1$

   If $u = \varepsilon$, then $\mathsf{lcs}(w, wu) = w = \mathsf{lcs}(w, wu^k) = \mathsf{lcs}(w, \top)$. So we assume $u \neq \varepsilon$ from here on. We pick m such that $w = w'u''u^m$ for suitable w', u', u'' with $u = u'u''$ and $u'' \overset{s}{\sqsubset} u$ and $\mathsf{lcs}(w, u) = u''u^m$ and $\mathsf{lcs}(w', u') = \varepsilon$. Hence for all $k \geq 1$,

$$\begin{aligned} \mathsf{lcs}(w, wu^k) &= \mathsf{lcs}(w'u''u^m, w'u''u^{m+k}) \\ &= \mathsf{lcs}(w'u''u^m, u^{m+1}) \\ &= \mathsf{lcs}(w', u')u''u^m \\ &= \mathsf{lcs}(w, u^\omega) \end{aligned}$$

5. $\mathsf{lcs}(w^\omega, u^\omega) = \begin{cases} (wu)^\omega & \text{if } wu = uw \\ \mathsf{lcs}(uw, u^\omega) = \mathsf{lcs}(wu, w^\omega) = \mathsf{lcs}(wu, uw) & \text{if } wu \neq uw \end{cases}$

   W.l.o.g. $|w| \leq |u|$. If $w = \varepsilon$, then $\mathsf{lcs}(w^\omega, u^\omega) = \mathsf{lcs}(\top, u^\omega) = u^\omega = (wu)^\omega$. Thus, we assume $0 < |w| \leq |u|$.

   Assume that $w \overset{s}{\not\sqsubseteq} wu \vee u \overset{s}{\not\sqsubseteq} uw$. W.l.o.g. consider the case $w \overset{s}{\not\sqsubseteq} wu$. Then $\mathsf{lcs}(w, u^\omega) = \mathsf{lcs}(w, wu^l) \overset{s}{\sqsubset} w$ for all $l > 0$. So we assume $w \overset{s}{\sqsubseteq} wu \wedge u \overset{s}{\sqsubseteq} uw$ from now on. Then it exists $\hat{w}, \hat{u}\colon wu = \hat{u}w \wedge uw = \hat{w}u$.

   Assume that $\hat{w} = w \vee \hat{u} = u$. Then $wu = uw$ as $wu = \hat{u}w = uw \vee uw = \hat{w}u = wu$. Hence, $w^\omega = u^\omega = (wu)^\omega = \mathsf{lcs}(w^\omega, u^\omega)$ as w, u have the same primitive root.

Now, assume that $\hat{w} \neq w \wedge \hat{u} \neq u$. Then $u = u'w$ as $|w| \leq |u|$ and $u \overset{s}{\sqsubseteq} uw = \hat{w}u$. Hence $\hat{u} = wu'$ as $\hat{u}w = wu = wu'w$. Then $\mathsf{lcs}(w^\omega, u^\omega) = \mathsf{lcs}(\hat{w}^\omega \hat{w}u, u^\omega u'wu) = \mathsf{lcs}(\hat{w}, w)u \overset{s}{\sqsubset} wu$. From this the second case of the statement follows.

6. $\mathsf{lcs}(u^\omega, (wu)^\omega) = \mathsf{lcs}(u^\omega, w^\omega)u$

$$\begin{aligned} \mathsf{lcs}(u^\omega, (wu)^\omega) &= \begin{cases} (wu)^\omega & \text{if } uw = wu \\ \mathsf{lcs}(uwu, wuu) = \mathsf{lcs}(uw, wu)u & \text{if } uw \neq wu \end{cases} \\ &= \mathsf{lcs}(u^\omega, w^\omega)u \end{aligned}$$

7. $\mathsf{lcs}(u^\omega, w^\omega, (wu)^\omega) = \mathsf{lcs}(u^\omega, w^\omega)$

$$\begin{aligned} \mathsf{lcs}(u^\omega, w^\omega, (wu)^\omega) &= \mathsf{lcs}(\mathsf{lcs}(u^\omega, w^\omega), \mathsf{lcs}(u^\omega, (wu)^\omega)) \\ &= \mathsf{lcs}(\mathsf{lcs}(u^\omega, w^\omega), \mathsf{lcs}(u^\omega, w^\omega)u) \\ &= \mathsf{lcs}(\mathsf{lcs}(u^\omega, w^\omega), u^\omega) \\ &= \mathsf{lcs}(u^\omega, w^\omega) \end{aligned}$$

□

The next lemma formalizes that whenever L is not empty, we can find for any $x \in L$ a witness $y \in L$ s.t. $\mathsf{lcs}(L) = \mathsf{lcs}(x, y)$ which we will use in the following frequently without explicitly referring to this lemma every time.

Lemma 2. *Let $L \subseteq \Sigma^*$ be nonempty. Then for any $x \in L$ we have $\mathsf{lcs}(L) = \mathsf{lcs}(\mathsf{lcs}(x, z) \mid z \in L)$; in particular, there is some* witness *$y \in L$ (w.r.t. x) s.t. $\mathsf{lcs}(L) = \mathsf{lcs}(x, y) = \mathsf{lcs}(x, y, z)$ for all $z \in L$.*

Proof. Trivially true if $L = \emptyset$. Therefore, assume $L \neq \emptyset$. Let $R = \mathsf{lcs}(L)$ and pick any $x \in L$. If $x = R$, we have $\mathsf{lcs}(x, y) = R$ for all $y \in L$; so choose any $y \in L$. Thus, assume $R \overset{s}{\sqsubset} x$. Let $L_x = \{\mathsf{lcs}(x, y) \mid y \in L\}$ and $S := \mathsf{lcs}(L_x)$. Then $R \overset{s}{\sqsubseteq} S$ as for all $y \in L$ we have $R \overset{s}{\sqsubseteq} y$ and thus $R \overset{s}{\sqsubseteq} \mathsf{lcs}(x, y)$. But also $S \overset{s}{\sqsubseteq} R$ as $\forall y \in L$: $S \overset{s}{\sqsubseteq} \mathsf{lcs}(x, y) \overset{s}{\sqsubseteq} y$. Thus, $R \in L_x$ as $\forall z \in L_x$: $R \overset{s}{\sqsubseteq} z \overset{s}{\sqsubseteq} x$. Then there is some $y \in L$ s.t. $R = \mathsf{lcs}(x, y)$; by minimality of R we trivially have that $\mathsf{lcs}(x, y, z) = \mathsf{lcs}(x, y)$ for all $z \in L$. □

Involutive Monoid. We briefly recall the basic definitions and properties of the finitely generated involutive monoid, but refer the reader for details and a formal treatment to e.g. [Sak]. Let A be a finite alphabet (of opening brackets/letters). From A we derive the alphabet $\overline{\mathsf{A}} := \{\overline{a} \mid a \in \mathsf{A}\}$ of closing letters where we assume that $\mathsf{A} \cap \overline{\mathsf{A}} = \emptyset$. Set $\mathsf{B} := \mathsf{A} \cup \overline{\mathsf{A}}$. We use Latin letters $p, q, \ldots$ to denote words over A, while Greek letters $\alpha, \beta, \gamma, \ldots$ will denote words over B. We extend $\bar{\cdot}$ to an involution on B^* by means of $\overline{\varepsilon} := \varepsilon$, $\overline{\overline{a}} := a$ for all $a \in \mathsf{A}$, and $\overline{\alpha\beta} := \overline{\beta}\,\overline{\alpha}$ for all other $\alpha, \beta \in \mathsf{B}^*$. Let $\xrightarrow{\rho}$ be the binary relation on B^* defined by $\alpha a \overline{a} \beta \xrightarrow{\rho} \alpha\beta$ for any $\alpha, \beta \in \mathsf{B}^*$ and $a \in \mathsf{A}$, i.e. $\xrightarrow{\rho}$ cancels nondeterministically one pair of matching opening and closing brackets. A word $\alpha \in \mathsf{B}^*$ is *reduced* if it does not contain any infix of the form $a\overline{a}$ for any $a \in \mathsf{A}$, i.e. α is reduced if and only if it has no direct successor w.r.t. $\xrightarrow{\rho}$. For every $\alpha \in \mathsf{B}^*$ canceling all matching brackets in any arbitrary order always results in the same unique reduced word which we denote by $\rho(\alpha)$; we write $\alpha \overset{\rho}{=} \beta$ if $\rho(\alpha) = \rho(\beta)$. Then $\mathsf{B}^* / \overset{\rho}{=}$ is the free involutive monoid generated by A, and $\rho(\alpha)$ is the shortest word in the $\overset{\rho}{=}$-equivalence class of α. For $L \subseteq \mathsf{B}^*$ we set $\rho(L) := \{\rho(w) \mid w \in L\}$.

Well-formed Languages and Context-free Grammars. We introduce the notation of context-free grammars (CFGs) that we use in the following. We write $\to_G$ for the rewrite rules of the context-free grammar G and assume that G is reduced to the productive nonterminals that are reachable from its axiom S. For simplicity, we assume for the proofs and constructions that the rules of G are of the form

$$X \to_G YZ \qquad X \to_G Y \qquad X \to_G \overline{u}\, v$$

for nonterminals X, Y, Z and $u, v \in \mathsf{A}^*$. We write $L_X := \{\alpha \in \mathsf{B}^* \mid X \to_G^* \alpha\}$ for the language generated by the nonterminal X. Specifically for the axiom S of G we set $L := L_S$. The height of a derivation tree w.r.t. G is measured in the maximal number of nonterminals occurring along a path from the root to any leaf, i.e. any derivation tree has height at least 1. We write $L_X^{\leq h}$ for the subset of L_X of words that possess a derivation tree of height at most h s.t.

$$L_X^{\leq 1} = \{\overline{u}\, v \mid X \to_G \overline{u}\, v\} \quad L_X^{\leq h+2} = L_X^{\leq h+1} \cup \bigcup_{X \to_G YZ} L_Y^{\leq h+1} L_Z^{\leq h+1} \cup \bigcup_{X \to_G Y} L_Y^{\leq h+1}$$

We will also write $L_X^{<h}$ for $L_X^{\leq h-1}$ and $L_X^{=h}$ for $L_X^{\leq h} \setminus L_X^{<h}$. The *prefix closure* of $L \subseteq \mathsf{B}^*$ is denoted by $\mathsf{Prf}(L) := \{\alpha' \mid \alpha'\alpha'' \in L\}$. Given a derivation tree of G that yields the word κ, a path within this tree, and a specific nonterminal X of G, we may factorize κ into the product of *(word) contexts* (finite words with a "hole" which represent a pumping tree w.r.t. G) (α, β), (σ_1, τ_1), …, (σ_k, τ_k) and a single word γ s.t. $S \to_G^* \alpha X \beta$, $X \to_G^* \sigma_i X \tau_i$, and $X \to_G^* \gamma$. We denote such factorizations by simply writing $\kappa = (\alpha, \beta)(\sigma_1, \tau_1) \ldots (\sigma_k, \tau_k)\gamma$. Concatenation of contexts with contexts resp. words is thus defined by means of substituting the right operand into the "hole" of the context, i.e. $(\sigma, \tau)(\mu, \nu) = (\sigma\mu, \nu\tau)$ and $(\sigma, \tau)\gamma = \sigma\tau\gamma$. Given such a factorization, we are often interested in the "simple" linear language induced by it:

$$(\alpha, \beta)[(\sigma_1, \tau_1) + \ldots + (\sigma_k, \tau_k)]^*\gamma := \{\alpha\sigma_{i_1} \ldots \sigma_{i_k}\gamma\tau_{i_k} \ldots \tau_{i_1}\gamma \mid i_1 \ldots i_l \in \{1, \ldots, k\}^*\}$$

Formally, this is the language generated by the "simple" linear grammar

$$S \to \alpha X \beta \qquad X \to \sigma_1 X \tau_1 \mid \ldots \mid \sigma_k X \tau_k \mid \gamma$$

and is thus always a sublanguage of $L(G)$.

Definition 2 ((Weakly) Well-formedness, Height difference)**.** *Let $\alpha \in \mathsf{B}^*$ and $L \subseteq \mathsf{B}^*$.*

1. *Let $\Delta(\alpha) := |\alpha|_{\mathsf{A}} - |\alpha|_{\overline{\mathsf{A}}}$ be the difference of opening brackets to closing brackets. α is* nonnegative *if $\forall \alpha' \overset{p}{\sqsubseteq} \alpha \colon \Delta(\alpha') \geq 0$. $L \subseteq \mathsf{B}^*$ is* nonnegative *if every $\alpha \in L$ is nonnegative.*
2. *A context-free grammar G with $L(G) \subseteq \mathsf{B}^*$ is* nonnegative *if $L(G)$ is nonnegative. For a nonterminal X of G let $d_X := \sup(\{-\Delta(\alpha') \mid \alpha'\alpha'' \in L_X\} \cup \{0\})$. Note that $d_X \geq 0$ as we can always choose $\alpha' = \varepsilon$ in the definition of d_X.*
3. *A word α is* weakly well-formed (wwf for short) resp. well-formed (wf for short) *if $\rho(\alpha) \in \overline{\mathsf{A}}^* \mathsf{A}^*$ resp. if $\rho(\alpha) \in \mathsf{A}^*$. A context-free grammar G is wf if $L(G)$ is wf. $L \subseteq \mathsf{B}^*$ is* wwf resp. wf *if every word of L is wwf resp. wf.*
4. *A context-free grammar G is* bounded well-formed (bwf for short) *if it is wwf and for every nonterminal X there is a (shortest) word $r_X \in \mathsf{A}^*$ with $|r_X| = d_X$ s.t. $r_X L_X$ is wf.*

We summarize some consequences of Definition 2.

Remark 1. *Let $L \subseteq \mathsf{B}^*$.*

- *L is wf if and only if $\mathsf{Prf}(L)$ is wf if and only if L is a subset of the prefix closure of the Dyck language generated by $S \to \varepsilon \quad S \to SS \qquad S \to aS\overline{a}$ $(a \in \mathsf{A})$.*
- *L is nonnegative if and only if the image of L under the homomorphism that collapses A to a singleton is wf. Hence, if L is wf, then L is nonnegative.*
- *Δ is an ω-continuous homomorphism from the language semiring generated by B to the tropical semiring $\langle \mathbb{Z} \cup \{-\infty\}, \min, +\rangle$. Thus it is decidable in polynomial time if G is nonnegative using the Bellman-Ford algorithm.*
- *If L is not wf, then there exists some $\alpha \in \mathsf{Prf}(L) \setminus \{\varepsilon\}$ s.t. $\Delta(\alpha) < 0$ or $\alpha \overset{\rho}{=} ua\overline{b}$ for $u \in \mathsf{A}^*$ and $a, b \in \mathsf{A}$ (with $a \neq b$).*
- *If L_X is wwf, then $d_X = \sup\{|y| \mid \gamma \in L_X, \rho(\gamma) = \overline{y}\, z\}$.*

In particular, because G is context-free it follows that, if G is wf, then for every nonterminal X there is $r_X \in \mathsf{A}^*$ s.t.

- $\overline{r_X} \in \rho(\mathsf{Prf}(L_X))$,
- $|r_X| = d_X$ and
- $r_X L_X$ is wf.

Lemma 3. *A context-free grammar G is well-formed iff G is bounded well-formed with $r_S = \varepsilon$ for S the axiom of G.*

Proof. Let S be the axiom of G. W.l.o.g. G is reduced to the nonterminals which are reachable from S and which are productive. First, assume that G is well-formed. Then for every nonterminal X of G we can define its *left-context* $L^l_X = \{\alpha \in \mathsf{B}^* \mid S \to^*_G \alpha X \beta\}$. As $L := L(G)$ is well-formed and every $\alpha \in L^l_X$ is a prefix of some word of L, also L^l_X is well-formed; hence $m_X := \min\{\Delta(L^l_X)\} \geq 0$ is defined. Fix any $\lambda_X \in L^l_X$ with $\Delta(\lambda_X) = m_X$. Then for any $\gamma = \gamma'\gamma'' \in L_X$ also $\lambda_X \gamma'$ is a prefix of a word of L, thus well-formed, and therefore $\Delta(\lambda_X \gamma') \geq 0$. It follows that $d_X = \max\{-\Delta(\gamma') \mid \gamma'\gamma'' \in L_X\} \leq m_X$. In particular, there is a $\gamma_X = \gamma'_X \gamma''_X \in L_X$ s.t. $-\Delta(\gamma'_X) = d_X$; as L is well-formed, L_X has to be weakly well-formed s.t. $\gamma'_X \overset{\rho}{=} \overline{r_X}\, s_X$ and $\lambda_X \overset{\rho}{=} x r_X$. Hence, for every $\gamma \in L_X$ we have $\gamma \overset{\rho}{=} \overline{y}\, z$ and $\lambda_X \gamma \overset{\rho}{=} x r_X \overline{y}\, z$ are well-formed, i.e. $y \overset{s}{\sqsubseteq} r_X$ and thus $r_X L_X$ is well-formed. In particular, we have $r_S = \varepsilon$ for the axiom S.
Now, assume that G is bwf and thus by definition also nonnegative. We fix for every X any $r_X \in \mathsf{A}^*$ s.t. $r_X L_X$ is well-formed; hence, L_X is weakly well-formed and $d_X = \max\{|y| \mid \gamma \in L_X, \rho(\gamma) = \overline{y}\, z\}$. Then $\hat{r}_X := \max^{\overset{s}{\sqsubseteq}}\{y \mid \gamma \in L_X, \rho(\gamma) = \overline{y}\, x\} \overset{s}{\sqsubseteq} r_X$ is well defined with $d_X = |r_X|$. As G is also nonnegative, we have $d_S = 0$ resp. $r_S = \varepsilon$ and thus $L = L_S$ is well-formed. □

The words r_X mentioned in Definition 2 can be computed in polynomial time using the Bellman-Ford algorithm similar to [TM07]; more precisely, a *straight-line program (SLP for short)* (see e.g. [Loh12] for more details on SLPs), i.e. a context-free grammar generating exactly one word, can be extracted from G for each r_X. We therefore use the prefix closure of a context-free grammar G, that can be constructed in polynomial time.

Remark 2. *Let G be a context-free grammar over the nonterminals $\mathfrak{X}$. Define G_p by the following rules:*

- *If $X \to_G YZ$, then $X \to_{G_p} YZ$, $X_p \to_{G_p} Y_p$, and $X_p \to_{G_p} YZ_p$ and $X_p \to_{G_p} YZ$.*

- *If $X \to_G Y$, then $X \to_{G_p} Y$, $X_p \to_{G_p} Y_p$, and $X_p \to_{G_p} Y$.*

- *If $X \to_G \overline{u}\, v$, then $X \to_{G_p} \overline{u}\, v$, and $X_p \to_{G_p} \overline{u}\, v$*

Then $L_X(G) = L_X(G_p)$ and $L_{X_p}(G_p) \cup \{\varepsilon\} = \mathsf{Prf}(L_X(G))$. In particular, we can construct G_p in time polynomial in the size of G.

Lemma 4. *Let $L = L(G)$ be wf. Let X be some nonterminal of G. Derive from G the CFG G^p s.t. $\mathsf{Prf}(L_X) = L(G^p_X)$. Let $r_X \in \mathsf{A}^*$ be the shortest word s.t. $r_X L_X$ is wf.*

1. *Then r_X is also the shortest word s.t. $r_X \mathsf{Prf}(L_X)$ is wf.*
2. *There is a shortest $\alpha \in \mathsf{Prf}(L_X)$ s.t. $\rho(\alpha) = \overline{r_X}$.*
3. *Every shortest $\alpha \in \mathsf{Prf}(L_X)$ with $\rho(\alpha) = \overline{r_X}$ has a derivation tree w.r.t. G^p_X that does not contain any pumping tree and thus has height bounded by the number of nonterminals of G^p_X.*
4. *An SLP for r_X can be computed in polynomial time.*

Proof.

1. As $L_X \subseteq \mathsf{Prf}(L_X)$, we only need to show that $u_X\mathsf{Prf}(L_X)$ is still wf. For every $\alpha' \in \mathsf{Prf}(L_X)$ there is some $\alpha \in L_X$ s.t. $\alpha = \alpha'\alpha''$ (by definition). As L is wf, L_X is wwf, hence α is wwf, and thus α' and α'' are wwf, too. Hence, $\rho(\alpha') = \overline{r}\, s$, $\rho(\alpha'') = \overline{u}\, v$ and $\rho(\alpha) = \overline{x}\, y$ for some $r, s, u, v, x, y \in \mathsf{A}^*$ s.t. $\overline{r}\, s \overline{u}\, v \overset{\rho}{=} \overline{x}\, y$. If $s = s'u$, then $\overline{r} \; = \overline{x}$; else $u = u's$ and $\overline{x} \; = \overline{r}\; \overline{u'}$. Thus $\overline{r} \overset{p}{\sqsubseteq} \overline{x} \overset{p}{\sqsubseteq} \overline{u_X}$.
2. There is some $\beta \in L_X$ s.t. $\rho(\beta) = \overline{u_X}\, y$ for some $y \in \mathsf{A}^*$. Then there is some prefix $\beta' \overset{p}{\sqsubseteq} \beta$ s.t. $\rho(\beta') = \overline{u_X}$. By definition, $\beta' \in \mathsf{Prf}(L_X)$. Hence, there is also some shortest $\alpha \in \mathsf{Prf}(L_X)$ s.t. $\rho(\alpha) = \overline{u_X}$.
3. Let $\alpha \in \mathsf{Prf}(L_X) = L(G^p_X)$ be a shortest word s.t. $\rho(\alpha) = \overline{u_X}$. Assume that there is some factorization $\alpha = \beta\rho\gamma\varrho\delta$ s.t. $\beta\rho^k\gamma\varrho^k\delta \in L(G^p_X)$ for all $k \in \mathbb{N}_0$.

   Consider $k = 0$ and let $\rho(\beta\gamma\delta) = \overline{r}\, s$. If $r = u_X$, then there would be a prefix of $\beta\gamma\delta$ that would reduce to $\overline{u_X}$ contradicting our assumption that α is a shortest such word. Hence, $\overline{r} \overset{p}{\sqsubset} \overline{u_X}$. Thus $\Delta(\beta\gamma\delta) \geq -|r| > -|u_X|$.

   Note that $-|u_X| = \Delta(\alpha) = \Delta(\beta\gamma\delta) + \Delta(\rho\varrho)$. Thus, $\Delta(\rho\varrho) < 0$; a contradiction to the wfness of $u_X\mathsf{Prf}(L_X)$.
4. Split every nonterminal Y of G^p_X into $N + 1$ copies $Y_0, \ldots, Y_N$, split every rule $Y \to UV$ into the rules $Y_{i+1} \to U_iV_i \mid Y_i$, and derive from every rule $Y \to \gamma$ the rule $Y_0 \to \gamma$. In other words, unfold G^p_X into an acyclic grammar that generates exactly all derivation trees of height at most N. We compute inductively for every nonterminal a pair of SLPs representing a wwf word $\overline{u}\, v$ as follows: For every rule $Y_0 \to \gamma \in \mathsf{B}$, we choose either $u = \gamma$, $v = \varepsilon$ or $u = \varepsilon$,

$v = \gamma$ such that $\overline{u}\, v = \gamma$. For every rule $Y_{i+1} \to U_i V_i$, we have inductively computed SLPs for U_i and V_i representing words $\overline{r}\, s$ and $\overline{u}\, v$, respectively. Then we can compute SLPs representing the reduced word $\rho(\overline{r}\, s \overline{u}\, v)$: either $s = s'u$ (i.e. $|s| \geq |u|$) or $u = u's$ (i.e. $|s| \leq |u|$), i.e. we simply have to restrict and then concatenate the respective SLPs. For the rule $Y_{i+1} \to Y_i$ there is nothing to do. We are thus left for Y_{i+1} with a family of SLPs representing words $\overline{u_i}\, v_i$: w.l.o.g. assume $|u_0| \geq |u_i|$ for all i; as for every derivation $X_N \to^* \alpha Y_{i+1} \gamma$ we need to have that $\alpha \overline{u_i}\, v_i \gamma$ is wwf for every i, we also have that $\alpha \overline{u_0}\, u_0 \overline{u_i}\, v_i \gamma$ is wwf for every i. We thus may normalize all SLP pairs by means of $\overline{u_i}\, v_i \mapsto \overline{u_0}\ \rho(u_0 \overline{u_i}\,) v_i$. As we want to maximize the descent, we then assign to Y_{i+1} the pair of SLPs encoding $\overline{u_0}$ and the shortest of all $\rho(u_0 \overline{u_i}\,) v_i$. This amounts to a constant amount of SLP operations per rule of the unfolded grammar.

□

We show that deciding whether a context-free grammar G produces a negative word, i.e., a word α such that $\Delta(\alpha) < 0$ can be done considering only derivation trees up to height N with N the number of nonterminals in G.

Lemma 5. *Let $L = L(G) = \mathit{Prf}(L(G))$ be a prefix-closed context-free language. We can decide in time polynomial in G whether there is a word $\alpha \in L$ s.t. $\Delta(\alpha) < 0$.*

Proof. Let N be the number of nontermimals of G. Assume there is a word $\alpha \in L$ with $\Delta(\alpha) < 0$, then w.l.o.g. $\Delta(\alpha) = -1$ as L is prefix-closed. Pick any shortest such $\alpha \in L$ with $\Delta(\alpha) = -1$.
If α has a derivation tree of height at most N, then we simply apply standard fixed-point/Kleene iteration to the operator F obtained from the rewrite rules of G via the homomorphism Δ over the tropical semiring

$$F(\mathfrak{X})_X := \min\{\Delta(Y) + \Delta(Z), \Delta(Y), \Delta(\gamma) \mid X \to_G YZ, X \to_G Y, X \to_G \gamma\}$$

Then $F^N(\infty)_S = \min\{\Delta(\beta) \mid \beta \in L^{\leq N}\} \leq \Delta(\alpha) = -1$ with $L^{\leq N}$ all words of L that possess a derivation tree of height at most N.
Assume thus that every such α has a derivation tree of height at least $N+1$. Pick a longest path from the root to a leaf in such a derivation tree, and moving bottom-up along this path, pick the first nonterminal X occurring a second time in order to obtain a factorization $\alpha = \beta\rho\gamma\varrho\delta$ s.t. $\beta\rho^k\gamma\varrho^k\delta \in L$ for all $k \in \mathbb{N}_0$ and $\rho\varrho \neq \varepsilon$; in particular, note that $\rho\varrho \in L_X^{\leq N}$. Then $-1 = \Delta(\alpha) = \Delta(\beta\gamma\delta) + \Delta(\rho\varrho)$ and $\Delta(\beta\gamma\delta) \geq 0$; otherwise there would be a prefix π of $\beta\gamma\delta$ with $\Delta(\pi) = -1$ contradicting the minimality of α. Hence, $\Delta(\rho\varrho) \leq -1$. Thus, we only need to decide whether there is a pumpable derivation tree $X \to_G^{\leq N} \rho X \varrho$ of height at most N s.t. $\Delta(\rho\varrho) < 0$. This can be done by transforming the rewrite rules $X \to YZ$

into weighted edges $X \xrightarrow{F_Y^N(\infty)} Z$ and $X \xrightarrow{F_Z^N(\infty)} Y$, and then check for negative cycles in this graph. (This amounts to take the derivative of F at $F^N(\infty)$.) □

2.2 lcs-equivalent Sublanguages

We show that we can compute the longest common suffix of the union $L \cup L'$ and the concatenation LL' of two languages $L, L' \subseteq \Sigma^*$ if we know both $\mathsf{lcs}(L)$ and $\mathsf{lcs}(L')$, and in addition, the longest word $\mathsf{lcsext}(L)$ resp. $\mathsf{lcsext}(L')$ by which we can extend $\mathsf{lcs}(L)$ resp. $\mathsf{lcs}(L')$ when concatenating another language from left. Analogous results w.r.t. the lcp have been presented already in [LPS18] but the *maximal prefix extension* is not studied there explicitly.
We then show that the relation $L \approx_{\mathsf{lcs}} L' \Leftrightarrow \mathsf{lcs}(L) = \mathsf{lcs}(L') \wedge \mathsf{lcsext}(L) = \mathsf{lcsext}(L')$ is an equivalence relation on Σ^* that respects both union and concatenation of languages (see Lemmas 8 and 9). It follows that for every language $L \subseteq \Sigma^*$ there is some subset $\mathsf{T}_{\mathsf{lcs}}(L) \subseteq L$ of size at most 3 with $L \approx_{\mathsf{lcs}} \mathsf{T}_{\mathsf{lcs}}(L)$.
In this section we do not consider the involution, thus let Σ denote an arbitrary alphabet.

Definition 3. *For $L \subseteq \Sigma^*$ with $R = \mathsf{lcs}(L)$ the* maximal suffix extension *(lcsext) of L is defined by $\mathsf{lcsext}(L) := \mathsf{lcs}(z^\omega \mid zR \in L)$.*
For $L, L' \subseteq \Sigma^*$, $L \approx_{\mathsf{lcs}} L'$ if $\mathsf{lcs}(L) = \mathsf{lcs}(L')$ and $\mathsf{lcsext}(L) = \mathsf{lcsext}(L')$, i.e., if both the longest common suffix and the maximal suffix extension are the same.

Note that by definition we have both $\mathsf{lcsext}(\emptyset) = \mathsf{lcs}(\emptyset) = \top$ and $\mathsf{lcsext}(\{R\}) = \mathsf{lcs}(\varepsilon^\omega) = \top$. The definition of lcsext can be motivated as follows:

Example 8. *Consider the language $L = \{R, xR, yR\}$ with $\mathsf{lcs}(L) = R$ and $\mathsf{lcsext}(L) = \mathsf{lcs}(x^\omega, y^\omega)$. Assume we prepend some word $u \in \Sigma^*$ to L resulting in the language $uL = \{uR, uxR, uyR\}$. Then $\mathsf{lcs}(uL)$ is given by $\mathsf{lcs}(u, \mathsf{lcsext}(L))\,\mathsf{lcs}(L)$:*

$$\begin{aligned}
\mathsf{lcs}(u\{xR, yR, R\}) &= \mathsf{lcs}(u, ux, uy)R \\
&= \mathsf{lcs}(\mathsf{lcs}(u, ux), \mathsf{lcs}(u, uy))R \quad (\text{as } \mathsf{lcs}(u, ux) = \mathsf{lcs}(u, x^\omega)) \\
&= \mathsf{lcs}(\mathsf{lcs}(u, x^\omega), \mathsf{lcs}(u, y^\omega))R \\
&= \mathsf{lcs}(u, \mathsf{lcs}(x^\omega, y^\omega))R \\
&= \mathsf{lcs}(u, \mathsf{lcsext}(L))\,\mathsf{lcs}(L)
\end{aligned}$$

In particular, if $xy = yx$, we can extend lcs by any finite suffix of $\mathsf{lcsext}(L) = (xy)^\omega$ — note that, if $x = \varepsilon = y$, then $\mathsf{lcsext}(L) = \top$ by our definition that $\varepsilon^\omega = \top$; but if $xy \neq yx$, we can extend it at most to $\mathsf{lcsext}(L) = \mathsf{lcs}(x^\omega, y^\omega) = \mathsf{lcs}(xy, yx) \overset{s}{\sqsubset} xy$. □

Corollary 1. *The maximal suffix extension $\mathsf{lcsext}(L)$ is the longest word $u \in \Sigma^\infty$ such that $\mathsf{lcs}(uL) = u\,\mathsf{lcs}(L)$.*

If $\mathsf{lcs}(L)$ is not contained in L, then $\mathsf{lcs}(L)$ has to be a strict suffix of every shortest word in L, and thus immediately $\mathsf{lcsext}(L) = \varepsilon$. As in the case of the lcs, also $\mathsf{lcsext}(L)$ is already defined by two words in L.

Lemma 6. *Let $L \subseteq \Sigma^*$ with $|L| \geq 2$ and $R := \mathsf{lcs}(L)$. Fix any $xR \in L \setminus \{R\}$. Then there is some $yR \in L \setminus \{R\}$ such that for all $zR \in L$,*

$$\mathsf{lcsext}(L) = \mathsf{lcs}(x^{\mho}, y^{\mho}) = \mathsf{lcs}(x^{\mho}, y^{\mho}, z^{\mho}) = \begin{cases} x^{\mho} & \text{if } xy = yx \\ \mathsf{lcs}(xy, yx) & \text{if } xy \neq yx \end{cases}$$

If $xy = yx$, then $R \in L$.

Proof. Note that $x \neq \varepsilon$ as $xR \neq R$. Let p be the primitive root of x. If $\forall zR \in L\colon zx = xz$, then $\forall zR \in L\colon z \in p^*$. Thus $\mathsf{lcsext}(L) = p^{\mho} = x^{\mho} = z^{\mho} = \mathsf{lcs}(x^{\mho}, z^{\mho})$ and $R \in L$ as otherwise $R = \mathsf{lcs}(L) = p^i R$ for $p^i R$ the shortest word in L.
So assume $\exists zR \in L\colon zx \neq xz$, then $z \neq \varepsilon$ and $\mathsf{lcsext}(L) = \mathsf{lcs}(\mathsf{lcs}(x^{\mho}, y^{\mho}) \mid yR \in L \setminus \{R\}) \overset{s}{\sqsubseteq} \mathsf{lcs}(x^{\mho}, z^{\mho}) = \mathsf{lcs}(xz, zx)$. Hence, there is some $yR \in L \setminus \{R\}$ s.t. $\mathsf{lcsext}(L) = \mathsf{lcs}(x^{\mho}, y^{\mho}) = \mathsf{lcs}(xy, yx)$. □

The next lemma allows to reduce the computation of maximal suffix extension lcsext to that of the longest common suffix.

Lemma 7. *Let $L \subseteq \Sigma^*$ with $R = \mathsf{lcs}(L)$. If $\mathsf{lcsext}(L) \in \Sigma^*$, then for all $xR \in L \setminus \{R\}$ there exists $m \in \mathbb{N}$ such that*

$$\mathsf{lcs}(x^m L) = \mathsf{lcsext}(L)\,\mathsf{lcs}(L) \quad \text{and} \quad \mathsf{lcsext}(L) \overset{s}{\sqsubset} x^m$$

Proof. Fix any $xR \in L \setminus \{R\}$. Thus $x \neq \varepsilon$ s.t. there is some $m \in \mathbb{N}$ with $|x^m| > |\mathsf{lcsext}(L)|$. As $\mathsf{lcsext}(L) \in \Sigma^*$ there is some $y \in L$ s.t. for all $zR \in L$,

$$\begin{aligned} \mathsf{lcsext}(L) &= \mathsf{lcs}(x^{\mho}, y^{\mho}) \\ &= \mathsf{lcs}(xy, yx) \\ &= \mathsf{lcs}(x^{\mho}, y^{\mho}, z^{\mho}) \\ &= \mathsf{lcs}\begin{pmatrix} \mathsf{lcs}(x^{\mho}, y^{\mho}) \\ \mathsf{lcs}(x^{\mho}, z^{\mho}) \end{pmatrix} \overset{s}{\sqsubset} x^m \overset{s}{\sqsubseteq} x^{\mho} \end{aligned}$$

Pick any $zR \in L$. If $\mathsf{lcs}(x^{\mho}, z^{\mho}) \overset{s}{\sqsupseteq} x^m$, then $\mathsf{lcs}(x^{\mho}, z^{\mho}) \overset{s}{\sqsupseteq} \mathsf{lcs}(x^m, z^{\mho}) = x^m$.
If $\mathsf{lcs}(x^{\mho}, z^{\mho}) \overset{s}{\sqsubset} x^m$, then $\mathsf{lcs}(x^{\mho}, z^{\mho}) = \mathsf{lcs}(x^m, z^{\mho}) = \mathsf{lcs}(x^m, x^m z)$ (Lemma 1). In particular for $z = y$ we thus have

$$\mathsf{lcsext}(L) = \mathsf{lcs}(x^{\mho}, y^{\mho}) = \mathsf{lcs}(x^m, x^m y)$$

Hence,
$$\begin{aligned}\mathsf{lcs}(x^m L) = \mathsf{lcs}(x^m zR \mid zR \in L) &= \mathsf{lcs}(x^m xR, x^m yR, x^m zR \mid zR \in L)\\ &= \mathsf{lcs}(x^m xR, x^m yR)\\ &= \mathsf{lcs}(x^m, x^m y)R\\ &= \mathsf{lcsext}(L)R\end{aligned}$$

□

Example 9. *Consider $L = \{s^k t^k R \mid k \in \mathbb{N}_0\}$ with $R = \mathsf{lcs}(L)$. By definition we have $\mathsf{lcsext}(L) = \mathsf{lcs}((st)^\omega, (s^{k+2}t^{k+2})^\omega \mid k \geq 0)$. If s and t commute, then $\mathsf{lcsext}(L) = (st)^\omega$, and thus $\mathsf{lcsext}(L)$ is unbounded. Thus assume $st \neq ts$ s.t. $\mathsf{lcs}(ts, st^{k+1}) = \mathsf{lcs}(ts, st) \overset{s}{\sqsubset} ts, st$ and*

$$\mathsf{lcs}((st)^\omega, (s^{k+2}t^{k+2})^\omega) = \mathsf{lcs}((ts)^\omega, (ts^{k+2}t^{k+1})^\omega)t = \mathsf{lcs}(tst, stt)$$

Hence, $\mathsf{lcs}(ststL) = \mathsf{lcs}(tst, stt)R = \mathsf{lcsext}(L)\,\mathsf{lcs}(L)$.

We show that we can compute the lcs and the extension lcsext of the union respectively the concatenation of two languages solely from their lcs and lcsext. To this end, we define the lcs-*summary* of a language.

Definition 4. *For a language $L \subseteq \Sigma^*$ we define the lcs-summary $\mathsf{lcssum}(L) = (\mathsf{lcs}(L), \mathsf{lcsext}(L))$. The equivalence relation $\approx_{\mathsf{lcs}}$ on 2^{Σ^*} is defined by*

$$L \approx_{\mathsf{lcs}} L' \text{ iff } \mathsf{lcssum}(L) = \mathsf{lcssum}(L')$$

Lemma 8. *Let $L, L' \subseteq \Sigma^*$ with $\mathsf{lcssum}(L) = (R, E)$ and $\mathsf{lcssum}(L') = (R', E')$, then*

$$\mathsf{lcssum}(L \cup L') = \begin{cases} (R', E') & \text{if } R = \top \\ (R, E) & \text{if } R' = \top \\ (R, \mathsf{lcs}(E, \mathsf{lcs}(E', E'\delta)\delta)) & \text{if } R' = \delta R \overset{s}{\sqsubset} \top \\ (R', \mathsf{lcs}(E', \mathsf{lcs}(E, E\delta)\delta)) & \text{if } R = \delta R' \overset{s}{\sqsubset} \top \\ (\mathsf{lcs}(R, R'), \varepsilon) & \text{else} \end{cases}$$

Proof. Assume that $R' = \top$, then $L' = \emptyset$ and therefore $\mathsf{lcssum}(L \cup L') = \mathsf{lcssum}(L) = (R, E)$. The case $R = \top$ is symmetric.

W.l.o.g. $R \neq \top \neq R'$ from here on. If $R \overset{s}{\not\sqsubseteq} R'$ and $R' \overset{s}{\not\sqsubseteq} R$ then the maximal suffix extension of $L \cup L'$ is empty and $\mathsf{lcssum}(L \cup L') = (\mathsf{lcs}(R, R'), \varepsilon)$. Thus, we assume $R \overset{s}{\sqsubseteq} R' = \delta R$ from here on s.t. $\mathsf{lcs}(L \cup L') = R$. (The case $R' \overset{s}{\sqsubseteq} R = \delta R'$ is symmetric.) If $R \notin L$, then $E = \mathsf{lcsext}(L) = \varepsilon$ and thus $\mathsf{lcsext}(L \cup L') = \varepsilon = \mathsf{lcs}(E, \mathsf{lcs}(E', E'\delta)\delta)$.
So assume $R \in L$. If $R' \notin L'$, then for suitable $xR', yR' \in L'$, $\varepsilon = E' = \mathsf{lcsext}(L') = \mathsf{lcs}(x^\mho, y^\mho) = \mathsf{lcs}(x, y)$. Thus, $\mathsf{lcs}((z\delta)^\mho \mid z\delta R \in L') = \mathsf{lcs}(x\delta, y\delta) = \delta$ and hence

$$\mathsf{lcsext}(L \cup L') = \mathsf{lcs}(\mathsf{lcsext}(L), \delta) = \mathsf{lcs}(E, \mathsf{lcs}(E', E'\delta)\delta)$$

Thus also assume that $R' \in L'$. Then

$$\begin{array}{rcl} \mathsf{lcsext}(L \cup L') & = & \mathsf{lcs}(w^{m}, (z\delta)^{m} \mid z\delta R \in L', wR \in L) \\ & = & \mathsf{lcs}\left(\begin{array}{r} \mathsf{lcsext}(L), \\ \mathsf{lcs}(z\delta)^{m} \mid z\delta R \in L') \end{array} \right) \end{array}$$

From $R' = \delta R \in L'$ follows that $\mathsf{lcs}(z\delta)^{m} \mid z\delta R \in L') = \mathsf{lcs}(\mathsf{lcs}(\delta^{m}, (z\delta)^{m})) \mid z\delta R \in L')$. In Lemma 1 was shown that $\mathsf{lcs}(\delta^{m}, (z\delta)^{m}) = \mathsf{lcs}(\delta^{m}, z^{m})\delta$ and $\mathsf{lcs}(E', \delta^{m}) = \mathsf{lcs}(E', E'\delta)$. Thus

$$\mathsf{lcs}(\mathsf{lcs}(\delta^{m}, (z\delta)^{m}) \mid z\delta R \in L') = \mathsf{lcs}(\delta^{m}, \mathsf{lcsext}(L'))\delta = \mathsf{lcs}(\mathsf{lcsext}(L'), \mathsf{lcsext}(L')\delta)\delta$$

Therefore $\mathsf{lcsext}(L \cup L') = \mathsf{lcs}(E, \mathsf{lcs}(E', E'\delta)\delta)$. □

Lemma 9. *Let $L, L' \subseteq \Sigma^*$ with $\mathsf{lcssum}(L) = (R, E)$ and $\mathsf{lcssum}(L') = (R', E')$, then*

$$\mathsf{lcssum}(LL') = \begin{cases} (\top, \top) & \text{if } RR' = \top \\ (RR', E) & \text{if } RR' \overset{s}{\sqsubset} \top \wedge E' = \top \\ (\mathsf{lcs}(R, E')R', \varepsilon) & \text{if } RR' \overset{s}{\sqsubset} \top \wedge R \overset{s}{\not\sqsubseteq} E' \\ (RR', \mathsf{lcs}(E, \delta)) & \text{if } RR' \overset{s}{\sqsubset} \top \wedge E' = \delta R \end{cases}$$

Proof. If $R = \top$ or $R' = \top$, then $L = \emptyset$ or $L' = \emptyset$, respectively, such that $LL' = \emptyset$. Therefore we obtain $\mathsf{lcssum}(LL') = (\top, \top)$. Thus, we assume $R \neq \top \neq R'$, i.e. $L \neq \emptyset \neq L'$ from here on. If $E' = \top$, then $L' = \{R'\}$ such that $\mathsf{lcssum}(LL') = \mathsf{lcssum}(LR') = (RR', E)$.

Thus, we additionally assume that $E' \neq \top$ and therefore L' contains at least two words. Fix $xR', yR' \in L' \setminus \{R'\}$ such that

$$E' = \mathsf{lcsext}(z^{m} \mid zR' \in L') = \mathsf{lcs}(x^{m}, y^{m})$$

Consider

$$\mathsf{lcs}(LL') = \mathsf{lcs}(wRzR' \mid wR \in L, zR' \in L')$$

If $E' = \varepsilon$ and therefore $\mathsf{lcs}(x^{m}, y^{m}) = \varepsilon$, then

$$\begin{array}{rcl} \mathsf{lcs}(LL') & = & \mathsf{lcs}(wRzR' \mid wR \in L, zR' \in L') \\ & \overset{s}{\sqsubseteq} & \mathsf{lcs}(wRxR', wRyR' \mid wR \in L) \\ & = & \mathsf{lcs}(xR', yR') \\ & = & R' \end{array}$$

and

$$\begin{array}{rcl} \mathsf{lcsext}(LL') & = & \mathsf{lcs}((wRz)^{m} \mid wR \in L, zR' \in L') \\ & \overset{s}{\sqsubseteq} & \mathsf{lcs}((wRx)^{m}, (wRy)^{m} \mid wR \in L) \\ & \overset{s}{\sqsubseteq} & \mathsf{lcs}(x, y) \\ & = & \varepsilon \end{array}$$

Therefore, $\mathsf{lcssum}(LL') = (R', \varepsilon) = (\mathsf{lcs}(R, E')R', \varepsilon)$.
Now, assume that $E' \neq \varepsilon$ such that $R' \in L'$. Then

$$\begin{aligned} \mathsf{lcs}(LL') &= \mathsf{lcs}(wRzR' \mid wR \in L, zR' \in L') \\ &= \mathsf{lcs}\begin{pmatrix} R \\ \mathsf{lcs}(\mathsf{lcs}(wR, wRz) \mid wR \in L, zR' \in L') \end{pmatrix} R' \end{aligned}$$

Consider then

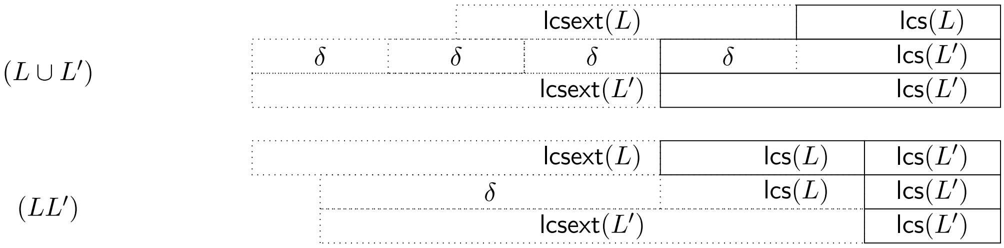
$$\mathsf{lcs}(R, E') = \mathsf{lcs}(R, \mathsf{lcsext}(L')) = \mathsf{lcs}(R, \mathsf{lcs}(z^{\mho} \mid zR' \in L')) = \mathsf{lcs}(R, Rx, Ry)$$

If $R \overset{s}{\not\sqsubseteq} E'$, then $R \overset{s}{\sqsupset} \mathsf{lcs}(R, E') = \mathsf{lcs}(R, Rx, Ry)$. Thus also $\mathsf{lcs}(LL') = \mathsf{lcs}(RL') = \mathsf{lcs}(R, E')R' \overset{s}{\sqsubset} RR'$. Therefore, $\mathsf{lcssum}(LL') = (\mathsf{lcs}(R, E')R', \varepsilon)$.
If $R \overset{s}{\sqsubseteq} E' = \delta R$, then $\mathsf{lcs}(LL') = RR'$ and

$$\mathsf{lcsext}(LL') = \mathsf{lcs}(z, \delta \mid zR \in L \setminus \{R\}) = \mathsf{lcs}(\mathsf{lcs}(z^{\mho} \mid zR \in L), \delta) = \mathsf{lcs}(E, \delta)$$

□

The computation of the lcs-summary as described in Lemmas 8 and 9 can be illustrated as follows.



Example 10. *Consider $L = \{a, baa\}$ and $L' = \{aa, baaa\}$ such that lcssum$(L) = (a, (ba)^\omega)$ and lcssum$(L') = (aa, (ba)^\omega)$. Then*

$$\begin{array}{lll} \mathit{lcs}(L \cup L') & = \mathit{lcs}\big(a, aa\big) & = a \\ \mathit{lcs}(LL') & = \mathit{lcs}\big(a, (ba)^\omega\big)aa & = aaa \\ \mathit{lcsext}(L \cup L') & = \mathit{lcs}\big((ba)^\omega, \mathit{lcs}((ba)^\omega, (ba)^\omega a)\big) & = a \\ \mathit{lcsext}(LL') & = \mathit{lcs}\big((ab)^\omega, a\big) & = \varepsilon \quad \textit{as } (ba)^\omega = (ab)^\omega a \end{array}$$

As both the lcs and the lcsext are determined by already two words (cf. Lemmas 2 and 6), it follows that every $L \subseteq \Sigma^*$ is $\approx_{\mathsf{lcs}}$-equivalent to some sublanguage $\mathsf{T}_{\mathsf{lcs}}(L) \subseteq L$ consisting of at most three words where the words xR, yR can be chosen arbitrarily up to the stated constraints (with $R = \mathsf{lcs}(L)$).

$$\mathsf{T}_{\mathsf{lcs}}(L) := \begin{cases} L & \text{if } |L| \leq 2 \\ \{R, xR, yR\} & \text{if } \{R, xR, yR\} \subseteq L \wedge \mathsf{lcsext}(L) = \mathsf{lcs}(x^{\mho}, y^{\mho}) \\ \{xR, yR\} & \text{if } R = \mathsf{lcs}(xR, yR) \wedge R \notin L \wedge \{xR, yR\} \subseteq L \end{cases}$$

Corollary 2. *Let $L \subseteq \Sigma^*$ be a language. Then L is $\approx_{\mathsf{lcs}}$-equivalent to some sublanguage $\mathsf{T}_{\mathsf{lcs}}(L) \subseteq L$ consisting of at most three words.*

In summary, Lemmas 8 and 9 yield that given L, L' with $L \approx_{\mathsf{lcs}} L'$, for all L''

$$L \cup L'' \approx_{\mathsf{lcs}} L' \cup L'' \qquad L''L \approx_{\mathsf{lcs}} L''L' \qquad LL'' \approx_{\mathsf{lcs}} L'L''$$

The equivalence classes given by $\approx_{\mathsf{lcs}}$ build a language semiring with the operations union and concatenation. Therefore lcssum is a homomorphism, i.e., for all L, L'

$$\mathsf{lcssum}(L \cup L') = \mathsf{lcssum}(L) \cup \mathsf{lcssum}(L') \qquad \mathsf{lcssum}(LL') = \mathsf{lcssum}(L)\,\mathsf{lcssum}(L')$$

This allows us to use the sublanguages $\mathsf{T}_{\mathsf{lcs}}(L)$ in Section 2.3 as polynomial size representation for the language L produced by a nonterminal.

2.3 Deciding Well-formedness

In this section we introduce the results leading to the polynomial time algorithm for deciding well-formedness of a language given by a context-free grammar. The decision procedure combines results of [TM07] and [LPS18]. Let G be a context-free grammar with $L = L(G) \subseteq \mathsf{B}^*$. In [TM07] it was shown how to decide in polynomial time whether L is *balanced*, i.e. $\rho(L) = \{\varepsilon\}$. The main observation is that for every sentential form $\alpha X \beta$ produced by G (with $\alpha, \beta \in \mathsf{B}^*$) and for any $\gamma \in L_X$ it has to hold that $\alpha\gamma\beta \overset{\rho}{=} \varepsilon$. This implies that all words in L_X are weakly well-formed ($\rho(L_X) \in \overline{\mathsf{A}}^*\mathsf{A}^*$) and have a constant height difference as the number of closing and opening letters (after reduction) is constrained by the context (α, β). This in turn implies that L_X has a canonical maximally reduced representative $\overline{r_X}\, v_X$ with $d_X = |r_X|$ and $\rho(r_X L_X) = \{v_X\}$. In case of well-formedness such a canonical representative does not need to exist anymore. While the question whether L is balanced enforces that the height difference has to be constant, in the case of well-formedness the height-difference is only bounded from below, but in general unbounded from above as the following example shows.

Example 11. *Let G be a context-free grammar with axiom X and rules*

$$X \to_G aaX\overline{a} \mid \varepsilon$$

Then $L = L(G) = \{a^{2k}\overline{a}^k \mid k \in \mathbb{N}_0\}$ with $\rho(L) = a^*$ and we obtain $r_X = \varepsilon$, $r_X L_X = L \approx_{\mathsf{lcs}} \{\varepsilon, a\}$ and therefore $\mathsf{lcssum}(\rho(r_X L_X)) = (\varepsilon, a^\omega)$.

Fixed-point iteration We use $\mathsf{T}_{\mathsf{lcs}}$ to compute a finite $\approx_{\mathsf{lcs}}$-equivalent representation $\mathsf{T}_X^{\leq h}$ of the *reduced* language generated by each nonterminal X of the given context-free grammar inductively for increasing derivation height h. In particular, we show that we only have to compute up to derivation height $4N+1$ (with N the number of nonterminals) in order to decide whether G is wf. In Lemma 11 we show that, if G is wf, then we have $\mathsf{T}_X^{\leq 4N+1} \approx_{\mathsf{lcs}} \mathsf{T}_X^{\leq 4N}$ for all nonterminals X of G – we have reached the fixed point. The complementary result is then shown in Lemma 12, i.e., if G is not wf, then we either cannot compute up to $\mathsf{T}_X^{\leq 4N+1}$ as we discover some word that is not wf, or we have $\mathsf{T}_X^{\leq 4N} \not\approx_{\mathsf{lcs}} \mathsf{T}_X^{\leq 4N+1}$ for at least one nonterminal X.

In the following, we assume that G is a context-free grammar over $\mathsf{B} = \mathsf{A} \cup \overline{\mathsf{A}}$ with nonterminals $\mathfrak{X}$. Set $N := |\mathfrak{X}|$. We further assume that G is nonnegative, and that we have computed for every nonterminal X of G a word $r_X \in \mathsf{A}^*$ represented as an SLP such that $|r_X| = d_X$ and $\overline{r_X} \in \mathsf{Prf}(\rho(L_X))$, cf. Lemma 4. Note that $\overline{r_X}$ is (after reduction) a longest word of closing brackets in $\rho(L_X)$; if G is wf, then r_X is unique. In order to decide whether G is wf we compute the languages $\rho(r_X L_X^{\leq h})$ modulo $\approx_{\mathsf{lcs}}$ for increasing derivation height h using fixed-point iteration. Assume inductively that

- $r_X L_X^{\leq h}$ is wf and
- we have computed $\mathsf{T}_X^{\leq h} := \mathsf{T}_{\mathsf{lcs}}(\rho(r_X L_X^{\leq h})) \approx_{\mathsf{lcs}} \rho(r_X L_X^{\leq h})$ for all $X \in \mathfrak{X}$ up to height h.

Then we can compute $\mathsf{T}_{\mathsf{lcs}}(\rho(r_X L_X^{\leq h+1}))$ for each nonterminal as follows:

$$\mathsf{T}_X^{\leq h+1} = \mathsf{T}_{\mathsf{lcs}}\Big(\rho\Big(\mathsf{T}_X^{\leq h} \cup \bigcup_{X \to_G Y} r_X \overline{r_Y}\ \mathsf{T}_Y^{\leq h} \cup \bigcup_{X \to_G YZ} r_X \overline{r_Y}\ \mathsf{T}_Y^{\leq h}\ \overline{r_Z}\ \mathsf{T}_Z^{\leq h}\Big)\Big) \tag{2.1}$$

Inductively, we can still observe that $\mathsf{T}_X^{\leq h+1} \approx_{\mathsf{lcs}} \rho(r_X L_X^{\leq h+1})$ as

$$\begin{aligned}
& \rho(r_X L_X^{\leq h+1}) \\
= \quad & \rho(r_X L_X^{\leq h}) \cup \textstyle\bigcup_{X \to_G Y} \rho(r_X \overline{r_Y}\ \ r_Y L_Y^{\leq h}) \cup \textstyle\bigcup_{X \to_G YZ} \rho(r_X \overline{r_Y}\ \ r_Y L_Y^{\leq h}\ \overline{r_Z}\ \ r_Z L_Z^{\leq h}) \\
\approx_{\mathsf{lcs}} \quad & \mathsf{T}_X^{\leq h} \cup \textstyle\bigcup_{X \to_G Y} \rho(r_X \overline{r_Y}\ \ \mathsf{T}_Y^{\leq h}) \cup \textstyle\bigcup_{X \to_G YZ} \rho(r_X \overline{r_Y}\ \ \mathsf{T}_Y^{\leq h}\ \overline{r_Z}\ \ \mathsf{T}_Z^{\leq h}) \\
\approx_{\mathsf{lcs}} \quad & \mathsf{T}_{\mathsf{lcs}}\Big(\rho\Big(\mathsf{T}_X^{\leq h} \cup \textstyle\bigcup_{X \to_G Y} r_X \overline{r_Y}\ \ \mathsf{T}_Y^{\leq h} \cup \textstyle\bigcup_{X \to_G YZ} r_X \overline{r_Y}\ \ \mathsf{T}_Y^{\leq h}\ \overline{r_Z}\ \ \mathsf{T}_Z^{\leq h}\Big)\Big) \\
= \quad & \mathsf{T}_X^{\leq h+1}
\end{aligned}$$

Note that each language $\mathsf{T}_X^{\leq h+1}$ consists of at most three words (cf. Corollary 2). Thus, each step of the fixed-point iteration can be computed in polynomial time.

Example 12. *Consider the following context-free grammar*

$$S = abbabaX \qquad X = baX\overline{aba}\,b\,aba \mid aXbb\,aba \mid \varepsilon$$

We obtain $r_S = \varepsilon$ and $r_X = a$. Thus, the grammar can be equivalently rewritten to nonterminals $[r_S S], [r_X X]$ producing the languages $r_S L_S$ and $r_X L_X$, respectively.

$$[r_S S] = abbab[r_X X] \qquad [r_X X] = ab[r_X X]\overline{aba}\,b\,aba \mid [r_X X]bb\,aba \mid a$$

Therefore, the following languages are computed in the fixed-point iteration

$$\begin{array}{rcl} r_S L_S{}^{\leq h+1} & = & r_S L_S^{\leq h} \cup abbab\, r_X L_X{}^{\leq h} \\ r_X L_X{}^{\leq h+1} & = & r_X L_X^{\leq h} \cup ab\, r_X L_X{}^{\leq h}\, \overline{aba}\, baba \cup r_X L_X{}^{\leq h}\, bbaba \cup a \end{array}$$

As $r_S L_S{}^{\leq h+1}$ depends on $r_X L_X{}^{\leq h}$ and $r_X L_X{}^{\leq h+1}$ only on $r_X L_X{}^{\leq h}$ we can calculate $r_X L_X^h$ for increasing h until convergence. Afterwards $r_S L_S$ is computed.

$$\begin{array}{rcrl}
r_X L_X^0 & = & \emptyset & \\
\rho([r_X L_X]^0) & = & \emptyset & \quad \mathsf{lcs} = \top, \mathsf{lcsext} = \top \\
\rho([r_X L_X]^0) & \approx_{\mathsf{lcs}} & \emptyset & \\
r_X L_X^{\leq 1} & = & a & \\
\rho(r_X L_X{}^{\leq 1}) & = & a & \quad \mathsf{lcs} = a, \mathsf{lcsext} = \top \\
\rho(r_X L_X{}^{\leq 1}) & \approx_{\mathsf{lcs}} & a & \\
r_X L_X{}^{\leq 2} & = & a & \\
 & \cup & ab\, a\, \overline{aba}\, baba & \\
 & \cup & a\, bbaba & \\
\rho(r_X L_X{}^{\leq 2}) & = & a & \\
 & \cup & baba & \\
 & \cup & abbaba & \quad \mathsf{lcs} = a, \mathsf{lcsext} = \mathsf{lcs}(bab^{\rotatebox[origin=c]{180}{ω}}, abbab^{\rotatebox[origin=c]{180}{ω}}) \\
 & & & \quad \phantom{\mathsf{lcs} = a, \mathsf{lcsext}} = babbab \\
\rho(r_X L_X{}^{\leq 2}) & \approx_{\mathsf{lcs}} & \rho(r_X L_X{}^{\leq 2}) & \\
r_X L_X{}^{\leq 3} & = & a & \\
 & \cup & ab\, a\, \overline{aba}\, baba & \\
 & \cup & a\, bbaba & \\
 & \cup & ab\, baba\, \overline{aba}\, baba & \\
 & \cup & ab\, abbaba\, \overline{aba}\, baba & \\
 & \cup & baba\, bbaba & \\
 & \cup & abbaba\, bbaba & \\
\rho(r_X L_X{}^{\leq 3}) & = & a & \\
 & \cup & baba & \\
 & \cup & abbaba & \\
 & \cup & abbbaba & \\
 & \cup & ababbbaba & \\
 & \cup & bababbaba & \\
 & \cup & abbababbaba & \quad \mathsf{lcs} = a, \mathsf{lcsext} = \mathsf{lcs}(bab^{\rotatebox[origin=c]{180}{ω}}, abbbab^{\rotatebox[origin=c]{180}{ω}})
\end{array}$$

$$= bbab$$

$$\begin{array}{lcrl}
\rho(r_X {L_X}^{\leq 3}) & \approx_{\mathsf{lcs}} & a & \\
 & \cup & baba & \\
 & \cup & abbbaba & \\
\rho(r_X L_X^{\leq 4}) & \approx_{\mathsf{lcs}} & a & \\
 & \cup & ab\, a\, \overline{aba}\, baba & \\
 & \cup & ab\, baba\, \overline{aba}\, baba & \\
 & \cup & ab\, abbbaba\, \overline{aba}\, baba & \\
 & \cup & a\, bbaba & \\
 & \cup & baba\, bbaba & \\
 & \cup & abbbaba\, bbaba & \\
\rho(r_X L_X^{\leq 4}) & \approx_{\mathsf{lcs}} & a & \\
 & \cup & baba & \\
 & \cup & abbbaba & \\
 & \cup & ababbbbaba & \\
 & \cup & abbaba & \\
 & \cup & bababbaba & \\
 & \cup & abbbababbaba & \mathsf{lcs} = a, \mathsf{lcsext} = \mathsf{lcs}(bab^{\frown}, abbbab^{\frown}) \\
 & & & = bbab \\
\rho(r_X L_X^{\leq 4}) & \approx_{\mathsf{lcs}} & a & \\
 & \cup & baba & \\
 & \cup & abbbaba &
\end{array}$$

Thus, $\mathsf{lcssum}(\rho(r_X L_X^{\leq h}))$ *converges after three iterations. I.e., the above fixed-point iteration yields* $\rho(r_X L_X) \approx_{\mathsf{lcs}} \{a, baba, abbbaba\}$ *with* $\mathsf{lcs}(\rho(r_X L_X)) = a$ *and* $\mathsf{lcsext}(\rho(r_X L_X)) = bbab$*. For* $\rho(r_S L_S)$ *we therefore obtain*

$$\begin{array}{lcrl}
\rho(r_S L_S) & \approx_{\mathsf{lcs}} & abbab\, a & \\
 & \cup & abbab\, baba & \\
 & \cup & abbababbbaba & \mathsf{lcs} = bbaba, \mathsf{lcsext} = \mathsf{lcs}(a^{\frown}, abbabab^{\frown}) = \varepsilon \\
\rho(r_S L_S) & \approx_{\mathsf{lcs}} & abbaba & \\
 & \cup & abbababbbaba &
\end{array}$$

We show in the remaining part of this section that the above introduced fixed-point iteration (2.1) can be used to decide well-formedness of context-free grammars. First, we consider well-formed languages and show that the fixed-point iteration terminates after at most $4N$ iterations with N the number of nonterminals. Second, we show that in the case that the language produced by a context-free grammar is *not* well-formed we either find a counterexample in $4N$ iterations or the representations $\mathsf{T}_X^{\leq h}$ of at least one nonterminal of the context-free grammar does not converge within $4N$ iterations. Again, N is the number of nonterminals of the context-free grammar.

Convergence for well-formed CFGs Lemma 11 states the first result, i.e., if G is wf then the representations $\mathsf{T}_X^{\leq h}$ have converged at the latest for $h = 4N$ modulo $\approx_{\mathsf{lcs}}$. The basic idea underlying the proof of Lemma 11 is similar to [LPS18]. We show that from every derivation tree of height at least $4N+1$ we can construct a derivation tree of height at most $4N$ such that both trees carry the same information w.r.t. the lcs and lcsext (after reduction). However, Lemma 11 extends the central result of [LPS18]. There, a polynomial time algorithm to compute the longest common prefix (lcp) of a language given by a context-free grammar was presented. But the languages were just interpreted over a simple alphabet without taking any reduction into account. In case of the computation of the longest common prefix it therefore sufficed to show that the lcp has already converged at derivation height $4N$. Here, we need to show that also the maximal extension lcsext of the lcs has converged. This then guarantees that, if G is not wf, but the minimal derivation height h_0 for which we find a counterexample is higher than $4N$, then we will detect this as $\mathsf{lcssum}(r_X L_X^{\leq 4N})$ has not converged (modulo $\approx_{\mathsf{lcs}}$) for at least one nonterminal X. [1]
We sketch the proof idea of Lemma 11. In order to show that we only need to compute $\mathsf{T}^{\rho}_{\mathsf{lcs}}(r_X L_X^{\leq 4N})$ if G is wf, we have to show that any derivation tree w.r.t. $r_X L_X$ of height at least $4N+1$ has no influence on $\mathsf{lcssum}(\rho(r_X L_X))$. More precisely, we assume that a witness κ for the reduced longest common suffix with respect to a shortest word (after reduction) has a derivation tree of height at least $4N+1$. Any such derivation tree induces a "simple" well-formed linear grammar of the form

$$S \to \alpha X \beta \quad X \to \sigma_1 X \tau_1 \mid \sigma_2 X \tau_2 \mid \sigma_3 X \tau_3 \mid \gamma \qquad (\alpha, \beta, \gamma, \sigma_i, \tau_i \in \mathsf{B}^*)$$

The language L' generated by this grammar can be described by the rational expression

$$(\alpha,\beta)[(\sigma_1,\tau_1) + (\sigma_2,\tau_2) + (\sigma_3,\tau_3)]^*\gamma := \sum_{i_1 \ldots i_l \in \{1,2,3\}^*} \alpha \sigma_{i_1} \ldots \sigma_{i_l} \gamma \tau_{i_l} \ldots \tau_{i_1} \beta$$

[1] As every context-free grammar is a special case of well-formed $\mathsf{LTW_B}$ we can use Lemma 29 to remove closing brackets for all nonterminals that produce an ultimately periodic language after reduction. But unfortunately, we currently do not know if we can transform in polynomial time a well-formed context-free grammar G over B into a context-free grammar G' over A s.t. $\rho(L(G)) = L(G')$ (and further s.t. derivations are in bijection). We have not been able to make direct use of the results of [LPS18]: When transforming a well-formed simple linear grammar over B, our proofs in Section 2.4 indicate that we might need to introduce auxiliary nonterminals in order to partition the language w.r.t. the pumping trees resp. contexts that are used at least once s.t. the resulting grammar might be exponentially larger.

L' is a sublanguage of the language L generated by G and has the same reduced longest common suffix as L. We first show that we can remove the negative letters $\overline{a} \in \overline{\mathsf{A}}$ occurring in α, γ, σ_i and that each τ_i is either of the form $\tau_i \overset{\rho}{=} \overline{r_i}\,\overline{t_i}\,r_i$ with $t_i \neq \varepsilon$ or $\tau_i \overset{\rho}{=} \overline{r_i}\,t_i r_i$ (with $r_i, t_i \in \mathsf{A}^*$).

Lemma 10. *Let $L = (\alpha, \beta)[(\sigma_1, \tau_1) + \ldots + (\sigma_n, \tau_n)]^*\gamma$ be a wf language with $\alpha, \beta, \sigma_i, \gamma \in \mathsf{B}^*$ and $n \geq 0$. Then there are $u, s_i, w, r_i, t_i \in \mathsf{A}^*$ such that*

$$L \overset{\rho}{=} (u, \beta)[(s_1, \tau_1) + \ldots + (s_n, \tau_n)]^* w \qquad \textit{and}$$

$$\rho(\tau_i) = \overline{r_i}\,t_i r_i \quad \textit{or} \quad \rho(\tau_i) = \overline{r_i}\,\overline{t_i}\,r_i, t_i \neq \varepsilon.$$

Moreover, for any $i_1, \ldots, i_l \in \{1, \ldots, n\}$, it holds that $\alpha\sigma_{i_1} \ldots \sigma_{i_l}\gamma\tau_{i_l} \ldots \tau_{i_1}\beta \overset{\rho}{=} us_{i_1} \ldots s_{i_l} w\tau_{i_l} \ldots \tau_{i_1}\beta$.

Proof. As α has to be wf, simply set $u = \rho(\alpha) \in \mathsf{A}^*$. Let $\rho(\sigma_i) = \overline{x_i}\,y_i$ for any $i \in \{1, \ldots, n\}$. Then $u\overline{x_i}$ has to be wf for all $i \in \{1, \ldots, n\}$, i.e. we have $u \overset{\rho}{=} u\overline{x_i}\,x_i$ for all $i \in \{1, \ldots, n\}$. As $u\sigma_i\sigma_i \overset{\rho}{=} u\overline{x_i}\,y_i\overline{x_i}\,y_i$ has to be wf, we have $y_i = s_i x_i$. Pick $J \in \{1, \ldots, n\}$ such that x_J is a longest word of $\{x_1, \ldots, x_n\}$. Then $u\sigma_i\sigma_J \overset{\rho}{=} u\overline{x_J}\,x_J\overline{x_i}\,s_i x_i\overline{x_J}\,s_J x_J$ has to be wf, i.e., $(x_J\overline{x_i}\,)s_i\overline{x_J\overline{x_i}}$ is wf for all $i \in \{1, \ldots, n\}$. Thus there exist $\hat{s}_i$ such that $\rho(x_J\overline{x_i}\,)s_i = \hat{s}_i\,\rho(x_J\overline{x_i}\,)$, resp. $x_J\overline{x_i}\,s_i \overset{\rho}{=} \hat{s}_i x_J\overline{x_i}$ as $\rho(x_J\overline{x_i}\,) \in \mathsf{A}^*$. Therefore,

$$\begin{aligned}
u\sigma_{i_1} \ldots \sigma_{i_l} &\overset{\rho}{=} u\overline{x_J}\,x_J\overline{x_{i_1}}\,s_{i_1}x_{i_1} \ldots \overline{x_{i_l}}\,s_{i_l}x_{i_l} \\
&\overset{\rho}{=} u\overline{x_J}\,\hat{s}_{i_1}\overline{x_J}\,\overline{x_{i_1}}\,s_{i_2}x_{i_2} \ldots \overline{x_{i_l}}\,s_{i_l}x_{i_l} \\
&\overset{\rho}{=} \ldots \\
&\overset{\rho}{=} u\overline{x_J}\,\hat{s}_{i_1} \ldots \hat{s}_{i_l}x_J
\end{aligned}$$

for all $i_1, \ldots, i_l \in \{1, \ldots, n\}$. Thus, set $u = \rho(u\overline{x_J}\,)$, $s_i = \hat{s}_i$ and $\gamma = \rho(x_J\gamma)$.
Now, we analogously show that $\rho(\gamma) = w \in \mathsf{A}^*$. If $\rho(\gamma) = \overline{x}\,w$, then $u = u'x$ resp. $u \overset{\rho}{=} u\overline{x}\,x$ and thus $us_i\gamma \overset{\rho}{=} u\overline{x}\,xs_i\overline{x}\,w$. So $xs_i\overline{x}$ is wf for all $i \in \{1, \ldots, n\}$. Hence, we find $\hat{s}_i$ with $xs_i = \hat{s}_i x$ such that $us_{i_1} \ldots s_{i_l}\gamma \overset{\rho}{=} u\overline{x}\,xs_{i_1} \ldots s_{i_n}\overline{x}\,w \overset{\rho}{=} u\overline{x}\,\hat{s}_{i_1} \ldots \hat{s}_{i_l}x\overline{x}\,w$ for $i_1, \ldots, i_l \in \{1, \ldots, n\}$. Thus, set $s_i = \hat{s}_i$ and $u = \rho(u\overline{x}\,)$.
It is left to show that for $i \in \{1, \ldots, n\}$ we have either $\rho(\tau_i) = \overline{r_i}\,t_i r_i$ or $\rho(\tau_i) = \overline{r_i}\,\overline{t_i}\,r_i$ with $t_i \neq \varepsilon$ in the last case. Let $\rho(\tau_i) = \overline{x_i}\,y_i$ for $i \in \{1, \ldots, n\}$. Then $L_i := (u, \beta)(s_i, \tau_i)^*w$ has to be wf for any $i \in \{1, \ldots, n\}$ as $L_i \subseteq L$ and L is wf by assumption. Thus $\tau_i^2 \overset{\rho}{=} \overline{x_i}\,y_i\overline{x_i}\,y_i$ has to be wwf. If $|y_i| \geq |x_i|$, then $y_i\overline{x_i}$ has to be wf, i.e. $x_i \overset{s}{\sqsubseteq} y_i$. Setting $r_i = x_i$ and $t_i = \rho(y_i\overline{x_i}\,)$, we have

$$\tau_i \overset{\rho}{=} \overline{x_i}\,y_i = \overline{x_i}\,\rho(y_i\overline{x_i}\,)x_i = \overline{r_i}\,t_i r_i$$

Otherwise $|y_i| < |x_i|$ and $x_i\overline{y_i}$ is wf with $y_i \overset{s}{\sqsubset} x_i$. Then set $r_i = y_i$ and $t_i = \rho(x_i\overline{y_i}\,) \neq \varepsilon$ such that

$$\tau_i \overset{\rho}{=} \overline{x_i}\, y_i = \overline{\rho(x_i\overline{y_i}\,)y_i}\, y_i = \overline{t_i r_i}\, r_i = \overline{r_i}\,\overline{t_i}\, r_i$$

□

Therefore, L' is of the form $(u,\beta)[(s_1,\tau_1)+(s_2,\tau_2)+(s_3,\tau_3)]^*w$ with $u, s_i, w \in \mathsf{A}^*$ and $\beta, \tau_i \in \mathsf{B}^*$. Our goal is to show that the reduced longest common suffix of L' is already determined by $uw\beta$ and a word $(u,\beta)(s_i,\tau_i)w$ or $(u,\beta)(s_i,\tau_i)(s_j,\tau_j)w$ (for $i \neq j$) as we could then prune one pumping tree from the derivation tree of the witness κ leading inductively to a witness with height at most $4N$. Therefore we use the central combinatorical observation from Section 2.4 (Lemmas 15, 16 and 18)[2] that states that for any well-formed language $\mathcal{L} \subseteq \mathsf{B}^*$ of the form

$$\mathcal{L} = (\alpha,\beta)[(\mu_1,\nu_1)+(\mu_2,\nu_2)]^*\gamma := \{\alpha\mu_{i_1}\dots\mu_{i_l}\gamma\nu_{i_l}\dots\nu_{i_1}\beta \mid i_1\dots i_l \in \{1,2\}^*\}$$

we have that its *longest common suffix after reduction* $\mathsf{lcs}^\rho(\mathcal{L}) := \mathsf{lcs}(\rho(\mathcal{L}))$ is determined by the reduced longest common suffix of $\alpha\gamma\beta$ and either $(\alpha,\beta)(\mu_i,\nu_i)\gamma = \alpha\mu_i\gamma\nu_i\beta$ or $(\alpha,\beta)(\mu_i,\nu_i)(\mu_j,\nu_j)\gamma = \alpha\mu_i\mu_j\gamma\nu_j\nu_i\beta$ for some $i \in \{1,2\}$ but *arbitrary* $j \in \{1,2\}$ in the latter case I.e. we need at most two copies of the pumping trees where we can choose to either use the same pumping tree at most twice or two distinct pumping trees at most once.

Lemma 11. *Let G be a context-free grammar with N nonterminals and $L(G)$ be well-formed. For every nonterminal X let $r_X \in \mathsf{A}^*$ s.t. $|r_X| = d_X$ and $r_X L_X$ wf. Then $\rho(r_X L_X) \approx_{\mathsf{lcs}} \rho(r_X L_X^{\leq 4N})$.*

Proof. Let S be the axiom of G. W.l.o.g. G is reduced to the nonterminals which are reachable from S and which are productive. Additionally assume that $L(G)$ contains at least two words; otherwise $\mathsf{lcsext}(\rho(r_X L_X)) = \top$ and the claim of the lemma follows directly. As for every nonterminal X, $r_X L_X$ is wf, we have for any $\zeta \in L_X$ that $\rho(\zeta) = \overline{u}\, v$, $u, v \in \mathsf{A}^*$ with $u \overset{s}{\sqsubseteq} r_X = r'_X u$ s.t. $r_X\zeta \overset{\rho}{=} r'_X v$ and thus

$$|\rho(r_X\zeta)| = |r'_X v| = |r_X v| - |u| = \Delta(r_X\zeta)$$

Let $\kappa_0 \in L_X$ be a shortest word after reduction, i.e., $\Delta(r_X\kappa_0) = \min\{\Delta(r_X\zeta) \mid \zeta \in r_X L_X\}$. Let $\kappa_1 \in L_X$ be a second shortest word after reduction, i.e., $\Delta(r_X\kappa_1) = \min\{\Delta(r_X\zeta) \mid \zeta \in r_X L_X, \Delta(\zeta) > \Delta(\kappa_0)\}$. We show that then w.l.o.g. $\kappa_0 \in L_X^{\leq N}$ and $\kappa_1 \in L_X^{\leq 2N}$.

[2] This observation strengthens the combinatorical results in [LPS18] and also allows to greatly simplify the original proof of convergence given there.

Claim $\kappa_0 \in L_X^{\leq N}$ and $\kappa_1 \in L_X^{\leq 2N}$. For any $S \to_G^* \alpha X \beta$ and $X \to_G^* \gamma$ we need to have that for all $k \geq 0$,

$$\Delta((\alpha,\beta)(\sigma,\tau)^k\gamma) = \Delta(\alpha\gamma\beta) + k\Delta(\sigma\tau) \geq 0$$

and thus $\Delta(\tau) \geq -\Delta(\sigma)$. Any word $\zeta \in L_X \setminus L_X^{\leq N}$ has a derivation tree with a path from its root to some leaf along which at least $N+1$ nontermimals occur, i.e. along at least one nonterminal occurs twice which gives rise to a factorization of the form

$$\zeta = (\alpha,\beta)(\sigma,\tau)\gamma$$

such that

$$|\rho(r_X\zeta)| = \Delta(r_X\zeta) = \Delta(r_X\alpha\gamma\beta) + \Delta(\sigma\tau) \geq \Delta(r_X\alpha\gamma\beta) = |\rho(r_X\alpha\beta\gamma)|$$

Removing the pumping tree that gives rise to the factor (σ,τ) thus leads to a word $\alpha\gamma\beta$ that is shorter than ζ before reduction, and at most as long as ζ after reduction. Hence, $r_X L_X^{\leq N}$ already contains all shortest words after reduction, i.e.

$$\min\{\Delta(r_X\zeta) \mid \zeta \in r_X L_X\} = \min\{\Delta(r_X\zeta) \mid \zeta \in r_X L_X^{\leq N}\}$$

and thus w.l.o.g. $\kappa_0 \in L_X^{\leq N}$.
Next, we consider κ_1. Any path that consists of at least $2N+1$ nonterminals contains at least one terminal three times which gives rise to a factorization of the form

$$\kappa_1 = (\alpha,\beta)(\sigma_1,\tau_1)(\sigma_2,\tau_2)\gamma$$

If $\Delta(\sigma_1\tau_1) = \Delta(\sigma_2\tau_2) = 0$, we can remove both pumping trees and obtain $\alpha\beta\gamma \in L_X^{\leq N}$ with $|\rho(\alpha\gamma\beta)| = |\kappa_1|$. So assume $\Delta(\sigma_i\tau_i) > 0$ for $i = 1$ or $i = 2$. Removing (σ_i,τ_i) leads to $(\alpha,\beta)(\sigma_j,\tau_j)\gamma$ $(j \neq i)$ with

$$\Delta(r_X\kappa_1) > \Delta((r_X\alpha,\beta)(\sigma_j,\tau_j)\gamma) \geq \Delta(r_X\kappa_0)$$

As κ_1 is a second shortest word after reduction, $\Delta((\alpha,\beta)(\sigma_j,\tau_j)\gamma) = \Delta(\kappa_0)$ has to hold and thus $\Delta(\sigma_j\tau_j) = 0$. Therefore, we can remove (σ_j,τ_j) to obtain the word $(\alpha,\beta)(\sigma_i,\tau_i)\gamma \in L_X^{\leq 2N}$ as a second shortest word after reduction as $\Delta(r_X\kappa_1) = \Delta((r_X\alpha,\beta)(\sigma_i,\tau_i)\gamma)$. Thus, the claim follows.
Let $R := \mathsf{lcs}^\rho(r_X L_X)$. If $\mathsf{lcsext}(\rho(r_X L_X)) \in \mathsf{A}^\varpi$, then $\rho(\kappa_0) = R$ and $\rho(\kappa_1) = xR$ for some $x \in \mathsf{A}^+$. Then for any $\zeta \in L_X$ we have $\rho(r_X\zeta) = yR$ with $x^\varpi = y^\varpi$, i.e. $xy = yx$. Thus, $\mathsf{T}_{\mathsf{lcs}}(\rho(r_X L_X))$ is in this case given by $\{\rho(r_X\kappa_0), \rho(r_X\kappa_1)\} \subseteq r_X L_X^{\leq 2N}$.
Therefore, assume from here on that the lcs^ρ *of* $r_X L_X$ *can at most be finitely extended, i.e.,* $\mathsf{lcsext}(\rho(r_X L_X)) \in A^*$. Moreover, we can assume that $\rho(r_X L_X)$ contains at least three distinct words; otherwise the lemma follows directly with the

shown bounds on the height of the derivation trees of κ_0 and κ_1. We distinguish the two cases whether $R = \mathsf{lcs}^{\rho}(r_X L_X)$ is a strict suffix of every word in $r_X L_X$, in particular $R \overset{s}{\sqsubset} \rho(r_X\kappa_0)$, or if R is a, and thus the shortest word after reduction, in particular $R = \rho(r_X\kappa_0)$.

- If $R = \mathsf{lcs}^{\rho}(r_X L_X) \overset{s}{\sqsubset} \rho(r_X\kappa_0)$, there is some witness $\kappa \in L_X$ s.t.

  $$R = \mathsf{lcs}(\rho(r_X\kappa_0), \rho(r_X\kappa)) \overset{s}{\sqsubset} r_X\kappa_0$$

  In particular, we have that $\rho(r_X\kappa_0) = \dots aR$ and $\rho(r_X\kappa) = \dots bR$ for two distinct opening parenthesis $a, b \in \mathsf{A}$ ($a \neq b$).

- If $R = \mathsf{lcs}^{\rho}(r_X L_X) = \rho(r_X\kappa_0)$, then recall Lemma 7. The maximal extension E of the lcs R is given by

  $$E = \mathsf{lcsext}^{\rho}(L) = \mathsf{lcs}(\rho(r_X\zeta\overline{R}\,)^{\reflectbox{m}} \mid \zeta \in L_X, \rho(r_X\zeta) \neq R)$$

  As E is assumed to be finite, $\rho(r_X L_X)$ has to contain at least two other reduced words, both longer than $\rho(r_X\kappa_0)$. In particular, there has to be a second shortest word after reduction κ_1 s.t. we find a witness κ for E w.r.t. κ_1, i.e.,

  $$E = \mathsf{lcs}(\rho(r_X\zeta\overline{R}\,)^{\reflectbox{m}} \mid \zeta \in L_X, \rho(r_X\zeta) \neq R) = \mathsf{lcs}(\rho(r_X\kappa_1\overline{R}\,)^{\reflectbox{m}}, \rho(r_X\kappa\overline{R}\,)^{\reflectbox{m}})$$

  and

  $$ER = \mathsf{lcs}^{\rho}(x^m r_X L_X) \overset{s}{\sqsubset} \rho(x^m r_X\kappa_0) = x^m R = x^{m-1}\,\rho(r_X\kappa_1)$$

  Thus, we can reduce this case to the case where R is a strict suffix of any word in $r_X L_X$ by extending r_X to $x^m r_X$.

  Note that then any witness for ER w.r.t. $x^m r_X\kappa_0$ is also a witness w.r.t. $x^m r_X\kappa_1$ and vice versa.

Assume thus w.l.o.g. that $R = \mathsf{lcs}^{\rho}(r_X L_X) \overset{s}{\sqsubset} \rho(r_X\kappa_0)$ *from here on.* Choose κ in L_X such that

1. $\mathsf{lcs}(\rho(r_X\kappa_0), \rho(r_X\kappa)) = \mathsf{lcs}^{\rho}(r_X L_X)$.
2. $|\kappa|$ is minimal w.r.t. to all words in $r_X L_X$ satisfying **1.**,
3. $|\rho(r_X\kappa)|$ is minimal w.r.t. to all words in $r_X L_X$ satisfying **2.**.

W.l.o.g. $\rho(r_X\kappa_0) = \dots aR$ and $\rho(r_X\kappa) = \dots bR$ with $a \neq b$ and $a, b \in \mathsf{A}$. Note that there is a unique factorization $r_X\kappa = \zeta b\xi$ s.t. both ζ and ξ are wf and $\rho(\xi) = R$. Therefore, for every prefix (before reduction) ψ of $r_X\kappa$ we can interpret $\Delta(\psi)$ as the height of the last letter of ψ. Then the letter b in $\rho(r_X\kappa) = \ldots bR$ is the last letter in $r_X\kappa$ of height $\Delta(r_X\kappa) - |R|$ and is, thus, uniquely identified. This specific letter b splits $r_X\kappa$ into $r_X\kappa = \zeta b\xi$; as this b is the last letter in κ on height $\Delta(\kappa) - |R|$, ξ has to be wf with $\rho(\xi) = R$; as $r_X\kappa$ is wf and ζ is a prefix thereof, trivially also ζ is wf.

Assume every derivation tree of κ contains a path to a letter within $b\xi$ along which some nontermimal A occurs at least 4 times (see Fig. 2.2). This gives rise to a factorization

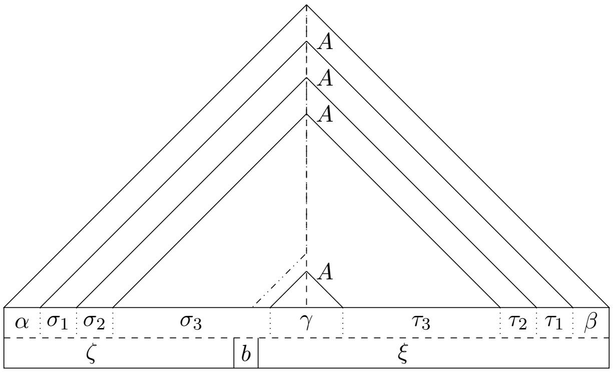


Figure 2.2: We assume that the given derivation tree of the witness κ contains a path (drawn as dashed line) which (i) leads to one of the letters within $b\xi$ and (ii) consists of at least $3N+1$ nonterminals so that by the pigeonhole principle at least one nontermimal A occurs at least 4 times; specifically, consider precisely the first $3N+1$ nonterminals along such path and let A be the nonterminal that occurs both at least 4 times within this fragment and also occurs the earliest. W.r.t. the nonterminal A we factorize the witness as $\kappa = (\alpha, \beta)(\sigma_1, \tau_1)(\sigma_2, \tau_2)(\sigma_3, \tau_3)\gamma \stackrel{\rho}{=} \ldots b\, \mathsf{lcs}^\rho(L)$.

$$\kappa = (\alpha, \beta)(\sigma_1, \tau_1)(\sigma_2, \tau_2)(\sigma_3, \tau_3)\gamma$$

Any word

$$\eta \in \{(\alpha, \beta)(\sigma_1, \tau_1)^{k_1}(\sigma_2, \tau_2)^{k_2}(\sigma_3, \tau_3)^{k_3}\gamma \mid k_i \in \{0,1\}, k_1 + k_2 + k_3 < 3\}$$

is shorter (before reduction) than κ, hence cannot be a witness w.r.t. κ_0, i.e., $aR \overset{s}{\sqsubseteq} \mathsf{lcs}^\rho(r_X\kappa, r_X\eta)$. Let

$$L = (r_X\alpha, \beta)[(\sigma_1, \tau_1) + \ldots + (\sigma_3, \tau_3)]^*\gamma$$

Then L is wf with $R = \mathsf{lcs}^\rho(r_X L_X) = \mathsf{lcs}^\rho(L)$ as $L \subseteq r_X L_X$ contains both $r_X\kappa$ and $r_X\alpha\gamma\beta$ with the latter not a witness w.r.t. $r_X\kappa_0$ s.t. $\mathsf{lcs}^\rho(r_X\kappa, r_X\alpha\gamma\beta) = R$.

Our goal is to show that already $r_X\alpha\sigma_i\gamma\tau_i\beta$ or $r_X\alpha\sigma_i\sigma_j\gamma\tau_j\tau_i\beta$ for some $i \neq j$ is a witness w.r.t. $r_X\alpha\gamma\beta$. We make the following observations.

- $r_X\alpha$ has to be wf, all other factors $\beta, \gamma, \sigma_i, \tau_i$ have to be wwf.
- As noted already at the beginning, we have both $\Delta(\sigma_i) \geq 0$ and $\Delta(\sigma_i\tau_i) \geq 0$.
- By choice of the path used for the factorization, we have $\rho(\tau_3\tau_2\tau_1\beta) = \overline{x}\,y$ with $y \overset{s}{\sqsubseteq} R = \mathsf{lcs}^\rho(L) = \mathsf{lcs}^\rho(r_X L_X)$.

Lemma 10 reduces the factors $\alpha, \beta, \gamma, \sigma_i$ to words in A^* such that we assume $L = (u,\beta)[(s_1,\tau_1) + \ldots + (s_3,\tau_3)]^*w$ and $\rho(\tau_i) = \overline{r_i}\,t_i r_i$ or $\rho(\tau_i) = \overline{r_i}\,\overline{t_i}\,r_i$, $t_i \neq \varepsilon$. W.l.o.g. we may further assume that $\beta = \varepsilon$ as this amounts to changing $R = \mathsf{lcs}^\rho(L)$ to $R := \rho(\mathsf{lcs}^\rho(L)\overline{\beta}\,)$; we therefore set $\kappa = (\alpha,\varepsilon)(\sigma_1,\tau_1)(\sigma_2,\tau_2)(\sigma_3,\tau_3)\gamma$. Assume for some $i \in \{1,2,3\}$ that $\rho(\tau_i) = \overline{r_i}\,\overline{t_i}\,r_i$ with $t_i \neq \varepsilon$. As we show in Lemmas 15 and 16 we always have for $j \neq i$ and any $i_1, \ldots, i_l \in \{i, j\}$,

- $us_{i_1}\ldots s_{i_l}s_j w\tau_j \overset{\rho}{=} us_{i_1}\ldots s_{i_l}p^{k_j}w$ and
- $us_{i_1}\ldots s_{i_l}s_j w\overline{r_i}\,\overline{t_i}\,r_i \overset{\rho}{=} us_{i_1}\ldots s_{i_l}p^{k_i}w$.

Hence $L \overset{\rho}{=} u(p^{k_1}+p^{k_2}+p^{k_3})^*w$ and thus $\mathsf{lcs}^\rho(L) = \mathsf{lcs}^\rho(uw, us_1w\tau_1, us_2w\tau_2, us_3w\tau_3)$ and $\mathsf{lcsext}^\rho(L) = \mathsf{lcsext}^\rho(\{uw, us_1w\tau_1, us_2w\tau_2, us_3w\tau_3\})$, therefore $\rho(r_X L_X) \approx_{\mathsf{lcs}} \rho(r_X L_X^{\leq 2N})$.

So it remains the case that for all $i \in \{1,2,3\}$ we have $\tau_i = \overline{r_i}\,t_i r_i$. Hence

$$L = (u,\varepsilon)[(s_1,\overline{r_1}\,t_1r_1) + (s_2,\overline{r_2}\,t_2r_2) + (s_3,\overline{r_3}\,t_3r_3)]^*w$$

As before $r_i\overline{r_j}$ has to be wwf for any $i,j \in \{1,2,3\}$. Let $\{i_1,i_2,i_3\} = \{1,2,3\}$ such that $r_{i_3} \overset{s}{\sqsubseteq} r_{i_2} \overset{s}{\sqsubseteq} r_{i_1}$. Note that $\rho(t_3r_3\overline{r_2}\,t_2r_2\overline{r_1}\,t_1r_1) = \overline{x}\,y$ with $y \overset{s}{\sqsubseteq} \mathsf{lcs}^\rho(L) = R$ as $t_3r_3\overline{r_2}\,t_2r_2\overline{r_1}\,t_1r_1$ is a suffix of κ and κ is a shortest witness by assumption. Hence $|R| \geq |y| = \Delta(t_3r_3\overline{r_2}\,t_2r_2\overline{r_1}\,t_1r_1) + |x| = |t_1t_2t_3| + |r_3| + |x|$. We show that $|R| \geq |y| \geq |t_ir_i|$ for all $i \in \{1,2,3\}$.

- If $|r_1| \leq |t_2r_2| \wedge |r_2| \leq |t_3r_3|$ then $|y| = |t_1t_2t_3r_3| \geq |t_1t_2r_2| \geq |t_1r_1|$.
- If $|r_1| > |t_2r_2| \wedge |r_2| + |r_1| - |t_2r_2| \leq |t_3r_3|$ then $|t_2r_2| < |r_1| \wedge |r_1| \leq |t_2t_3r_3|$ and $|y| = |t_1t_2t_3r_3| \geq |t_1r_1| > |t_1t_2r_2|$.
- If $|r_1| \leq |t_2r_2| \wedge |r_2| > |t_3r_3|$, then $|x| = |r_2| - |t_3r_3|$ and $|y| = |t_1t_2t_3r_3| + |r_2| - |t_3r_3| = |t_1t_2r_2| \geq \max(|t_1r_1|, |t_1t_2t_3r_3|)$.
- If $|r_1| > |t_2r_2| \wedge |t_3r_3| < |r_2| + |r_1| - |t_2r_2|$, then $|x| = |r_1| - |t_2t_3r_3|$ and $|y| = |t_1t_2t_3r_3| + |r_1| - |t_2t_3r_3| = |t_1r_1| \geq \max(|t_1t_2r_2|, |t_1t_2t_3r_3|)$.

Consider the language

$$L' = (u,\varepsilon)[(s_1,\overline{r}_1 t_1r_1) + (s_2,\overline{r_2}\,t_2r_2)]^*(s_3,\overline{r}_3 t_3r_3)w$$

for that we show that $\mathsf{lcs}^{\rho}(L) = \mathsf{lcs}^{\rho}(L')$. We have $L' \subseteq L$ and thus $R = \mathsf{lcs}^{\rho}(L) \overset{s}{\sqsubseteq} \mathsf{lcs}^{\rho}(L')$. Note that $(u, \varepsilon)(s_3, \overline{r_3}\,t_3r_3)w$ cannot be a witness w.r.t. uw as its length after reduction is strictly smaller than that of $r_X\kappa$, hence the two words have to coincide on at least the last $1+|R|$ letters s.t. $\mathsf{lcs}^{\rho}(L') \overset{s}{\sqsubseteq} \mathsf{lcs}^{\rho}(r_X\kappa, (u, \varepsilon)(s_3, \overline{r_3}\,t_3r_3)w) = R$, i.e., $\mathsf{lcs}^{\rho}(L) = \mathsf{lcs}^{\rho}(L') = R$. Let

$$\tilde{w} = \rho(s_3w\overline{r_3}\,t_3r_3)$$

If $\tilde{w} \overset{\rho}{=} \overline{x}\,y$ is only wwf, then $u = u'x$ and suitable conjugates of s_i exist that allow us to move x from $u = u'x$ through any sequence $s_{i_1} \dots s_{i_l}$ next to $\tilde{w}$ as done before. Thus assume w.l.o.g. that $\tilde{w}$ is already wf. By Lemma 18, we have

$$R = \mathsf{lcs}^{\rho}(L') = \mathsf{lcs}^{\rho}\begin{pmatrix} u\tilde{w} \\ us_1\tilde{w}\overline{r_1}\,t_1r_1 \\ us_2\tilde{w}\overline{r_2}\,t_2r_2 \\ us_1s_1\tilde{w}\overline{r_1}\,t_1t_1r_1 \\ us_2s_2\tilde{w}\overline{r_2}\,t_2t_2r_2 \end{pmatrix}$$

Neither $us_1\tilde{w}\overline{r_1}\,t_1r_1$ nor $us_2\tilde{w}\overline{r_2}\,t_2r_2$ can be witnesses again because their length before reduction is strictly less than that of $r_X\kappa$. Hence, either $us_1s_1\tilde{w}\overline{r}_1t_1t_1r_1$ or $us_2s_2\tilde{w}\overline{r}_2t_2t_2r_2$ is a witness w.r.t. $us_3w\overline{r_3}\,t_3r_3$ and thus also w.r.t. uw. W.l.o.g. $us_1s_1\tilde{w}\overline{r}_1t_1t_1r_1$ is a witness. Consider then the language

$$L'' = (u, \varepsilon)[(s_1, \overline{r}_1t_1r_1)^* + (s_3, \overline{r_3}\,t_3r_3)^*]w$$

Again $L'' \subseteq L$ s.t. $R = \mathsf{lcs}^{\rho}(L) \overset{s}{\sqsubseteq} \mathsf{lcs}^{\rho}(L'')$. On the other side also $\mathsf{lcs}^{\rho}(L'') \overset{s}{\sqsubseteq} \mathsf{lcs}^{\rho}(uw, us_1s_1w\overline{r_1}\,t_1t_1r_1) = R$ such that $R = \mathsf{lcs}^{\rho}(L) = \mathsf{lcs}^{\rho}(L'')$. Using Lemma 18 and now that $r_1, r_2, r_3 \overset{s}{\sqsubseteq} R = \mathsf{lcs}^{\rho}(L'')$, we obtain

$$R = \mathsf{lcs}^{\rho}(L'') = \mathsf{lcs}^{\rho}\begin{pmatrix} uw \\ us_1w\overline{r_1}\,t_1r_1 \\ us_3w\overline{r_3}\,t_3r_3 \\ us_1s_3w\overline{r_3}\,t_3r_3\overline{r_1}\,t_1r_1 \\ us_3s_1w\overline{r_1}\,t_1r_1\overline{r_3}\,t_3r_3 \end{pmatrix}$$

But by our assumption that $r_X\kappa$ is a witness w.r.t. uw of minimal length before reduction, none of theses words can be witnesses. Hence, our assumption that such a factorization exists, cannot hold.

So, every path leading to the occurrence of b that defines the lcs^{ρ} of L or to a letter right of it has to have height at most $3N$. By minimality, we can also assume that any path fragment that leads from the main path (leading to lcs^{ρ}-defining occurrence of b) to a letter left of this b contains any nonterminal at most once, see Fig. 2.3. Hence, the derivation tree can have height at most $4N$. □

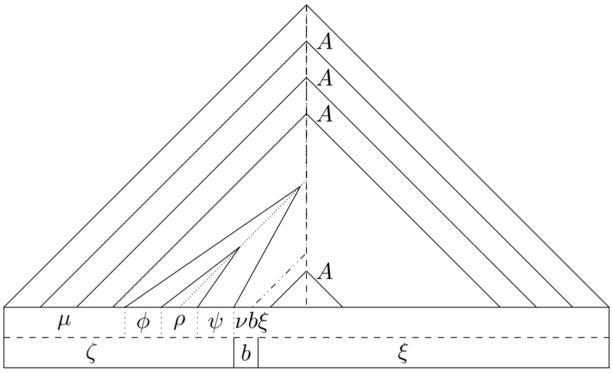


Figure 2.3: Assume that the lcs^ρ-defining occurrence of b is not contained in r_X such that $\kappa = \zeta b\xi$ with both $r_X\zeta$ and ξ wf and $\rho(r_X\kappa) = \rho(r_X\zeta)b\,\mathsf{lcs}^\rho(L)$. Consider any path that leads to a letter within ζ. (If b is contained in r_X, then this cannot happen.) The first nonterminal along this path that is not also contained in the path leading to b defines a subtree that does not contain the marked b anymore. Assume this subtree contains a path with at least $N+1$ nonterminals such that we can factorize $\zeta = (\mu,\nu)(\phi,\psi)\rho$. Then $\kappa = (\mu,\nu b\xi)(\phi,\psi)\rho$ and $L' = (\mu,\nu b\xi)(\phi,\psi)^*\rho$ is a sublanguage of L_X; thus $r_X L'$ is wf. Hence, $(r_X\mu,\nu b\xi)(\phi,\psi)^0\rho = r_X\mu\rho\nu b\xi$ is wf. As $b\xi$ is wf, too, we have that $r_X\mu\rho\nu b\xi \overset{\rho}{=} \rho(r_X\mu\rho\nu)b\,\mathsf{lcs}^\rho(L)$ is a shorter (before reduction) witness than κ. Hence, we can always assume that all subtrees rooted at a node left of the path leading to the marked b have height at most $n-1$. Thus, if all paths leading to a letter within $b\xi$ contain at most $3N$ nonterminals, then the derivation tree can have at most height $4N$.

If a context-free grammar G contains only words over A^* then G is well-formed by definition and $\mathsf{lcs}(\rho(L(G))) = \mathsf{lcs}(L(G))$. We therefore conclude that the longest common suffix of a context-free grammar (over A^*) can be computed in polynomial time (cf. [LPS18]).

Corollary 3. *Let G be a context-free grammar. Then an SLP representing the longest common suffix of $L(G)$ can be computed in polynomial time.*

Convergence for not well-formed CFGs We consider the case that the fixed-point iteration 2.1 is applied on a context-free grammar that produces a language that is *not* well-formed. Note that, if all constants $r_X\overline{r_Y}$ and all $\mathsf{T}_X^{\leq h}$ are wf, but G is not wf, then the computation has to fail while computing $r_X\overline{r_Y}\,\mathsf{T}_Y^{\leq h}\overline{r_Z}$ as illustrated by the following example.

Example 13. *Consider the nonnegative context-free grammar G given by the rules*

$$\begin{array}{llllllllllll} S & \to & Uc & U & \to & AV \mid W_n & V & \to & U\overline{B} & W_i & \to & W_{i-1}W_{i-1} \quad (2 \le i \le n) \\ A & \to & a & B & \to & b & \overline{B} & \to & \overline{b} & W_1 & \to & BB \end{array}$$

with axiom S and a fixed parameter $n \in \mathbb{N}$. Except for $\overline{B}$ all nonterminals generate nonnegative languages. Note that the nonterminals W_n to W_1 form an SLP *that encodes the word b^{2^n} by means of iterated squaring which only becomes productive at height $h = n+1$. For $h \ge n+3$ we have:*

$$\begin{array}{lllllllll} L_S^{\le h} & = & \{a^k b^{2^n} \overline{b}^{\,k} c \mid k \le \lfloor \frac{h-(n+3)}{2} \rfloor\} & & & & & & \\ L_U^{\le h} & = & \{a^k b^{2^n} \overline{b}^{\,k} \mid k \le \lfloor \frac{h-(n+2)}{2} \rfloor\} & L_{W_i}^{\le h} & = & \{b^{2^i}\} & L_B^{\le h} & = & \{b\} \\ L_V^{\le h} & = & \{a^k b^{2^n} \overline{b}^{\,k+1} \mid k \le \lfloor \frac{h-(n+3)}{2} \rfloor\} & L_A^{\le h} & = & \{a\} & L_{\overline{B}}^{\le h} & = & \{\overline{b}\} \end{array}$$

Here the words r_X used to cancel the longest prefix of closing brackets (after reduction) are $r_S = r_U = r_V = r_W = r_A = r_B = \varepsilon$ and $r_{\overline{B}} = b$. Note that $r_X L_X^{\le h}$ is wf *for all nonterminals X up to $h \le h_0 = 2^{n+1} + (n+2)$ s.t. $\mathsf{T}_{\mathsf{lcs}}(\rho(r_S L_S^{\le h})) \approx_{\mathsf{lcs}} \mathsf{T}_S^{\le h} = \{b^{2^n}c, a^k b^{2^n - k(h)} c\}$ for $k(h) = \lfloor (h-(n+3))/2 \rfloor$ and $n+3 \le h \le h_0$; in particular, the* lcs *of $\mathsf{T}_S^{\le h}$ has already converged to c at $h = n+3$, only its maximal extension* lcsext *changes for $n+3 \le h \le h_0$. We discover the first counterexample $a^{2^n}\overline{b}$ that G is not* wf *while computing $\mathsf{T}_V^{\le h_0+1} = \mathsf{T}_{\mathsf{lcs}}(\rho(\mathsf{T}_U^{\le h_0}\overline{b}))$.*

As illustrated in Example 13, if G is not wf, then the minimal derivation height $h_0 + 1$ at which we discover a counterexample might be exponential in the size of the grammar. The following lemma states that up to this derivation height h_0 the representations $\mathsf{T}_X^{\le h}$ cannot have converged (modulo $\approx_{\mathsf{lcs}}$).

Lemma 12. *If $L = L(G)$ is not* wf*, then there is some least h_0 s.t. $r_X L_Y^{\le h_0} \overline{r_Z}$ is not* wf *with $X \to_G YZ$. For $h \le h_0$, all $r_X L_X^{\le h}$ are* wf *s.t. $T_X^{\le h} \approx_{\mathsf{lcs}} \rho(r_X L_X^{\le h})$. If $h_0 \ge 4N+1$, then at least for one nonterminal X we have $T_X^{\le 4N+1} \not\approx_{\mathsf{lcs}} T_X^{\le 4N}$.*

Proof. We write $\mathsf{T}^{\rho}_{\mathsf{lcs}}(L)$ and $\mathsf{lcs}^{\rho}(L)$ for $\mathsf{T}_{\mathsf{lcs}}(\rho(L))$ and $\mathsf{lcs}(\rho(L))$, respectively. We assume that all nullary rules $X \to \overline{u}\,v$ are already reduced and w.l.o.g. wwf. Further w.l.o.g. G is nonnegative (cf. Lemma 5) and for all constant rules $X \to_G r$, $r_X r$ is wf. Then there is some word $\alpha \in L(G)$ that is *not* wf. We show that there is some rule $X \to_G YZ$ and words $\alpha_X = \alpha_Y \alpha_Z$ with $\alpha_Y \in L_Y$ and $\alpha_Z \in L_Z$ such that

- $r_X \alpha_Y \overline{r_Z}$ is not wf,
- $r_X \overline{r_Y}$ is wf and
- $r_Y \alpha_Y$ is wf.

To this end, consider any derivation of α. We set $X := S$ and $\alpha_X := \alpha$. We have $r_X = r_S = \varepsilon$ with $r_X \alpha_X$ is not wf.

As $r_X\alpha_X$ is not wf, there is some rule $X \to_G YZ$ and factorization $\alpha_X = \alpha_Y\alpha_Z$. Otherwise we have a contradiction to the assumption that r_Xr is wf for all constant rules $X \to_G r$.

If $r_Y\alpha_Y$ is not wf:

Redefine $X := Y$ and $\alpha_X := \alpha_Y$ and descend accordingly into the derivation tree of α_Y.

If $r_Z\alpha_Z$ is not wf:

Redefine $X := Z$ and $\alpha_X := \alpha_Z$ and descend accordingly into the derivation tree of α_Z.

Otherwise $r_Z\alpha_Z$ is wf, thus $\alpha_Z \overset{\rho}{=} \overline{u_Z}\, v_Z$ with $r_Z = r'_Z u_Z$.

Thus $r_X\alpha_Y\overline{r_Z}$ is not wf as $r_X\alpha_X = r_X\alpha_Y\alpha_Z \overset{\rho}{=} r_X\alpha_Y\overline{r_Z}\, r'_Z v_Y$

This procedure yields that there is some least derivation height n_0 such that

- $r_X L_X^{\leq n_0}$ is wf for every nonterminal X,
- $r_X\overline{r_Y}$ is wf for all rules $X \to_G YZ$ and
- there exists a nonterminal X_0 with $X_0 \to_G YZ$, $\alpha_Y \in L_Y^{n_0}$, and $r_{X_0}\alpha_Y\overline{r_Z}$ not wf anymore.

As all $r_Y L_Y^{\leq n_0}$ are wf, we have $\rho(r_Y L_Y^{\leq n_0}) \approx_{\mathsf{lcs}} \mathsf{T}^{\rho}_{\mathsf{lcs}}(r_Y L_Y^{\leq n_0}) \approx_{\mathsf{lcs}} \mathsf{T}_Y^{\leq n_0}$. Thus also

$$\rho(r_X L_Y^{\leq n_0}) = \rho(r_X\overline{r_Y}\,)\,\rho(r_Y L_Y^{\leq n_0}) \approx_{\mathsf{lcs}} \rho(r_X\overline{r_Y}\,)\mathsf{T}_Y^{\leq n_0}$$

As G is nonnegative, also $r_X L_Y^{\leq n_0}\overline{r_Z}$ is nonnegative. Thus, as $r_X\alpha_Y\overline{r_Z}$ is not wf, we have that $\mathsf{lcs}^{\rho}(r_X L_Y^{\leq n_0})\overline{r_Z}$ is not wf, and thus $\rho(r_X\overline{r_Y}\,)\mathsf{T}_Y^{\leq n_0}\overline{r_Z}$ is not wf. If $n_0 \leq 4N{+}1$ we discover the error by iteratively computing $\mathsf{T}_X^{\leq h} \approx_{\mathsf{lcs}} \mathsf{T}_{\mathsf{lcs}}(\rho(r_X L_X^{\leq h}))$. Otherwise

$$r_Z \overset{s}{\sqsubseteq} \mathsf{lcs}^{\rho}(r_X L_Y^{\leq 4N+1}) = \mathsf{lcs}(\rho(r_X\overline{r_Y}\,)\mathsf{T}_X^{\leq 4N+1}) \text{ but } r_Z \overset{s}{\not\sqsubseteq} \mathsf{lcs}^{\rho}(r_X L_Y^{\leq n_0})$$

Thus $\rho(r_X L_Y^{\leq 4N+1}) \not\approx_{\mathsf{lcs}} r_X L_Y^{\leq n_0}$ and thus $r_Y L_Y^{\leq 4N+1} \not\approx_{\mathsf{lcs}} r_Y L_Y^{\leq n_0}$. Therefore, for at least one nonterminal lcssum cannot have converged. □

Lemmas 11 and 12 yield that the fixed-point iteration (2.1) converges latest in step $4N$ if and only if the language given by the context-free grammar G is well-formed. Deciding well-formedness in polynomial time therefore amounts to the following steps. Let G be a context-free grammar with N the number of nonterminals. W.l.o.g. we assume that all nonterminals are reachable from the axiom S and productive.

- For all nonterminals X in G compute SLPs representing r_X and check whether G produces any negative word, cf. Lemmas 4 and 5. If r_S for the axiom S is not empty or G is not nonnegative then G is not well-formed.
- For all nonterminals X compute iteratively $\mathsf{T}_X^{\leq h}$ (2.1) up to height at most $4N + 1$.

- If for all nonterminals X the languages $\mathsf{T}_X^{\leq 4N}$ converged, i.e., $\mathsf{T}_X^{\leq 4N} = \mathsf{T}_X^{\leq 4N+1}$, then the language produced by G is well-formed.
  Otherwise, if during the fixed-point iteration some calculation failed as the result would not be well-formed or there is some nonterminal X such that the representation $\mathsf{T}_X^{\leq 4N}$ has not converged, i.e., $\mathsf{T}_X^{\leq 4N} \neq \mathsf{T}_X^{\leq 4N+1}$, then the language produced by G is not well-formed.

Theorem 1. *Well-formedness of context-free grammars over B can be decided in polynomial time.*

2.4 Reduced Longest Common Suffix Computation

This section contains the combinatorical results underlying the proofs of Lemmas 11 and 12. They are concerned with the reduced lcs of *simple* linear grammars of the form

$$S \to \alpha X \beta \qquad X \to \sigma_1 X \tau_1 \mid \ldots \mid \sigma_k X \tau_k \mid \gamma \qquad (\alpha, \beta, \sigma_i, \tau_i, \gamma \in \mathsf{B}^*)$$

Assuming that the grammar is well-formed, we showed in Lemma 10 that we can rewrite each rule so that the simple linear grammar takes the form

$$S \to uX \qquad X \to s_1 X \tau_1 \mid \ldots \mid s_k X \tau_k \mid w$$

with $u, v, w, s_i, r_i, t_i \in \mathsf{A}^*$ and $\tau_i = \overline{r_i}\, t_i r_i$ or $\tau_i = \overline{r_i}\, \overline{t_i}\, r_i$. Both grammars generate the same language after reduction, and there is a one-to-one correspondence of the rewrite rules s.t. the derivations of both grammars are in bijection.
These results generalize and improve on the results of [LPS18] by specifically partitioning L into classes of conjugates w.r.t. $R = \mathsf{lcs}^\rho(L)$ which allows us to swap equivalent witnesses for R. Lemmas 13 and 14 consider the longest common suffix in the case without closing letters. Both lemmas are stronger versions of the analogous results for the longest common prefix as presented in [LPS18]. Most importantly, both lemmas now state that, if e.g. $(u, \varepsilon)(s_1, t_1)^2 w = u s_1^2 w t_1^2$ is a witness of the lcs w.r.t. $(u, \varepsilon) w = uw$, then also $(u, \varepsilon)(s_1, t_1)(s_2, t_2) w = u s_1 s_2 w t_2 t_1$ is a witness w.r.t. uw; i.e. only the outer context resp. pumping tree matters in the end.

Lemma 13. *Let $L = (u, \varepsilon)[(s_1, t_1) + (s_2, t_2)]^* \varepsilon$ be well-formed. Then*

$$\mathsf{lcs}(L) = \mathsf{lcs}\left(\begin{array}{r} u \\ u s_1 t_1 \\ u s_1 s_1 t_1 t_1 \\ u s_2 t_2 \\ u s_2 s_2 t_2 t_2 \end{array}\right)$$

Additionally, if $s_it_i \neq \varepsilon$, $i \in \{1,2\}$ holds, then

$$\mathsf{lcs}(L) = \mathsf{lcs}\begin{pmatrix} u \\ us_1t_1 \\ us_1s_2t_2t_1 \\ us_2t_2 \\ us_2s_1t_1t_2 \end{pmatrix}$$

If $s_i = \varepsilon$, then $us_is_it_it_i$ is not required.

Proof. First, assume that $s_2t_2 = \varepsilon$. Then in [LPS18, Theorem 7] was shown that $\mathsf{lcs}(L) = \mathsf{lcs}(u, us_1t_1, us_1s_1t_1t_1)$. In [LPS18] the longest common prefix was considered, however, the case of the longest common suffix is symmetric.
We thus will assume from here on that $s_1t_1 \neq \varepsilon \neq s_2t_2$. Let $R := \mathsf{lcs}(L)$ and $u = u'R$. As $R \overset{s}{\sqsubseteq} us_i^kt_i^k$ for all $k \geq 0$ we have $R \overset{s}{\sqsubset} t_i^\omega$ if $t \neq \varepsilon$. Thus, it exists $\hat{t}_i$ such that $\boldsymbol{Rt_i = \hat{t}_iR}$. If $t_i = \varepsilon$, then set $\hat{t}_i = \varepsilon$. We show that in the case that $R \overset{s}{\sqsubset} t_1 = t'_1R$ and $R \overset{s}{\sqsubset} t_2 = t'_2R$ then

$$\mathsf{lcs}(L) = \mathsf{lcs}\begin{pmatrix} u \\ us_1t_1 \\ us_2t_2 \end{pmatrix} = \mathsf{lcs}\begin{pmatrix} u' \\ us_1t'_1 \\ us_2t'_2 \end{pmatrix} R$$

As $t'_1 \neq \varepsilon \neq t'_2$ we get $L = (u'R, \varepsilon)[(s_1, t'_1R) + (s_2, t'_2R)]^*\varepsilon$. Therefore $\mathsf{lcs}(u', t'_1) = \varepsilon$ or $\mathsf{lcs}(u', t'_2) = \varepsilon$. Thus, assume w.l.o.g. that $t_1 \overset{s}{\sqsubseteq} R$ with $\boldsymbol{R = \dot{R}t_1}$ from here on. Then $\dot{R}t_1t_1 = Rt_1 = \hat{t}_1R = \hat{t}_1\dot{R}t_1$. Thus, canceling t_1 from the right yields $\boldsymbol{R = \dot{R}t_1 = \hat{t}_1\dot{R}}$. Additionally, we observe that $\dot{R} \overset{s}{\sqsubset} \dot{R}s_1$ as $\dot{R}t_1 = R \overset{s}{\sqsubseteq} us_1t_1 = u'\dot{R}t_1s_1t_1 = u'\hat{t}_1\dot{R}s_1t_1$. Therefore, it exists $\dot{s}_1$ such that $\boldsymbol{\dot{R}s_1 = \dot{s}_1\dot{R}}$. If $s_1 = \varepsilon$, we set $\dot{s}_1 = \varepsilon$.
We consider the case $R \overset{s}{\sqsubset} t_2 = t'_2R = t'_2\hat{t}_1\dot{R}$ with $t'_2 \neq \varepsilon$. Then we have

$$\begin{aligned}
(u,\varepsilon)(s_1,t_1)^+\varepsilon &= (u'\hat{t}_1\dot{R},\varepsilon)(s_1,t_1)^+\varepsilon \\
&= (u'\hat{t}_1\dot{s}_1, R)(\dot{s}_1,\hat{t}_1)^*\varepsilon \\
(u,\varepsilon)(s_1,t_1)^*(s_2,t_2)[(s_1,t_1)+(s_2,t_2)]^*\varepsilon &= (u,\varepsilon)(s_1,t_1)^*(s_2,t'_2R)[(s_1,t_1)+(s_2,t_2)]^*\varepsilon \\
&= (u,R)(s_1,\hat{t}_1)^*(s_2,t'_2)[(s_1,t_1)+(s_2,t_2)]^*\varepsilon
\end{aligned}$$

Therefore $\varepsilon \overset{!}{=} \mathsf{lcs}(u', \hat{t}_1\dot{s}_1, \dot{s}_1\hat{t}_1^+, t'_2\hat{t}_1^*)$. If $\mathsf{lcs}(u', \hat{t}_1\dot{s}_1) = \varepsilon$ then us_1t_1 is a witness. If $\mathsf{lcs}(u', \dot{s}_1\hat{t}_1^+) = \varepsilon$ and $\hat{t}_1 \neq \varepsilon$ then $\mathsf{lcs}(u', \dot{s}_1\hat{t}_1) = \varepsilon$ and $us_1s_1t_1t_1$ is a witness and $us_2s_1t_1t_2$. Note that if $\hat{t}_1 = \varepsilon$ then $\dot{s}_1 \neq \varepsilon$ and we are in the first case where us_1t_1 is a witness. If $\mathsf{lcs}(u', t'_2) = \varepsilon$ then us_2t_2 is a witness.

We therefore consider the case that $t_2 \overset{s}{\sqsubseteq} R$ with $\boldsymbol{R = \ddot{R}t_2}$ from here on. Then $\ddot{R}t_2t_2 = Rt_2 = \hat{t}_2R = \hat{t}_2\ddot{R}t_2$ and canceling t_2 from the right yields $\boldsymbol{R = \ddot{R}t_2 = \hat{t}_2\ddot{R}}$ and $\boldsymbol{u = u'R = u'\ddot{R}t_2 = u'\hat{t}_2\ddot{R}}$. W.l.o.g. we assume that $|t_1| \leq |t_2|$. We find conjugates $\ddot{s}_2, z, \ddot{z}, \ddot{s}_1, \ddot{t}_1$ such that

- $\boldsymbol{\ddot{R}s_2 = \ddot{s}_2\ddot{R}}$ as $\ddot{R}t_2 = R \overset{s}{\sqsubseteq} us_2t_2 = u'\ddot{R}t_2s_2t_2 = u'\hat{t}_2\ddot{R}s_2t_2$ and therefore $\ddot{R} \overset{s}{\sqsubseteq} \ddot{R}s_2$.
- $\boldsymbol{t_2 = zt_1} \wedge \boldsymbol{\dot{R} = \ddot{R}z}$ as $R = \dot{R}t_1 = \ddot{R}t_2$ and $|t_1| \leq |t_2|$; therefore $\ddot{R}zt_1 = \dot{R}t_1$.
- $\boldsymbol{\ddot{R}z = \ddot{z}\ddot{R}}$ as $\hat{t}_2\ddot{R} = \ddot{R}t_2 = R = \dot{R}t_1 = \hat{t}_1\dot{R} = \hat{t}_1\ddot{R}z$ and therefore $\ddot{R} \overset{s}{\sqsubseteq} \ddot{R}z$.
- $\boldsymbol{\ddot{R}s_1 = \ddot{s}_1\ddot{R}}$ as $\ddot{R}t_1 \overset{s}{\sqsubseteq} \ddot{z}\ddot{R}t_1 = \ddot{R}zt_1 = \ddot{R}t_2 = R \overset{s}{\sqsubseteq} us_1t_1 = u'\hat{t}_2\ddot{R}s_1t_1$ and therefore $\ddot{R} \overset{s}{\sqsubseteq} \ddot{R}s_1$ if $t_2 \neq \varepsilon$. Otherwise, if $t_2 = \varepsilon$ then $\ddot{R} = Rt_2 = R = \dot{R} = \ddot{R}$ and $\ddot{s}_1 = \dot{s}_1 = s$.
- $\boldsymbol{\ddot{R}t_1 = \ddot{t}_1\ddot{R}}$ as $\hat{t}_2\ddot{R} = \ddot{R}t_2 = R \overset{s}{\sqsubseteq} \hat{t}_1R = Rt_1 = \ddot{R}t_2t_1 = \hat{t}_2\ddot{R}t_1$ and therefore $\ddot{R} \overset{s}{\sqsubseteq} \ddot{R}t_1$.
- $\boldsymbol{\hat{t}_2 = \ddot{z}\ddot{t}_1 = \hat{t}_1\ddot{z}}$ as $\hat{t}_2\ddot{R} = R = \dot{R}t_1 = \hat{t}_1\dot{R} = \hat{t}_1\ddot{R}z = \hat{t}_1\ddot{z}\ddot{R}$ and thus $\hat{t}_2 = \hat{t}_1\ddot{z}$. Additionally, $\hat{t}_2\ddot{R} = R = \ddot{R}t_2 = \ddot{R}zt_1 = \ddot{z}\ddot{R}t_1 = \ddot{z}\ddot{t}_1\ddot{R}$ holds and thus $\hat{t}_2 = \ddot{z}\ddot{t}_1$.

Using these conjugates we obtain

$$\begin{array}{cll}
 & (u,\varepsilon)(s_1,t_1)^+\varepsilon & \\
= & (u'\hat{t}_1\dot{R},\varepsilon)(s_1,t_1)^+\varepsilon & (u = u'R \wedge R = \hat{t}_1\dot{R}) \\
= & (u'\hat{t}_1\dot{s}_1,R)(\dot{s}_1,\hat{t}_1)^*\varepsilon & (\dot{R}s_1 = \dot{s}_1\dot{R} \wedge R = \dot{R}t_1 \wedge Rt_1 = \hat{t}_1R) \\
= & u'\hat{t}_1\dot{s}_1R & us_1t_1 \\
+ & (u'\hat{t}_1\dot{s}_1\dot{s}_1,\hat{t}_1R)(\dot{s}_1,\hat{t}_1)^*\varepsilon & (u,\varepsilon)(s_1,t_1)^{\geq 2}\varepsilon
\end{array}$$

$$\begin{array}{ll}
\quad (u,\varepsilon)(s_1,t_1)^*(s_2,t_2)[(s_1,t_1)+(s_2,t_2)]^*\varepsilon & \\
= (u'\hat{t}_2\ddot{R},\varepsilon)(s_1,t_1)^*(s_2,t_2)[(s_1,t_1)+(s_2,t_2)]^*\varepsilon & u = u'\hat{t}_2\ddot{R} \\
= (u'\hat{t}_2,\ddot{R})(\ddot{s}_1,\ddot{t}_1)^*(\ddot{s}_2,\hat{t}_2)[(\ddot{s}_1,\ddot{t}_1)+(\ddot{s}_2,\hat{t}_2)]^*\varepsilon & \ddot{R}s_i = \ddot{s}_i\ddot{R} \wedge \ddot{R}t_1 = \ddot{t}_1\ddot{R} \wedge \\
 & \ddot{R}t_2 = \hat{t}_2\ddot{R} \\
= (u'\hat{t}_2,\ddot{R})(\ddot{s}_1,\ddot{t}_1)^*(\ddot{s}_2,\hat{t}_1\ddot{z})[(\ddot{s}_1,\ddot{t}_1)+(\ddot{s}_2,\hat{t}_2)]^*\varepsilon & \hat{t}_2 = \hat{t}_1\ddot{z} \\
= (u'\hat{t}_2,\ddot{z}\ddot{R})(\ddot{s}_1,\hat{t}_1)^*(\ddot{s}_2,\hat{t}_1)[(\ddot{s}_1,\ddot{t}_1)+(\ddot{s}_2,\hat{t}_2)]^*\varepsilon & \ddot{z}\ddot{t}_1 = \hat{t}_1\ddot{z} \\
= (u'\hat{t}_2,\dot{R})(\ddot{s}_1,\hat{t}_1)^*(\ddot{s}_2,\hat{t}_1)[(\ddot{s}_1,\ddot{t}_1)+(\ddot{s}_2,\hat{t}_2)]^*\varepsilon & \ddot{z}\ddot{R} = \ddot{R}z \wedge \dot{R} = \ddot{R}z \\
= (u'\hat{t}_2,\hat{t}_1\dot{R})(\ddot{s}_1,\hat{t}_1)^*(\ddot{s}_2,\varepsilon)[(\ddot{s}_1,\ddot{t}_1)+(\ddot{s}_2,\hat{t}_2)]^*\varepsilon & \text{move } \hat{t}_1 \text{ from } (\ddot{s}_2,\hat{t}_1) \text{ to the} \\
 & \text{right} \\
= (u'\hat{t}_2,R)(\ddot{s}_1,\hat{t}_1)^*(\ddot{s}_2,\varepsilon)[(\ddot{s}_1,\ddot{t}_1)+(\ddot{s}_2,\hat{t}_2)]^*\varepsilon & R = \hat{t}_1\dot{R} \\
= (u'\ddot{z}\ddot{t}_1,R)(\ddot{s}_1,\hat{t}_1)^*(\ddot{s}_2,\varepsilon)[(\ddot{s}_1,\ddot{t}_1)+(\ddot{s}_2,\ddot{z}\ddot{t}_1)]^*\varepsilon & \hat{t}_2 = \ddot{z}\ddot{t}_1 \\
= u'\ddot{z}\ddot{t}_1\ddot{s}_2R & us_2t_2 \\
+ (u'\ddot{z}\ddot{t}_1\ddot{s}_1,\hat{t}_1R)(\ddot{s}_1,\hat{t}_1)^*(\ddot{s}_2,\varepsilon)[(\ddot{s}_1,\ddot{t}_1)+(\ddot{s}_2,\ddot{z}\ddot{t}_1)]^*\varepsilon & (u,\varepsilon)(s_1,t_1)^+(s_2,t_2)[(s_i,t_i)]^*\varepsilon
\end{array}$$

$$\begin{array}{ll} + (u'\ddot{z}\ddot{t}_1, \ddot{t}_1 R)(\ddot{s}_2\ddot{s}_1, \varepsilon)[(\ddot{s}_1, \ddot{t}_1) + (\ddot{s}_2, \ddot{z}\ddot{t}_1)]^*\varepsilon & (u, \varepsilon)(s_2, t_2)(s_1, t_1)[(s_i, t_i)]^*\varepsilon \\ + (u'\ddot{z}\ddot{t}_1, \ddot{z}\ddot{t}_1 R)(\ddot{s}_2\ddot{s}_2, \varepsilon)[(\ddot{s}_1, \ddot{t}_1) + (\ddot{s}_2, \ddot{z}\ddot{t}_1)]^*\varepsilon & (u, \varepsilon)(s_2, t_2)^{\geq 2}[(s_i, t_i)]^*\varepsilon \end{array}$$

If us_1t_1 or us_2t_2 is a witness then the claim of the lemma follows. Thus, assume that neither us_1t_1 nor us_2t_2 is a witness w.r.t. u, i.e.

$$\mathsf{lcs}(u', \hat{t}_1\dot{s}_1, \ddot{z}\ddot{t}_1\ddot{s}_2) \neq \varepsilon$$

W.l.o.g. $t_2 \neq \varepsilon$ and thus also $\hat{t}_2 \neq \varepsilon$ as otherwise $t_1 = \varepsilon$ as $0 = |t_2| \geq |t_1|$ s.t. $R = \dot{R} = \ddot{R}$ and $\dot{s}_1 = \ddot{s}_1$. Then $L = u(s_1 + s_2)^* = u'(\ddot{s}_1 + \ddot{s}_2)^* R$ and thus $\mathsf{lcs}(u', \ddot{s}_1, \ddot{s}_2) = \varepsilon$. Therefore $us_1t_1 = us_1$ or $us_2t_2 = us_2$ would be a witness.

Assume that $t_1 = \varepsilon$. Then we have $\hat{t}_1 = \ddot{t}_1 = \varepsilon$ and $\dot{R} = R$ and $\dot{s}_1 \neq \varepsilon \neq \ddot{s}_1$ and $\hat{t}_2 = \ddot{z}\ddot{t}_1 = \ddot{z} \neq \varepsilon$ s.t.

$$(u, \varepsilon)(s_1, t_1)^+\varepsilon = u'\dot{s}_1^+ R$$

$$(u, \varepsilon)(s_1, t_1)^*(s_2, t_2)[(s_1, t_1) + (s_2, t_2)]^*\varepsilon = (u'\ddot{z}, R)(\ddot{s}_1, \varepsilon)^*(\ddot{s}_2, \varepsilon)[(\ddot{s}_1, \varepsilon) + (\ddot{s}_2, \ddot{z})]^*\varepsilon$$

Therefore $\varepsilon \overset{!}{=} \mathsf{lcs}(u', \dot{s}_1, \ddot{z}\ddot{s}_2, \ddot{s}_1, \ddot{z})$. If $\mathsf{lcs}(u', \dot{s}_1) = \varepsilon$ then us_1t_1 would be a witness.If $\mathsf{lcs}(u', \ddot{z}\ddot{s}_2) = \varepsilon$ then us_2t_2 would be a witness. We therefore need to consider the cases $\mathsf{lcs}(u', \ddot{s}_1) = \varepsilon$ and $\mathsf{lcs}(u', \ddot{z}) = \varepsilon$. In fact, $\mathsf{lcs}(\ddot{s}_1, \ddot{z}) \neq \varepsilon$ holds as $\ddot{z}\ddot{R} = \ddot{z}\ddot{t}_1\ddot{R} = \hat{t}_2\ddot{R} = R \overset{s}{\sqsubseteq} us_1^k t_1^k = u'Rs_1^k = u'\ddot{z}\ddot{R}s_1^k = u'\ddot{z}\ddot{s}_1^k\ddot{R}$ for all k and therefore $\ddot{z} \overset{s}{\sqsubset} \ddot{s}_1^{\omega}$. Thus $\mathsf{lcs}(u', \ddot{s}_1) = \varepsilon$ if and only if $\mathsf{lcs}(u', \ddot{z}) = \varepsilon$ and therefore $us_1s_2t_2t_1$ is a witness if and only if $us_2s_2t_2t_2$ is a witness.

Last, we consider the case that $t_1 \neq \varepsilon$. Then $\hat{t}_1 \neq \varepsilon \neq \ddot{t}_1$ and therefore $\varepsilon \overset{!}{=} \mathsf{lcs}(u', \hat{t}_1, \ddot{t}_1)$. We obtain

$$\begin{aligned} \mathsf{lcs}(L) &= \mathsf{lcs}\begin{pmatrix} u \\ us_1s_1t_1t_1 \\ us_2s_2t_2t_2 \end{pmatrix} = \mathsf{lcs}\begin{pmatrix} u'R \\ u'\hat{t}_1\dot{s}_1\dot{s}_1\hat{t}_1R \\ u'\ddot{z}\ddot{t}_1\ddot{s}_2\ddot{s}_2\ddot{z}\ddot{t}_1R \end{pmatrix} \\ &= \mathsf{lcs}\begin{pmatrix} u \\ us_1s_2t_2t_1 \\ us_2s_1t_1t_2 \end{pmatrix} = \mathsf{lcs}\begin{pmatrix} u'R \\ u'\ddot{z}\ddot{t}_1\ddot{s}_1\ddot{s}_2\hat{t}_1R \\ u'\ddot{z}\ddot{t}_1\ddot{s}_2\ddot{s}_1\ddot{t}_1R \end{pmatrix} \end{aligned}$$

□

In Lemma 13 we considere languages of the form $(u, \varepsilon)[(s_1, t_1) + (s_2, t_2)]^*\varepsilon$. Now, we extend this result to languages of the form $(u, \varepsilon)[(s_1, t_1) + (s_2, t_2)]^*w$.

Lemma 14. *Let $L = (u, \varepsilon)[(s_1, t_1) + (s_2, t_2)]^* w$ be well-formed.Then*

$$\mathsf{lcs}(L) = \mathsf{lcs}\left(\begin{array}{r} uw \\ us_1wt_1 \\ us_1s_1wt_1t_1 \\ us_2wt_2 \\ us_2s_2wt_2t_2 \end{array}\right)$$

and additionally, if $s_it_i \neq \varepsilon$ then

$$\mathsf{lcs}(L) = \mathsf{lcs}\left(\begin{array}{r} uw \\ us_1wt_1 \\ us_1s_2wt_2t_1 \\ us_2wt_2 \\ us_2s_1wt_1t_2 \end{array}\right)$$

If $s_i = \varepsilon$, then $us_is_iwt_it_i$ is not required.

Proof. Assume first that $s_2t_2 = \varepsilon$. Then in [LPS18, Theorem 7] was shown that $\mathsf{lcs}(L) = \mathsf{lcs}(uw, us_1wt_1, us_1s_1wt_1t_1)$.
We therefore assume from here on that $s_1t_1 \neq \varepsilon \neq s_2t_2$. Let $R = \mathsf{lcs}(L)$. We first consider the case $R \overset{s}{\sqsubset} w = w'R$ with $w' \neq \varepsilon$. Then $R \overset{s}{\sqsubseteq} Rt_i$ as $R \overset{s}{\sqsubseteq} us_iwt_i = us_iw'Rt_i$. Thus, it exists $\hat{t}_i$ such that $Rt_i = \hat{t}_iR$. Therefore

$$L = (u, 1)[(s_1, t_1) + (s_2, t_2)]^* w'R = (u, R)[(s_1, \hat{t}_1) + (s_2, \hat{t}_2)]^* w'$$

If follows that $\varepsilon \overset{!}{=} \mathsf{lcs}(w', w'\hat{t}_1, w'\hat{t}_2)$; thus $\mathsf{lcs}(w', w'\hat{t}_1) = \varepsilon$ or $\mathsf{lcs}(w', w'\hat{t}_2) = \varepsilon$ and therefore $\mathsf{lcs}(w', \hat{t}_1) = \varepsilon$ or $\mathsf{lcs}(w', \hat{t}_2) = \varepsilon$. With this we obtain $\mathsf{lcs}(L) = \mathsf{lcs}\left(\begin{array}{r} uw \\ us_1wt_1 \\ us_2wt_2 \end{array}\right)$.

Second, we consider the case $w \overset{s}{\sqsubseteq} R = R'w$. Then $w \overset{s}{\sqsubseteq} wt_i$ as $R = R'w \overset{s}{\sqsubseteq} us_iwt_i$. Thus, it exists $\hat{t}_i$ such that $wt_i = \hat{t}_iw$. Therefore

$$L = (u'R', w)[(s_1, \hat{t}_1) + (s_2, \hat{t}_2)]^* \varepsilon$$

and we can apply Lemma 13 to $L' = (u'R', \varepsilon)[(s_1, \hat{t}_1) + (s_2, \hat{t}_2)]^* \varepsilon$. □

We now consider the three cases where the right part of the two pumping trees can contain closing letters,

- both pumping trees are "negative" of the form $\overline{r}\,\overline{t}\,r$, i.e., $(u, \varepsilon)[(s_1, \overline{r_1}\,\overline{t_1}\,r_1) + (s_2, \overline{r_2}\,\overline{t_2}\,r_2)]^* w$,

- one pumping tree is “negative” of the form $\overline{r}\,\overline{t}\,r$ and one “negative” of the form $\overline{r}\,tr$, i.e., $(u,\varepsilon)[(s_1,\overline{r_1}\,\overline{t_1}\,r_1)+(\overline{r_2}\,t_2r_2)]^*w$,
- both pumping trees are “positive” of the form $\overline{r}\,tr$, i.e., $(u,\varepsilon)[(s_1,\overline{r_1}\,t_1r_1)+(\overline{r_2}\,t_2r_2)]^*w$.

The central observation in Lemmas 15 and 16 is that, if at least one of the contexts (s_i,τ_i) is *negative*, i.e. $\tau_i \overset{\rho}{=} \overline{r_i}\,\overline{t_i}\,r_i$ with $t_i \neq \varepsilon$, then the simple linear well-formed L can be normalized to a regular language over A whose lcs and lcsext are already determined by $(u,v)\varepsilon$ and $(u,v)(s_i,\tau_i)\varepsilon$. We give an example on that.

Example 14. *Consider the linear language L given by the rules*

$$S \to uX \qquad X \to sX\overline{r}\,\overline{t}\,r \mid \varepsilon$$

where we assume that the language is wf with $t \neq \varepsilon$ and, for the sake of this example, also $|tr| > |s|$. As $us^k\overline{r}\,\overline{t}^{\,k}r$ is wf for all $k \in \mathbb{N}$, we have $(s^{m})^R = (t^{m}r)^R$, i.e., there is a conjugate p of the primitive root q of t such that

- *$qr = rp$,*
- *$s = p^m$,*
- *$t = q^n$ and*
- *$m \geq n$ for suitable $m,n \in \mathbb{N}_0$.*

The last property has to hold as otherwise we could generate a negative word by pumping the context $(s,\overline{r}\,\overline{t}\,r)$. Further as $|tr| > |s|$ we have $tr\overline{s} \overset{\rho}{=} r\overline{p}^{\,m-n}$, s.t. $r = r'p^{m-n}$, $qr' = r'p$, and $u = u'r'$ as $us\overline{r}\,\overline{t}\,r$ is wf. We thus may replace $X \to sX\overline{r}\,tr$ with $X \to p^{m-n}X$ as

$$\begin{aligned}
us^{k+1}\overline{r}\,\overline{t}^{\,k+1}r &\overset{\rho}{=} u'r'(p^m)^{k+1}\overline{p}^{\,m-n}\overline{r'}\,(\overline{q}^{\,n})^{k+1}r'p^{m-n}\\
&\overset{\rho}{=} u'r'(p^m)^k p^n\overline{r'}\,(\overline{q}^{\,n})^{k+1}r'p^{m-n}\\
&\overset{\rho}{=} u'(q^m)^k q^n(\overline{q}^{\,n})^{k+1}q^{m-n}r'\\
&\overset{\rho}{=} u'(q^{m-n})^{k+1}r'\\
&\overset{\rho}{=} u(p^{m-n})^{k+1}
\end{aligned}$$

and obtain a regular language $L' \subseteq \mathsf{A}^*$ whose derivations are in bijection with those of L. Now, $\mathsf{lcssum}(L)$ is already determined by u and up which in turn implies that u and $us\overline{r}\,tr$ determine $\mathsf{lcssum}^\rho(L)$. In case of multiple contexts (s_j,τ_j) the existence of one context of the form $(s_i,\overline{r_i}\,\overline{t_i}\,r_i)$ enforces that all contexts have to be compatible with the primitive root of t_i which subsequently allows us to replace every rule $X \to s_iX\tau_i$ by a $\overset{\rho}{=}$-equivalent rule $X \to p^{k_i}X$ over A. □

We first consider the case that both pumping trees are negative.

Lemma 15. *Let $L = (u,\varepsilon)[(s_1,\overline{r_1}\,\overline{t_1}\,r_1)+(s_2,\overline{r_2}\,\overline{t_2}\,r_2)]^*w$ be wf with $t_1 \neq \varepsilon$. Then there is a primitive word p, constants k_1,k_2 such that for all $i_1,\ldots,i_l,j \in \{1,2\}$*

$$us_{i_1}\ldots s_{i_l}s_jw\overline{r_j}\,\overline{t_j}\,r_j \overset{\rho}{=} us_{i_1}\ldots s_{i_l}p^{k_j}w$$

and $L \overset{\rho}{=} u(p^{k_1} + p^{k_2})^* w$ *with* $\mathsf{lcs}^\rho(L) = \mathsf{lcs}^\rho \begin{pmatrix} uw \\ us_1 w\overline{r_1}\,\overline{t_1}\, r_1 \\ us_2 w\overline{r_2}\,\overline{t_2}\, r_2 \end{pmatrix}$.

Proof. Let $R = \mathsf{lcs}^\rho(L)$. We first show that there are $p_i, \hat{p}_i, m_i, n_i$ such that

$$s_i = p_i^{m_i} \quad t_i = \hat{p}_i^{n_i} \quad m_i \geq n_i \quad s_i w\overline{r_i}\,\overline{t_i} \overset{\rho}{=} p_i^{m_i - n_i} w\overline{r_i}$$

If $t_2 = \varepsilon$, then let p_2, m_2 be such that $s_2 = p_2^{m_2}$ and p_2 be primitive; set $n_2 = 0$. Assume thus $t_i \neq \varepsilon$. (Note that $t_1 \neq \varepsilon$ already by assumption of the lemma.) Then $us_i^k w\overline{r_i}\,\overline{t_i}^{\,k}$ is wf for all $k \geq 1$. Therefore $s_i^\omega w_i = t_i^\omega r_i$. Hence by Lemma 1 we find $p_i, \hat{p}_i, m_i, n_i$ such that $s_i = p_i^{m_i}$, $t_i = \hat{p}_i^{n_i}$, $p_i w\overline{r_i}\,\overline{\hat{p}_i} \overset{\rho}{=} w\overline{r_i}$, whereas $m_i \geq n_i$ as L is wf and thus nonnegative. For all $i_1, \ldots, i_l, j \in \{1, 2\}$ we thus have

$$\begin{aligned} us_{i_1} \ldots s_{i_l} s_j w\overline{r_j}\,\overline{t_j}\, r_j &\overset{\rho}{=} us_{i_1} \ldots s_{i_l} (p_j^{m_j - n_j}) w\overline{r_j}\, r_j \\ &\overset{\rho}{=} us_{i_1} \ldots s_{i_l} (p_j^{m_j - n_j}) w \end{aligned}$$

If $m_1 = n_1$ and $m_2 = n_2$ then we obtain for all $i_1, \ldots, i_l \in \{1, 2\}$,

$$us_{i_1} \ldots s_{i_l} w\overline{r_{i_l}}\,\overline{t_{i_l}}\, r_{i_l} \ldots \overline{r_{i_1}}\,\overline{t_{i_1}}\, r_{i_1} \overset{\rho}{=} uw$$

such that $L \overset{\rho}{=} uw$ with $\mathsf{lcs}^\rho(L) = uw$. Hence, p can be chosen as ε.
Thus, assume that $m_1 > n_1$ or $m_2 > n_2$. We then need to show that $p_1 = p_2$. If $s_2 = \varepsilon$, then $t_2 = \varepsilon$ as L is wf and thus nonnegative. Therefore, simply choose p_2 as p_1 and $m_2 = n_2 = 0$. So, assume $s_2 \neq \varepsilon$ such that $p_2 \neq \varepsilon$ and $m_2 > 0$. Then for all $k, l \geq 1$ the following has to be wwf:

$$\begin{aligned} s_1^k s_2^l w\overline{r_2}\,\overline{t_2}^{\,l} r_2 \overline{r_1}\,\overline{t_1}^{\,k} &= p_1^{km_1} p_2^{l(m_2 - n_2)} w\overline{r_2}\, r_2 \overline{r_1}\,\overline{t_1}^{\,k} \\ &\overset{\rho}{=} p_1^{km_1} \underline{p_2^{l(m_2 - n_2)} w\overline{r_1}\,\overline{t_1}^{\,k}} \end{aligned}$$

If $m_2 > n_2$, then $p_2^\omega w = t_1^\omega r_1 = p_1^\omega w$, i.e., $p_1 = p_2$. If $m_2 = n_2 > 0$ and thus $t_2 \neq \varepsilon$ and $m_1 > n_1$. Then for all $k, l \geq 1$ the following has to be wwf:

$$\begin{aligned} s_2^k s_1^l w\overline{r_1}\,\overline{t_1}^{\,l} r_1 \overline{r_2}\,\overline{t_2}^{\,k} &= p_2^{km_2} p_1^{l(m_1 - n_1)} w\overline{r_1}\, r_1 \overline{r_2}\,\overline{t_2}^{\,k} \\ &\overset{\rho}{=} p_2^{km_2} \underline{p_1^{l(m_1 - n_1)} w\overline{r_2}\,\overline{t_2}^{\,k}} \end{aligned}$$

Hence, $p_1^\omega w = t_2^\omega r_2 = p_2^\omega w$, i.e., $p_1 = p_2$.
As $s_1, s_2 \in p^+$ we obtain that $L \overset{\rho}{=} u(p^{m_1 - n_j} + p^{m_2 - n_2})^* w$ and therefore $\mathsf{lcs}^\rho(L) = \mathsf{lcs}^\rho(uw, us_1 w\overline{r_1}\,\overline{t_1}\, r_1, us_2 w\overline{r_2}\,\overline{t_2}\, r_2)$. □

Next, we consider the case that one pumping tree is negative and one is positive. Still, we can show that the language can be normalized to a regular language over A.

Lemma 16. *Let $L = (u,\varepsilon)[(s_1, \overline{r_1}\,\overline{t_1}\,r_1) + (s_2, \overline{r_2}\,t_2 r_2)]^* w$ be wf with $t_1 \neq \varepsilon$. Then there is a primitive period p and constants k_1, k_2 such that for all $i_1, \ldots, i_l \in \{1,2\}$*

1. $us_{i_1} \ldots s_{i_l} s_1 w\overline{r_1}\,\overline{t_1}\,r_1 \overset{\rho}{=} us_{i_1} \ldots s_{i_l} p^{k_1} w$
2. $us_{i_1} \ldots s_{i_l} s_2 w\overline{r_2}\,t_2 r_2 \overset{\rho}{=} us_{i_1} \ldots s_{i_l} p^{k_2} w$
3. $L \overset{\rho}{=} u(p^{k_1} + p^{k_2})^* w$
4. $\mathsf{lcs}^\rho(L) = \mathsf{lcs}^\rho \begin{pmatrix} uw \\ us_1 w\overline{r_1}\,\overline{t_1}\,r_1 \\ us_2 w\overline{r_2}\,\overline{t_2}\,r_2 \end{pmatrix}$

Proof. W.l.o.g. $t_2 \neq \varepsilon$ otherwise we can apply Lemma 15. Let $R = \mathsf{lcs}(L)$. As $t_1 \neq \varepsilon$ and $s_1^k w\overline{r_1}\,\overline{t_1}^{\,k}$ is wwf for all $k \geq 1$ we have $s_1^\omega w \overset{s}{=} t_1^\omega r_1$. Thus by Lemma 1 there exist $p, \hat{p}, m_1, n_1$ such that

$$s_1 = p^{m_1} \quad t_1 = \hat{p}^{n_1} \quad m_1 \geq n_1 \quad s_1 w\overline{r_1}\,\overline{t_1} \overset{\rho}{=} p^{m_1-n_1} w\overline{r_1}$$

W.l.o.g. p and $\hat{p}$ are primitive with $p^\omega w = \hat{p}^\omega r_1$. Thus, for all $i_1, \ldots, i_l \in \{1,2\}$ we have

$$\begin{aligned} us_{i_1} \ldots s_{i_l} s_1 w\overline{r_1}\,\overline{t_1}\,r_1 &= us_{i_1} \ldots s_{i_l} p^{m_1} w\overline{r_1}\,\overline{\hat{p}}^{\,n_1} r_1 \\ &\overset{\rho}{=} us_{i_1} \ldots s_{i_l} p^{m_1-n_1} w\overline{r_1}\,r_1 \\ &\overset{\rho}{=} us_{i_1} \ldots s_{i_l} (p^{m_1-n_1}) w \end{aligned}$$

I.e. the first statement follows.
As $t_2^l r_2 \overline{r_1}\,\bar{t}_1^{\,k} = t_2^l r_2 \overline{r_1}\,\hat{p}^{n_1 \cdot k}$ has to be wwf for all $k, l \geq 1$ it follows that $t_2^\omega r_2 = \hat{p}^\omega r_1$. Lemma 1 thus yields that there are $\check{p}, n_2$ such that

$$t_2 = \check{p}^{n_2} \quad \check{p} r_2 \overline{r_1}\,\overline{\hat{p}} \overset{\rho}{=} r_2 \overline{r_1}$$

Note that $\check{p}$ is primitive as $\hat{p}$ is primitive and $\check{p}^\omega r_2 = \hat{p}^\omega r_1$.
We show that $s_2 = p^{m_2}$ for some $m_2 \in \mathbb{N}_0$. If $s_2 = \varepsilon$ we set $m_2 = 0$. Thus, assume that $s_2 \neq \varepsilon$. Then $r_2 \overset{s}{\sqsubset} s_2^\omega w$ as $s_2^l w\overline{r_2}$ is wf for l sufficiently large. We have that $s_2^l w\overline{r_2}\,t_2^l r_2 \overline{r_1}\,\overline{t_1}^{\,k}$ is wwf for all $k, l \geq 1$. Let $c > n_2$ and l so large such that $r_2 \overset{s}{\sqsubset} s_2^l w$. Then

$$s_2^l w\overline{r_2}\,t_2^l r_2 \overline{r_1}\,\overline{t_1}^{\,cl} = s_2^l w\overline{r_2}\,\check{p}^{n_2 l} r_2 \overline{r_1}\,\overline{\hat{p}}^{\,cln_1} \overset{\rho}{=} s_2^l w\overline{r_2}\,r_2 \overline{r_1}\,\overline{\hat{p}}^{\,(cn_1-n_2)l} \overset{\rho}{=} s_2^l w\overline{r_1}\,\overline{\hat{p}}^{\,(cn_1-n_2)l}$$

Therefore, we obtain $s_2^\omega w \overset{s}{=} \hat{p}^\omega r_1 = p^\omega w$, i.e., $s_2 = p^{m_2}$ for some $m_2 \in \mathbb{N}$. Moreover, as $p^\omega w = \hat{p}^\omega r_1 = \check{p}^\omega r_2$ we obtain that $pw\overline{r_2}\,\overline{\check{p}} \overset{\rho}{=} w\overline{r_2}$.
In the next step, we show that for all $i_1 \ldots i_l \in \{1,2\}, l \geq 0$ we have

$$us_{i_1} \ldots s_{i_l} s_2 w\overline{r_2}\,t_2 r_2 \overset{\rho}{=} us_{i_1} \ldots s_{i_l} (p^{m_2+n_2}) w$$

W.l.o.g. $t_2 \neq \varepsilon$. First, assume that $w \overset{s}{\sqsubset} r_2 = r_2'w$. Then $us_{i_1} \dots s_{i_l} s_2 w\overline{r_2} \overset{\rho}{=} us_{i_1} \dots s_{i_l} s_2 \overline{r_2'}$ is wf for all $i_1 \dots i_l \in \{1,2\}, l \geq 0$. Further $\check{p} r_2 \overline{w}\, \overline{p} \overset{\rho}{=} r_2\overline{w} \overset{\rho}{=} r_2'$ such that

$$t_2 r_2 \overline{w} \overset{\rho}{=} \check{p}^{n_2} r_2 \overline{w}\, \overline{p}^{\,n_2} p^{n_2} \overset{\rho}{=} r_2 \overline{w}\, p^{n_2}$$

Hence, for all $i_1, \dots, i_l \in \{1,2\}$,

$$\begin{array}{rcll}
us_{i_1} \dots s_{i_l} s_2 w\overline{r_2}\, t_2 r_2 & \overset{\rho}{=} & us_{i_1} \dots s_{i_l} s_2 \overline{r_2'}\, r_2' w \overline{r_2}\, t_2 r_2 & r_2 = r_2' w \\
& \overset{\rho}{=} & us_{i_1} \dots s_{i_l} s_2 \overline{r_2'}\, t_2 r_2 \overline{w}\, w & w\overline{r_2} \overset{\rho}{=} \overline{r_2'} \\
& \overset{\rho}{=} & us_{i_1} \dots s_{i_l} s_2 \overline{r_2'}\, r_2' p^{n_2} w & t_2 r_2 \overline{w} \overset{\rho}{=} r_2 \overline{w}\, p^{n_2} \overset{\rho}{=} r_2' p^{n_2} \\
& \overset{\rho}{=} & us_{i_1} \dots s_{i_l} p^{m_2+n_2} w & s_2 = p^{m_2}
\end{array}$$

Second, assume that $r_2 \overset{s}{\sqsubseteq} w = w'r_2$. As $pw\overline{r_2} \overset{\rho}{=} pw\overline{r_2}\, \overline{\check{p}}\, \check{p} \overset{\rho}{=} w\overline{r_2}\, \check{p} \overset{\rho}{=} w'\check{p}$ we obtain $pw' = w'\check{p}$. Thus, for all $i_1, \dots, i_l \in \{1,2\}$,

$$\begin{array}{rcll}
us_{i_1} \dots s_{i_l} s_2 w\overline{r_2}\, t_2 r_2 & \overset{\rho}{=} & us_{i_1} \dots s_{i_l} s_2 w' t_2 r_2 & w = w' r_2 \\
& \overset{\rho}{=} & us_{i_1} \dots s_{i_l} p^{m_2} w' \check{p}^{n_2} t_2 r_2 & s_2 = p^{m_2}, t_2 = \check{p}^{n_2} \\
& \overset{\rho}{=} & us_{i_1} \dots s_{i_l} p^{m_2+n_2} w' r_2 & pw' = w'\check{p} \\
& \overset{\rho}{=} & us_{i_1} \dots s_{i_l} p^{m_2+n_2} w & w = w' r_2
\end{array}$$

Thus, the second statement of the lemma follows and we can conclude that

$$L \overset{\rho}{=} u(p^{m_1-n_1} + p^{m_2+n_2})^* w$$

and therefore $\mathsf{lcs}^\rho(L) = \mathsf{lcs}(uw, upw) = \mathsf{lcs}^\rho(uw, us_1 w\overline{r_1}\, \overline{t_1}\, r_1, us_2 w \overline{r_2}\, t_2 r_2)$. □

Last, we consider the case that both pumping trees (contexts) are positive, i.e., $\tau_i = \overline{r_i}\, t_i r_i$ for $i = 1, 2$. In this case the language $(u, \varepsilon)[(s_1, \overline{r_1}\, t_1 r_1) + (s_2, \overline{r_2}\, t_2 r_2)]^* w$ does not normalize to a regular language over A which makes the proof much more involved. Lemma 18 shows that the lcs and lcsext of the simple linear well-formed language L is already determined by $(u, v)\varepsilon$ and either some word $(u, v)(s_i, \tau_i)\varepsilon$ or some word $(u, v)(s_i, \tau_i)(s_j, \tau_j)\varepsilon$ for some $i \in \{1, 2\}$ with the important point that j can be chosen arbitrarily from $\{1, 2\}$ — this is central to the proof of Lemma 11. We illustrate the idea of the proof of Lemma 18 in the following example.

Example 15. *Consider the well-formed language*

$$L = (abbaba, \varepsilon)[(ba, \overline{aba}\, baba) + (a, bbaba)]^* \varepsilon$$

We have $R := \mathsf{lcs}^\rho(L) = bbaba$ *as we will see in the following. To this end, set* $u := a$*,* $r := aba$*,* $s_1 := ba$*,* $s_2 := a$*,* $t_1 := b$ *so that*

$$L = (uR, \varepsilon)[(s_1, \overline{r}\, t_1 r) + (s_2, R)]^* \varepsilon$$

Note that we have

- $r = aba = r's_1$ *for* $r' := a$
- $r's_1 = a\,ba = ab\,a = \hat{s}_1 r'$ *for* $\hat{s}_1 := ab$
- $r's_2 = a\,a = \hat{s}_2\,r'$ *for* $\hat{s}_2 := s_2 = a$
- $R = t_1^2 r = \mathsf{lcs}^\rho(L)$

In particular, note that $aba = r \overset{s}{\not\sqsubseteq} rs_2 = abaa$*, i.e. there is no conjugate that allows us to move* r *"through"* s_2*; we only find a conjugate* $\hat{s}_2$ *with* $r's_2 = \hat{s}_2 r'$ *(because of well-formedness). By splitting the language depending on the innermost context, we can use these conjugates to rewrite the contexts so that all factors are words over* A*:*

$$\begin{array}{cll}
 & (uR,\varepsilon)[(s_1,\overline{r}\,t_1 r) + (s_2,R)]^*\varepsilon & \\
= & (ut_1^2 r,\varepsilon)[(s_1,\overline{r}\,t_1 r) + (s_2,t_1^2 r)]^*\varepsilon & \\
= & (ut_1^2 r,\varepsilon)\varepsilon & \textit{split w.r.t. innermost} \\
+ & (ut_1^2 r,\varepsilon)[(s_1,\overline{r}\,t_1 r) + (s_2,t_1^2 r)]^*[s_1\overline{r}\,t_1 r + s_2 t_1^2 r] & \textit{context} \\
= & (ut_1^2 r,\varepsilon)\varepsilon & \textit{move } t_1 r \textit{ from innermost} \\
+ & (ut_1^2 r,t_1 r)[(s_1,t_1) + (s_2,t_1 r t_1)]^*[s_1\overline{r} + s_2 t_1] & \textit{context to the right} \\
= & (ut_1^2 r,\varepsilon)\varepsilon & r = r's_1 \\
+ & (ut_1^2 r's_1,t_1 r)[(s_1,t_1) + (s_2,t_1 r t_1)]^*[s_1\overline{r} + s_2 t_1] & \\
= & (ut_1^2 r,\varepsilon)\varepsilon & \textit{move } r' \textit{ to the middle} \\
+ & (ut_1^2 \hat{s}_1,t_1 r)[(\hat{s}_1,t_1) + (\hat{s}_2,t_1 r t_1)]^*[r's_1\overline{r} + r's_2 t_1] & \\
= & (ut_1^2 r,\varepsilon)\varepsilon & r_1 = r_1' s_1 \\
+ & (ut_1^2 \hat{s}_1,t_1 r)[(\hat{s}_1,t_1) + (\hat{s}_2,t_1 r t_1)]^*[\varepsilon + r's_2 t_1] &
\end{array}$$

Note that the derivations w.r.t. the linear grammars underlying both languages are in bijection. We split the rewritten language one last time, this time w.r.t. the outermost context, which leads us to:

$$\begin{array}{rcl}
L & \overset{\rho}{=} & (ut_1^2 r,\varepsilon)\varepsilon \\
 & + & (ut_1^2 \hat{s}_1,t_1 r)\varepsilon \\
 & + & (ut_1^2 \hat{s}_1,t_1 r)r's_2 t_1 \\
 & + & (ut_1^2 \hat{s}_1,t_1 r)(\hat{s}_1,t_1)[(\hat{s}_1,t_1) + (\hat{s}_2,t_1 r\,t_1)]^*[\varepsilon + r's_2 t_1] \\
 & + & (ut_1^2 \hat{s}_1,t_1 r)(\hat{s}_2,t_1 r\,t_1)[(\hat{s}_1,t_1) + (\hat{s}_2,t_1 r\,t_1)]^*[\varepsilon + r's_2 t_1]
\end{array}$$

Substituting the actual values yields

$$\begin{array}{rr} L \overset{\rho}{=} & a\,bbaba \\ + & (abbab, baba)\varepsilon \\ + & (abbab, baba)aab \\ + & (abbab, baba)(ab, b)[(ab, b) + (a, babab)]^*[\varepsilon + aab] \\ + & (abbab, baba)(a, babab)[(ab, b) + (a, babab)]^*[\varepsilon + aab] \\ = & a\,bbaba \\ + & \ldots a\,bbaba \\ + & \ldots a\,bbaba \\ + & \ldots b\,bbaba \\ + & \ldots a\,bbaba \end{array}$$

i.e. only the words included in

$$\begin{array}{rl} & (abbaba, \varepsilon)(ba, \overline{aba}\,baba)[(ba, \overline{aba}\,baba) + (a, bbaba)]^+\varepsilon \\ \overset{\rho}{=} & (abbab, baba)(ab, b)[(ab, b) + (a, babab)]^*[\varepsilon + aab] \\ = & \ldots a\,b\,bbaba \end{array}$$

are witnesses for the longest common suffix w.r.t. the shortest word abbaba. Thus, we need to use in this case at most two contexts where the outermost context has to be $(abbaba, \varepsilon)(ba, \overline{aba}\,baba)$*, while the remaining contexts can be chosen arbitrarily.*

We state one more technical lemma that we will use in the proof of Lemma 18.

Lemma 17. *Let* $L = (r, \varepsilon)(s, t)^+\varepsilon$ *be* wf *with* $r \overset{s}{\sqsubseteq} \mathsf{lcs}(L) \wedge t \neq \varepsilon \wedge rt = \hat{t}r \wedge r \overset{s}{\not\sqsubseteq} rs$*. Then it exist words* $x, y, \tilde{s} \in \mathsf{A}^*$ *such that for all* $k \geq 0$

$$(r, \varepsilon)(s, t)^{k+1} = (x, r)(\tilde{s}, \hat{t})^k y \quad \text{ and } \quad \mathsf{lcs}(L) = \mathsf{lcs}(xy, \hat{t})r = \mathsf{lcs}(st, ts)t \overset{s}{\sqsubset} \hat{t}r$$

Proof. We have $r \neq \varepsilon$ as $r \overset{s}{\not\sqsubseteq} rs$. Additionally it follows that $st \neq ts$ as otherwise there is a primitive p and constants m, n such that $s = p^m$ and $t = p^n$. Then $rp^n = rt = \hat{t}r$ such that $\mathsf{lcs}(r, rp) = \mathsf{lcs}(r, rp^{k+1}) = \mathsf{lcs}(r, rp^n) = r$, i.e., $r \overset{s}{\sqsubseteq} rp$. Therefore it exists conjugate q with $rp = qr$ and $\hat{t} = q^n$. Thus also $r \overset{s}{\sqsubseteq} rs = rp^m = q^m r$ which contradicts the assumption that $r \overset{s}{\not\sqsubseteq} rs$.
First, we consider the case $r \overset{s}{\sqsubset} t = t'r$. Then $t' \neq \varepsilon$ and $\hat{t} = rt'$ as $rt'r = rt = \hat{t}r$. With $rs^{k+1}t^{k+1} = rs^{k+1}(t'r)^{k+1} = rss^k t'\hat{t}^k r$ we obtain

$$L = (rs, r)(s, \hat{t})^* t' = rst'r + (rss, \hat{t}r)(s, \hat{t})^* t'$$

as $rst\overline{r} \overset{\rho}{=} rst'$. Therefore set $x = rs$, $y = t'$ and $\tilde{s} := s$. Then

$$\mathsf{lcs}\left(\begin{array}{r} rst \\ rs^{k+2}t^{k+2} \end{array}\right) = \mathsf{lcs}\left(\begin{array}{r} rst'r \\ rs^{k+2}t^{k}t'rt'r \end{array}\right) = \mathsf{lcs}\left(\begin{array}{r} rs \\ r \end{array}\right) t'r \overset{s}{\sqsubset} rt'r = \hat{t}r$$

and $\mathsf{lcs}\left(\begin{array}{r} rs \\ r \end{array}\right) t'r = \mathsf{lcs}\left(\begin{array}{r} rst' \\ \hat{t} \end{array}\right) r$. Thus,

$$\mathsf{lcs}(L) = \mathsf{lcs}\left(\begin{array}{r} rst \\ rsstt \end{array}\right) = \mathsf{lcs}\left(\begin{array}{r} rst' \\ rsst'rt' \end{array}\right) r = \mathsf{lcs}\left(\begin{array}{r} rst' \\ \hat{t} \end{array}\right) r = \mathsf{lcs}\left(\begin{array}{r} xy \\ \hat{t} \end{array}\right) r \overset{s}{\sqsubset} \hat{t}r$$

As $\mathsf{lcs}\left(\begin{array}{r} rst \\ rsstt \end{array}\right) = \mathsf{lcs}\left(\begin{array}{r} rst \\ rsst'rt \end{array}\right) = \mathsf{lcs}^{\rho}\left(\begin{array}{r} rs \\ r \end{array}\right) t = \mathsf{lcs}\left(\begin{array}{r} t'rs \\ st'r \end{array}\right) t = \mathsf{lcs}\left(\begin{array}{r} ts \\ st \end{array}\right) t$ we obtain $\mathsf{lcs}(st, ts)t = \mathsf{lcs}(L)$.

Second, we consider the case $t \overset{s}{\sqsubseteq} r = r't$. Then $r'tt = rt = \hat{t}r = \hat{t}r't$ and therefore $r't = \hat{t}r'$. As $r't = r \overset{s}{\sqsubseteq} rst = r'tst = \hat{t}r'st$ we obtain $r' \overset{s}{\sqsubseteq} r's$ and therefore we find $\check{s}$ such that $r's = \check{s}r'$. Then

$$rs^{k+1}t^{k+1} = r'ts^{k+1}t^{k+1} = \hat{t}\check{s}\check{s}^{k}\hat{t}^{k}r't = \hat{t}\check{s}\check{s}^{k}\hat{t}^{k}r$$

and $L = (\hat{t}\check{s}, r)(\check{s}, \hat{t})^{*}\varepsilon$. Therefore, we set $x = \hat{t}\check{s}$, $y = \varepsilon$ and $\tilde{s} = \check{s}$. As $\hat{t}r' = r't = r \overset{s}{\not\sqsubseteq} rs = r'ts = \hat{t}\check{s}r'$ we observe that $\hat{t} \overset{s}{\not\sqsubseteq} \hat{t}\check{s}$. Then $\mathsf{lcs}\left(\begin{array}{r} \hat{t}\check{s} \\ \check{s}\hat{t} \end{array}\right) = \mathsf{lcs}\left(\begin{array}{r} \hat{t}\check{s} \\ \hat{t} \end{array}\right) \overset{s}{\sqsubset} \hat{t}$ and

$$\mathsf{lcs}\left(\begin{array}{r} rst \\ rs^{k+2}t^{k+2} \end{array}\right) = \mathsf{lcs}\left(\begin{array}{r} \hat{t}\check{s}r \\ \hat{t}\check{s}^{k+2}\hat{t}^{k}\hat{t}r \end{array}\right) = \mathsf{lcs}\left(\begin{array}{r} \hat{t}\check{s} \\ \hat{t} \end{array}\right) r \overset{s}{\sqsubset} \hat{t}r$$

Therefore, $\mathsf{lcs}(L) = \mathsf{lcs}\left(\begin{array}{r} rst \\ rsstt \end{array}\right) = \mathsf{lcs}\left(\begin{array}{r} \hat{t}\check{s}r \\ \hat{t}\check{s}\check{s}\hat{t}r \end{array}\right) = \mathsf{lcs}\left(\begin{array}{r} \hat{t}\check{s} \\ \hat{t} \end{array}\right) r = \mathsf{lcs}\left(\begin{array}{r} xy \\ \hat{t} \end{array}\right) r \overset{s}{\sqsubset} \hat{t}r = rt$ and as $\mathsf{lcs}\left(\begin{array}{r} rst \\ rsstt \end{array}\right) = \mathsf{lcs}\left(\begin{array}{r} r'tst \\ r'tsstt \end{array}\right) = \mathsf{lcs}\left(\begin{array}{r} ts \\ st \end{array}\right) t$ we obtain $\mathsf{lcs}(st, ts)t = \mathsf{lcs}(L)$. □

Finally, we consider the most involved case of two positive pumping trees.

Lemma 18. *Let $L = (u, \varepsilon)[(s_1, \overline{r_1}\, t_1 r_1) + (s_2, \overline{r_2}\, t_2 r_2)]^{*}w$ be wf.*

Then $\mathsf{lcs}^{\rho}(L) = \mathsf{lcs}^{\rho}\left(\begin{array}{r} uw \\ us_1 w\overline{r_1}\, t_1 r_1 \\ us_2 w\overline{r_2}\, t_2 r_2 \\ us_1 s_1 w\overline{r}\,_1 t_1 t_1 r_1 \\ us_2 s_2 w\overline{r_2}\, t_2 t_2 r_2 \end{array}\right)$

If $r_1, r_2 \overset{s}{\sqsubseteq} \mathsf{lcs}^\rho(L)$*, then further* $\mathsf{lcs}^\rho(L) = \mathsf{lcs}^\rho \begin{pmatrix} uw \\ us_1 w\overline{r_1}\, t_1 r_1 \\ us_2 w\overline{r_2}\, t_2 r_2 \\ us_1 s_2 w\overline{r_2}\, t_2 r_2 \overline{r_1}\, t_1 r_1 \\ us_2 s_1 w\overline{r_1}\, t_1 r_1 \overline{r_2}\, t_2 r_2 \end{pmatrix}$

Proof. W.l.o.g. we assume that $|r_1| \geq |r_2|$ and as $us_2 s_1 w\overline{r_1}\, t_1 r_1 \overline{r_2}\, t_2 r_2 \in L$ is wf it follows that $r_1 \overline{r_2}$ is wf. Let therefore $\boldsymbol{r_2 \overset{s}{\sqsubseteq} r_1 = r_1' r_2}$, i.e.,

$$L = (u, \varepsilon)[(s_1, \overline{r_1' r_2}\, t_1 r_1' r_2) + (s_2, \overline{r_2}\, t_2 r_2)]^* w$$

Then $t_2^k r_2 \overline{r_1} \overset{\rho}{=} t_2^k \overline{r_1'}$ has to be wf for all $k > 0$ and therefore $r_1' \overset{s}{\sqsubset} t_2^m$. Thus, it exists $\hat{t}_2$ such that $\boldsymbol{r_1' t_2 = \hat{t}_2 r_1'}$. Let $R = \mathsf{lcs}^\rho(L)$. If $R \overset{s}{\sqsubset} r_2$ then $R \overset{s}{\sqsubset} r_2 \overset{s}{\sqsubseteq} (u, \varepsilon)[(s_1, \overline{r_1' r_2}\, t_1 r_1' r_2) + (s_2, \overline{r_2}\, t_2 r_2)]^+ w$ and thus $R = \mathsf{lcs}^\rho(L) = \mathsf{lcs}^\rho(uw, us_i w\overline{r_i}\, t_i r_i)$, $i \in \{1, 2\}$.

Therefore assume $\boldsymbol{r_2 \overset{s}{\sqsubseteq} R = \mathsf{lcs}^\rho(L)}$ from here on. If $r_1 \overset{s}{\sqsubseteq} w$ then it exists w' such that $w = w' r_1$. Moving r_1 from w to the end using $r_1' t_2 = \hat{t}_2 r_1$ yields

$$L \overset{\rho}{=} (u, r_1)[(s_1, t_1) + (s_2, \hat{t}_2)]^* w'$$

Thus, we can apply Lemma 14 on $(u, r_1)[(s_1, \varepsilon) + (s_2, \hat{t}_2)]^* w'$ such that we obtain $\mathsf{lcs}^\rho(L) = \mathsf{lcs}^\rho(uw, us_1 w\overline{r_1}\, t_1 r_1, us_2 w\overline{r_2}\, t_2 r_2, us_1 s_1 w\overline{r}\,_1 t_1 t_1 r_1, us_2 s_2 w\overline{r_2}\, t_2 t_2 r_2)$.

Therefore assume $\boldsymbol{w \overset{s}{\sqsubset} r_1 = \dot{r}_1 w}$ with $\dot{r}_1 \neq \varepsilon$ from here on. Consider the case $w \overset{s}{\sqsubset} r_2 = \dot{r}_2 w$. Then $\dot{r}_1 = r_1' \dot{r}_2$ as $\dot{r}_1 w = r_1 = r_1' r_2 = r_1' \dot{r}_2 w$. Furthermore $u = u' \dot{r}_2$ as $\dot{r}_2 w = r_2 \overset{s}{\sqsubseteq} R \overset{s}{\sqsubseteq} uw$. Thus

$$\begin{array}{rcl} L & = & (u, \varepsilon)[(s_1, \overline{r_1' \dot{r}_2 w}\, t_1 r_1' \dot{r}_2 w) + (s_2, \overline{\dot{r}_2 w}\, t_2 \dot{r}_2 w)]^* w \\ & \overset{\rho}{=} & (u' \dot{r}_2, w)[(s_1, \overline{r_1' \dot{r}_2}\, t_1 r_1' \dot{r}_2) + (s_2, \overline{\dot{r}_2}\, t_2 \dot{r}_2)]^* \varepsilon \end{array}$$

As $us_1 w\overline{r_1}\, t_1 r_1 \overset{\rho}{=} us_1 \overline{\dot{r}_2}\, \overline{r_1'}\, t_1 r_1$ and $us_2 w\overline{r_2}\, t_2 r_2 \overset{\rho}{=} us_2 \overline{\dot{r}_2}\, t_2 r_2$ are words of L and thus wf we observe that $us_i \overset{\rho}{=} us_i \overline{\dot{r}_2}\, \dot{r}_2$, $i \in \{1, 2\}$. Moreover, $us_1 s_1 w\overline{r_1}\, t_1 t_1 r_1 \overset{\rho}{=} us_1 \overline{\dot{r}_2}\ \dot{r}_2 s_1 \overline{\dot{r}_2}\ \overline{r_1'}\, t_1 t_1 r_1$ and $us_2 s_2 w\overline{r_2}\, t_2 t_2 r_2 \overset{\rho}{=} us_2 \overline{\dot{r}_2}\ \dot{r}_2 s_2 \overline{\dot{r}_2}\ t_2 t_2 r_2$ are wf and therefore there exist $\hat{s}_i$ such that $\dot{r}_2 s_i = \hat{s}_i \dot{r}_2$. Thus,

$$L \overset{\rho}{=} (u', r_2)[(\hat{s}_1, \overline{r_1'}\, t_1 r_1') + (\hat{s}_2, t_2)]^* \varepsilon$$

and therefore this case is a special case of $r_2 \overset{s}{\sqsubseteq} w$ with $r_2 = w = \varepsilon$.

Assume thus $\boldsymbol{r_2 \overset{s}{\sqsubseteq} w = w' r_2}$ from here on. As $r_2(\overline{r_1}\, t_1 r_1) \overset{\rho}{=} (\overline{r_1'}\, t_1 r_1') r_2$ and $r_2(\overline{r_2}\, t_2 r_1) \overset{\rho}{=} t_2 r_2$ we can move r_2 from $w = w' r_2$ to the end of L such that

$$\begin{array}{rcl} L & = & (u, \varepsilon)[(s_1, \overline{r_1' r_2}\, t_1 r_1' r_2) + (s_2, \overline{r_2}\, t_2 r_2)]^* w' r_2 \\ & \overset{\rho}{=} & (u, r_2)[(s_1, \overline{r_1'}\, t_1 r_1') + (s_2, t_2)]^* w' \end{array}$$

We will therefore assume w.l.o.g. that $\boldsymbol{r_2 = \varepsilon}$ from here on such that $\boldsymbol{w = w'}$, $\boldsymbol{r_1 = r_1'}$, $\boldsymbol{r_1 t_2 = \hat{t}_2 r_1}$ and

$$\begin{array}{rcl} L & = & (u,\varepsilon)[(s_1,\overline{r_1}\,t_1r_1)+(s_2,t_2)]^*w \\ & = & (u,\varepsilon)[(s_1,\overline{\dot{r}_1w}\,t_1\dot{r}_1w)+(s_2,t_2)]^*w \end{array}$$

If $R \overset{s}{\sqsubset} w$ then $w = w'R$ with $w' \neq \varepsilon$ and $r_1 = \dot{r}_1 w = \dot{r}_1 w'R$. Thus,

$$L \overset{\rho}{=} (u,\varepsilon)[(s_1,\overline{\dot{r}_1w'R}\,t_1\dot{r}_1w'R)+(s_2,t_2)]^*w'R$$

As $R \overset{s}{\sqsubseteq} us_2wt_2 = us_2w'Rt_2$ i.e., $R \overset{s}{\sqsubseteq} Rt_2$ it exists $\tilde{t}_2$ such that $Rt_2 = \tilde{t}_2R$. We therefore obtain

$$L \overset{\rho}{=} (u,R)[(s_1,\overline{\dot{r}_1w'}\,t_1\dot{r}_1w')+(s_2,\tilde{t}_2)]^*w'$$

and as $w' \neq \varepsilon$ we have $R = \mathsf{lcs}^\rho(w'R, \tilde{t}_2R)$, i.e., $R = \mathsf{lcs}^\rho(L) = \mathsf{lcs}^\rho(uw, us_2wt_2)$.

Therefore, assume $w \overset{s}{\sqsubseteq} R$ from here on. As $w \overset{s}{\sqsubseteq} R \overset{s}{\sqsubseteq} us_2wt_2$ i.e. $w \overset{s}{\sqsubseteq} wt_2$ it exists $\tilde{t}_2$ such that $wt_2 = \tilde{t}_2w$. Thus

$$L = (u,w)[(s_1,\overline{\dot{r}_1}\,t_1\dot{r}_1)+(s_2,\tilde{t}_2)]^*\varepsilon$$

and we can assume w.l.o.g. that $w = \varepsilon$. Hence $\boldsymbol{u = u'R}$ and $\boldsymbol{r_1 = \dot{r}_1 w = \dot{r}_1}$ such that

$$L = (u'R,\varepsilon)[(s_1,\overline{r_1}\,t_1r_1)+(s_2,t_2)]^*\varepsilon$$

As $us_1\overline{r_1}\,t_1r_1$ is well-formed it follows that $us_1 \overset{\rho}{=} us_1\overline{r_1}\,r_1$. Moreover, we have that $us_1s_1\overline{r_1}\,t_1t_1r_1 \overset{\rho}{=} us_1\overline{r_1}\,r_1s_1\overline{r_1}\,t_1t_1r_1$ is wf. Thus it exists $\hat{s}_1$ such that $\boldsymbol{r_1s_1 = \hat{s}_1r_1}$. As $us_1s_2^lt_2^l\overline{r_1}\,t_1r_1 \overset{\rho}{=} us_1\overline{r_1}\quad r_1s_2^lt_2^l\overline{r_1}\,t_1r_1$ is wf for all l we have that $r_1s_2^lt_2^l\overline{r_1}$ is wf for all l. Consider the case $R \overset{s}{\sqsubset} r_1$, i.e., $r_1 = r_1'R$ with $r_1' \neq \varepsilon$. If we have at least one copy of $(s_1,\overline{r_1}\,t_1r_1)$, then the words end on $r_1 = r_1'R$:

$$\begin{array}{rc} & (u,\varepsilon)[(s_1,\overline{r_1}\,t_1r_1)+(s_2,t_2)]^*(s_1,\overline{r_1}\,t_1r_1)[(s_1,\overline{r_1}\,t_1r_1)+(s_2,t_2)]^*\varepsilon \\ \overset{\rho}{=} & (u,r_1'R)[(s_1,\overline{r_1}\,t_1r_1)+(s_2,t_2)]^*(s_1,\overline{r_1}\,t_1)[(s_1,t_1)+(s_2,\hat{t}_2)]^*\varepsilon \end{array}$$

Thus, $R = \mathsf{lcs}^\rho(u'R, us_1\overline{r_1}\,t_1r_1)$.

Therefore, assume $\boldsymbol{r_1 \overset{s}{\sqsubseteq} R = R'r_1}$ from here on. Then $\boldsymbol{u = u'R'r_1}$ and there exists $\tilde{t}_1$ such that $\boldsymbol{R't_1 = \tilde{t}_1R'}$ as $R = R'r_1 \overset{s}{\sqsubseteq} us_1^k\overline{r_1}\,t_1^kr_1$ for all k. We therefore consider

$$L = (u'R'r_1,\varepsilon)[(s_1,\overline{r_1}\,t_1r_1)+(s_2,t_2)]^*\varepsilon$$

If $r_1 \overset{s}{\sqsubseteq} r_1s_2$ then it exists $\hat{s}_2$ such that $r_1s_2 = \hat{s}_2r_1$ and therefore

$$L = (u'R', r_1)[(\hat{s}_1,t_1)+(\hat{s}_2,\hat{t}_2)]^*\varepsilon$$

We thus can apply Lemma 13.

Therefore, assume $r_1 \overset{s}{\not\sqsubseteq} r_1s_2$ from here on. Thus,

- $\boldsymbol{s_2 t_2 \neq t_2 s_2}$ as otherwise we have $uR' r_1 s_2 s_2 t_2 \overline{r_1}\, t_1 r_1 = uR' \hat{s}_1 r_1 s_2 t_2 \overline{r_1}\, t_1 r_1 = uR' \hat{s}_1 r_1 t_2 s_2 \overline{r_1}\, t_1 r_1 = uR' \hat{s}_1 \hat{t}_2 r_1 s_2 \overline{r_1}\, t_1 r_1$, a contradiction to $r_1 \overset{s}{\not\sqsubseteq} r_1 s_2$.
- there exist $x_2, y_2, \tilde{s}_2$ such that for all $l \geq 0$, $\boldsymbol{r_1 s_2^{l+1} t_2^{l+1} = x_2 \tilde{s}_2^{l} y_2 \hat{t}_2^{l} r_1}$ and $\mathsf{lcs}((r_1, \varepsilon)(s_2, t_2)^{+}\varepsilon) = \mathsf{lcs}(x_2 y_2, \hat{t}_2) r_1 = \mathsf{lcs}(s_2 t_2, t_2 s_2) t_2 \overset{s}{\sqsubset} \hat{t}_2 r_1$ with $|x_2 y_2| = |s_2 t_2| \geq |t_2| = |\hat{t}_2|$, cf. Lemma 17.
- $R = R' r_1 \overset{s}{\sqsubseteq} \mathsf{lcs}((u' R' r_1, \varepsilon)(s_2, t_2)^{+}\varepsilon) = \mathsf{lcs}((x_2, r_1)(\tilde{s}_2, \hat{t}_2)^{*}\varepsilon) \overset{s}{\sqsubset} x_2 y_2 r_1, \hat{t}_2 r_1$ and therefore it exists $\hat{t}'_2$ such that $\boldsymbol{\hat{t}_2 = \hat{t}'_2 R'}$ with $\hat{t}'_2 \neq \varepsilon$ and it exists z_2 such that $\boldsymbol{x_2 y_2 = z_2 R'}$ with $z_2 \neq \varepsilon$.

We partition L into the following components

1. $$\begin{array}{cll} & u & \\ = & u'R & \end{array}$$

2. $$\begin{array}{cll} & (u, \varepsilon)(s_1, \overline{r_1}\, t_1 r_1)^{+}\varepsilon & \\ \overset{\rho}{=} & (u'R', r_1)(\hat{s}_1, t_1)^{+}\varepsilon & R = R' r_1, r_1 s_1 = \hat{s}_1 r_1 \end{array}$$

3. $$\begin{array}{cll} & (u, \varepsilon)(s_1, \overline{r_1}\, t_1 r_1)^{*}(s_2, t_2)\varepsilon & \\ = & (u'R', \varepsilon)(\hat{s}_1, \overline{r_1}\, t_1 r_1)^{*}(r_1 s_2, t_2)\varepsilon & r_1 s_1 = \hat{s}_1 r_1 \\ \overset{\rho}{=} & (u'R', r_1)(\hat{s}_1, t_1)^{*} x_2 y_2 & r_1 s_2 t_2 = x_2 y_2 r_1 \\ = & (u'R', r_1)(\hat{s}_1, t_1)^{*} z_2 R' & x_2 y_2 = z_2 R' \\ = & (u'R', R)(\hat{s}_1, \tilde{t}_1)^{*} z_2 & R' t_1 = \tilde{t}_1 R', R = R' r_1 \end{array}$$

4. $$\begin{array}{cll} & (u, \varepsilon)(s_1, \overline{r_1}\, t_1 r_1)^{*}(s_2, t_2)(s_2, t_2)^{+}\varepsilon & \\ = & (u'R', \varepsilon)(\hat{s}_1, \overline{r_1}\, t_1 r_1)^{*}(r_1 s_2, t_2)(s_2, t_2)^{+}\varepsilon & r_1 s_1 = \hat{s}_1 r_1 \\ = & (u'R', \varepsilon)(\hat{s}_1, \overline{r_1}\, t_1 r_1)^{*}(x_2 \tilde{s}_2, \hat{t}_2 r_1)(\tilde{s}_2, \hat{t}_2)^{*}\varepsilon & r_1 s_2^{l+1} t_2^{l+1} = x_2 \tilde{s}_2^{l} y_2 \hat{t}_2^{l} r_1 \\ \overset{\rho}{=} & (u'R', r_1)(\hat{s}_1, t_1)^{*}(x_2 \tilde{s}_2, \hat{t}'_2 R')(\tilde{s}_2, \hat{t}_2)^{*}\varepsilon & \hat{t}_2 = \hat{t}'_2 R' \\ = & (u'R', R)(\hat{s}_1, \tilde{t}_1)^{*}(x_2 \tilde{s}_2, \hat{t}'_2)(\tilde{s}_2, \hat{t}_2)^{*}\varepsilon & R' t_1 = \tilde{t}_1 R' \end{array}$$

5. $$\begin{array}{cl} & (u, \varepsilon)(s_1, \overline{r_1}\, t_1 r_1)^{*}(s_2, t_2)^{+}(s_1, \overline{r_1}\, t_1 r_1)[(s_1, \overline{r_1}\, t_1 r_1) + (s_2, t_2)]^{*}\varepsilon \\ \overset{\rho}{=} & (u, r_1)(s_1, t_1)^{*}(s_2, \hat{t}_2)(s_2, \hat{t}_2)^{*}(s_1, \overline{r_1}\, t_1)[(s_1, \overline{r_1}\, t_1 r_1) + (s_2, t_2)]^{*}\varepsilon \\ & \text{using } r_1 t_2 = \hat{t}_2 r_1 \\ = & (u, R)(s_1, \tilde{t}_1)^{*}(s_2, \hat{t}'_2)(s_2, \hat{t}_2)^{*}(s_1, \overline{r_1}\, t_1)[(s_1, \overline{r_1}\, t_1 r_1) + (s_2, t_2)]^{*}\varepsilon \\ & \text{using } \hat{t}_2 = \hat{t}'_2 R', R' t_1 = \tilde{t}_1 R', R = R' r_1 \end{array}$$

We consider the case $R' \overset{s}{\sqsubset} t_1$. Then $t_1 = t'_1 R'$ with $t'_1 \neq \varepsilon$ and $\tilde{t}_1 = R' t'_1$ as $R' t'_1 R' = R' t_1 = \tilde{t}_1 R'$. The partition of L thus becomes

1. $u'R$

2. $(u'R', r_1)(\hat{s}_1, t_1)^+\varepsilon = \ldots t'_1R$

3. $(u'R', R)(\hat{s}_1, \tilde{t}_1)^* z_2 = \ldots z_2\tilde{t}_1^*R = \ldots z_2R + \ldots t'_1R$

4. $(u'R', R)(\hat{s}_1, \tilde{t}_1)^*(x_2\tilde{s}_2, \hat{t}'_2)(\tilde{s}_2, \hat{t}_2)^*\varepsilon = \ldots \hat{t}'_2\tilde{t}_1^*R$
   $= \ldots \hat{t}'_2R + \ldots \tilde{t}_1R = \ldots \hat{t}'_2R + \ldots t'_1R$

5. $(u, R)(s_1, \tilde{t}_1)^*(s_2, \hat{t}'_2)(s_2, \hat{t}_2)^*(s_1, \overline{r_1}\,t_1)[(s_1, \overline{r_1}\,t_1r_1) + (s_2, t_2)]^*\varepsilon$
   $= \ldots \hat{t}'_2\tilde{t}_1^*R = \ldots \hat{t}'_2R + \ldots t'_1R$

Hence $R = \mathsf{lcs}(u'R, t'_1R, z_2R, \hat{t}'_2R)$ and therefore

$$\begin{array}{rcl} R & = & \mathsf{lcs}^\rho(u, us_1\overline{r_1}\,t_1r_1, us_2t_2, us_2s_2t_2t_2) \\ & = & \mathsf{lcs}^\rho(u, us_1\overline{r_1}\,t_1r_1, us_2t_2, us_2s_1\overline{r_1}\,t_1r_1t_2) \end{array}$$

Therefore, consider the case $t_1 \overset{s}{\sqsubseteq} R' = R''t_1$ from here on. Then $\boldsymbol{R''t_1 = \tilde{t}_1R''}$ as $R''t_1t_1 = R't_1 = \tilde{t}_1R' = \tilde{t}_1R''t_1$. We observe that $R = R''t_1r_1 \overset{s}{\sqsubseteq} us_1\overline{r_1}\,t_1r_1 = u'R''t_1r_1s_1\overline{r_1}\,t_1r_1 = u'\tilde{t}_1\; R''\hat{s}_1\; t_1r_1$, i.e., $R'' \overset{s}{\sqsubseteq} R''\hat{s}_1$. Therefore it exists $\tilde{s}_1$ such that $\boldsymbol{R''\hat{s}_1 = \tilde{s}_1R''}$. Hence the partition becomes

1. $$\begin{array}{rll} & u & \\ = & u'R & \end{array}$$

2. $$\begin{array}{rll} & (u, \varepsilon)(s_1, \overline{r_1}\,t_1r_1)^+\varepsilon & \\ \overset{\rho}{=} & (u'R''t_1, r_1)(\hat{s}_1, t_1)^+\varepsilon & R' = R''t_1 \\ \overset{\rho}{=} & (u'\tilde{t}_1\tilde{s}_1, R)(\tilde{s}_1, \tilde{t}_1)^*\varepsilon & R''t_1 = \tilde{t}_1R'', R''\hat{s}_1 = \tilde{s}_1R'' \\ = & \ldots \tilde{s}_1R + \ldots \tilde{s}_1\tilde{t}_1R & \end{array}$$

3. $$\begin{array}{rl} & (u, \varepsilon)(s_1, \overline{r_1}\,t_1r_1)^*(s_2, t_2)\varepsilon \\ \overset{\rho}{=} & (u'R', R)(\hat{s}_1, \tilde{t}_1)^*z_2 \\ = & \ldots z_2R + \ldots z_2\tilde{t}_1R \end{array}$$

4. $$\begin{array}{rl} & (u, \varepsilon)(s_1, \overline{r_1}\,t_1r_1)^*(s_2, t_2)(s_2, t_2)^+\varepsilon \\ \overset{\rho}{=} & (u'R', R)(\hat{s}_1, \tilde{t}_1)^*(x_2\tilde{s}_2, \hat{t}'_2)(\tilde{s}_2, \hat{t}_2)^*\varepsilon \\ = & \ldots \hat{t}'_2R + \ldots \hat{t}'_2\tilde{t}_1R \end{array}$$

5. $$\begin{array}{rl} & (u, \varepsilon)(s_1, \overline{r_1}\,t_1r_1)^*(s_2, t_2)^+(s_1, \overline{r_1}\,t_1r_1)[(s_1, \overline{r_1}\,t_1r_1) + (s_2, t_2)]^*\varepsilon \\ \overset{\rho}{=} & (u, R)(s_1, \tilde{t}_1)^*(s_2, \hat{t}'_2)(s_2, \hat{t}_2)^*(s_1, \overline{r_1}\,t_1)[(s_1, \overline{r_1}\,t_1r_1) + (s_2, t_2)]^*\varepsilon \\ = & \ldots \hat{t}'_2R + \ldots \hat{t}'_2\tilde{t}_1R \end{array}$$

If $\tilde{t}_1 \neq \varepsilon$ then

$$\begin{aligned} R &= \mathsf{lcs}(u'R, \tilde{s}_1R, \tilde{t}_1R, z_2R, \hat{t}'_2R) \\ &= \mathsf{lcs}^\rho(u, us_1\overline{r_1}\,t_1r_1, us_1s_1\overline{r_1}\,t_1t_1r_1, us_2t_2, us_2s_2t_2t_2) \\ &= \mathsf{lcs}^\rho(u, us_1\overline{r_1}\,t_1r_1, us_1s_2t_2\overline{r_1}\,t_1r_1, us_2t_2, us_2s_1\overline{r_1}\,t_1r_1t_2) \end{aligned}$$

Otherwise, if $\tilde{t}_1 = \varepsilon$ then

$$\begin{aligned} R &= \mathsf{lcs}(u'R, \tilde{s}_1R, z_2R, \hat{t}'_2R) \\ &= \mathsf{lcs}^\rho(u, us_1\overline{r_1}\,t_1r_1, us_2t_2, us_2s_2t_2t_2) \\ &= \mathsf{lcs}^\rho(u, us_1\overline{r_1}\,t_1r_1, us_2t_2, us_2s_1\overline{r_1}\,t_1r_1t_2) \end{aligned}$$

□

2.5 Summary

In this chapter we showed that well-formedness of a context-free grammar G is decidable in polynomial time. The decision problem was reduced to the computation of the *reduced* longest common suffix of G.

Section 2.1 analysed basic properties of well-formed (context-free) languages and introduced useful techniques for the computation of the longest common suffix. We showed that a nonterminal X of a well-formed context-free grammar produces not necessarily a well-formed language but a bounded well-formed language. Thus, there is a minimal word r_X such that r_XL_X is well-formed. We showed that for a context-free grammar G we can check that every nonterminal X produces a bounded well-formed language and if so compute an SLP representing r_X by only considering derivation trees up to height N with N the number of nonterminals in G. Then, the grammar G can be equivalently rewritten such that every nonterminal X produces the language r_XL_X. For this grammar the longest common suffix of each nonterminal is computed to check that for all left contexts α of X, i.e., $G \to^* \alpha X \beta$, r_X is in fact a suffix of $\rho(\alpha)$. To show that the reduced longest common suffix of a context-free grammar can be computed in polynomial time we needed two ingredients. First, we presented in Section 2.2 a polynomial size representation for a language L to compute the longest common suffix – the lcs-summary. The lcs-summary consists of the longest common suffix of L and the maximal suffix extension lcsext. The word lcsext is the longest word by that the longest common suffix of a language can be extended if another language is concatenated from the left. Second, we show in Section 2.3 that the fixed-point iteration to compute the reduced longest common suffix of a context-free grammar G terminates after at most $4N + 1$ rounds in both cases, if G is well-formed as well as if G is *not* well-formed. Again, N is the number of nonterminals in G. For that we proved that in both cases the lcs-summary converges after $4N + 1$ iterations or a counter-example showing the non well-formedness is found. The results for convergence are in both cases based on an analyses of the reduced longest common suffix of simple linear well-formed languages of the form $\alpha\sigma_1 \dots \sigma_k \gamma \tau_k \dots \tau_1 \beta$. In Section 2.4 we showed that for $L = (\alpha, \beta)[(\sigma_1, \tau_1) + (\sigma_2, \tau_2)]^* \gamma$ the reduced longest common suffix is determined by $\alpha\gamma\beta$ and either a word $(\alpha, \beta)(\sigma_i, \tau_i)\gamma$ or some word $(\alpha, \beta)(\sigma_i, \tau_i)(\sigma_j, \tau_j)\gamma$ with $i, j \in \{1, 2\}$ where j can be independently chosen from i.

Notably, the results of Section 2.4 subsume our polynomial time algorithm to compute the longest common prefix of a context-free grammar presented in [LPS18] and provide easier and more direct proofs. However, it remains as an open question whether well-formed context-free grammars can be rewritten to an equivalent context-free grammar without inverted letters in polynomial time. This would have allowed to use the results in [LPS18] to compute the reduced longest common suffix of well-formed context-free grammars.

3 Equivalence of Linear Tree Transducers

Tree transducers are a fundamental model in formal language theory and early results date back to the 1970s and 1980s where equivalence of bottom-up [Zac79] and top-down [Ési80] tree-to-tree transducers was shown to be decidable. Nowadays, tree-to-tree transductions are well understood as the output is constructed in a *structured* way [Ési80, FV98, EM99, Fri11]. In contrast, in case of tree-to-word transducers the output is not necessarily produced in a structured way and decidability of equivalence was a long standing open problem but solved in 2015 by Seidl, Maneth and Kemper [SMK15]. Due to the complexity of this problem different restricted classes of tree-to-word transducers have been studied. Staworko et al. [SLLN09] provided a polynomial time procedure for deciding equivalence of *sequential* tree-to-word transducers, i.e., transductions where the subtrees are always processed from left to right as given in the input (order-preserving) and not copied. For that they reduced the equivalence problem to the morphism equivalence problem for context-free languages that was shown to be decidable in polynomial time by Plandowski [Pla94]. Some years later, a canonical normal form for sequential tree-to-word transducers was provided that can be applied to learning [LLN$^+$11, LLN$^+$14]. Boiret showed that a normal form can be obtained for *linear* tree-to-word transducers, i.e., transductions that are non-copying but not necessarily order-preserving [Boi16]. However, the construction of this normal form may require exponential time. We improved on that by only normalizing the order in which the subtrees are processed and obtained a *polynomial* time decision procedure for equivalence of linear tree-to-word transducers [BP16]. We further extend this result by showing that the approach can be applied to linear tree transducers with outputs in the free group. Thus, output words are considered as equivalent not just when they are literally equal, but also when they become equal after cancellation of matching opening and closing occurrences of letters. In fact, the techniques used in [SMK18] to obtain the equivalence of general tree-to-word transducers are applicable in the free group. But in the case of linear tree transducers with outputs either in a free monoid or a free group a *randomized* polynomial time procedure to decide equivalence is given. We show that equivalence of linear tree transducers with outputs in a free group can be decided in polynomial time *without* randomization. Clearly, this result includes the case of outputs in a free monoid from 2016 [BP16]. The underlying idea of the procedures is the same. However, the construction as well as the arguments for its correctness, are not only more general but also much cleaner than in the case with outputs in a free monoid only. The observation that reasoning over the free group may simplify arguments has also been made, e.g., by Plandowski proving the existence of polynomial size test sets for context-free languages in free groups [Pla94] and thereby improving the previous result in free monoids [KPR92]. In his case he could shorten the proof of the main lemma from 10 to 2 pages. Similar to Plandowski's techniques we considered advanced properties of the periodicity of words for our proofs in a free monoid. The existence of inverses provided in a free group shorten these proofs a lot, cf. Section 3.1. We show that equivalence of *same-ordered* linear tree transducers, i.e., transducers that process their subtrees in the same order, can also be reduced to the morphism equivalence problem for context-free languages. Therefore, equivalence of linear tree transducers that are same-ordered is decidable in polynomial time. We characterize the languages produced by the states in equivalent transductions that process their subtrees in different orders. These languages have a common underlying periodicity and can therefore be with some adaptations arbitrarily permuted. With these observations we define an order in which subtrees have to be processed and normalize the transducers accordingly. Then equivalent transducers have to be same-ordered and thus equivalence can be tested in polynomial time using Plandowski's result on morphism equivalence on context-free grammars in the free group [Pla94].

Additionally, we recap in Section 3.3 the normalization results from [BP16] for linear tree transducers with output in the free monoid, i.e., tree-to-*word* transducers. Even if the equivalence result in this case is covered by the results with outputs in the free group (Section 3.2), the normalization procedure may introduce inverses. We therefore give a normalization algorithm for tree-to-word transducers such that the normalized (ordered) transducer is still a tree-to-word transducer. Thus, the rules do not contain any inverses as it may be the case for the ordered transducer over the free group – even if the transducer *before* the normalization was a tree-to-word transducer. In fact the construction for ordered tree-to-word transducers is more involved and moreover may lead to a (polynomial) size increase of the transducer. Note that compared to [BP16] the notation and proofs are adjusted and now based on the results over the free group presented in Section 3.2.

Outline In Section 3.1 the technical results for the characterization of the output languages occurring in equivalent transductions that process their inputs in different orders is given. Then the reduction from the equivalence of same-ordered linear tree transducers to the morphism equivalence problem is given in Section 3.2. Further, we define ordered linear tree transducers and show how any linear tree transducer can be ordered in polynomial time to obtain a polynomial decision procedure for equivalence of linear tree transducers with outputs in the free group. This result subsumes the equivalence result from 2016 [BP16]. We reformulate the results from [BP16] in Section 3.3 as the procedure to build an ordered linear tree-to-word transducer is much more involved since no inverses as in the free group can be used.

3.1 Preliminaries

Free Group. We use A to denote an *unranked* alphabet. The free monoid generated by A is denoted by $\langle \mathsf{A}^*, \cdot, \varepsilon\rangle$ with A^* the set of all finite words over A, $\cdot$ the concatenation of finite words, and ε the empty word. As usual, we simply write uv for $u \cdot v$ with $u, v \in \mathsf{A}^*$. From A we derive the alphabet of inverse elements $\mathsf{A}^{-1} = \{a^{-1} \mid a \in \mathsf{A}\}$ and let ${}^{-1}\colon (\mathsf{A} \cup \mathsf{A}^{-1})^* \to (\mathsf{A} \cup \mathsf{A}^{-1})^*$ be the canonical involution on $(\mathsf{A} \cup \mathsf{A}^{-1})^*$ such that $(uv)^{-1} = v^{-1}u^{-1}$ and $(u^{-1})^{-1} = u$ and $\varepsilon^{-1} = \varepsilon$ for $u, v \in (\mathsf{A} \cup \mathsf{A}^{-1})^*$. Further, let $=$ denote the least equivalence relation on $(\mathsf{A} \cup \mathsf{A}^{-1})^*$ for which $uv = uxx^{-1}v$ holds true for any $u, x, v \in (\mathsf{A} \cup \mathsf{A}^{-1})^*$. Then the free group $\mathcal{F}_{\mathsf{A}}$ generated by A is the quotient of the free monoid generated by A modulo $=$. A word $w \in (\mathsf{A} \cup \mathsf{A}^{-1})^*$ is *reduced* if it does not include any subword of the form aa^{-1} for $a \in (\mathsf{A} \cup \mathsf{A}^{-1})$. Every equivalence class w.r.t. $=$ contains exactly one reduced word. If not noted otherwise, we will assume that all words are reduced. We will always omit $\cdot$ for the concatenation of words over A^* but use $\cdot$ as group operation over reduced words of $\mathcal{F}_{\mathsf{A}}$, i.e., $\cdot$ is concatenation, followed by *reduction*, i.e., repeated cancellation of subwords $a\,a^{-1}$ or $a^{-1}a$. Thus, $a\,b\,c^{-1} \cdot c\,b^{-1}\,a = a\,a$. The inverse w^{-1} of some element $w \in \mathcal{F}_{\mathsf{A}}$ is obtained by reverting the order of the letters in w while replacing each letter a with a^{-1} and each a^{-1} with a. Thus, e.g., $(a\,b\,c^{-1})^{-1} = c\,b^{-1}a^{-1}$. In light of the inverse operation $(\,.\,)^{-1}$, we have that $v \cdot w = v'w'$ where $v = v'u$ (as words) for a maximal suffix u so that u^{-1} is a prefix of w with $w = u^{-1}w'$. For an element $w \in \mathcal{F}_{\mathsf{A}}$, $\langle w\rangle = \{w^l \mid l \in \mathbb{Z}\}$ denotes the cyclic subgroup of $\mathcal{F}_{\mathsf{A}}$ generated from w. As usual, we use the convention that $w^0 = \varepsilon$, and $w^{-l} = (w^{-1})^l$ for $l > 0$. An element $p \in \mathcal{F}_{\mathsf{A}}$ different from ε, is called *primitive* if $w^l = p$ for some $w \in \mathcal{F}_{\mathsf{A}}$ and $l \in \mathbb{Z}$ implies that $w = p$ or $w = p^{-1}$, i.e., p and p^{-1} are the only (trivial) roots of p. For example, ab and $(ab)^{-1} = b^{-1}a^{-1}$ are primitive; while $abba^{-1}$ is not primitive as $(aba^{-1})^2 = aba^{-1} \cdot aba^{-1} = abba^{-1}$. Thus, primitive elements generate *maximal* cyclic subgroups of $\mathcal{F}_{\mathsf{A}}$. We state two crucial technical lemmas.

Lemma 19. *Assume that $y^m = \beta \cdot y^n \cdot \beta^{-1}$ with $y \in \mathcal{F}_{\mathsf{A}}$ primitive. Then $m = n$, and $\beta = y^k$ for some $k \in \mathbb{Z}$.*

Proof. Since $\beta \cdot y^n \cdot \beta^{-1} = (\beta \cdot y \cdot \beta^{-1})^n$, we find by [LS15, Proposition 2.17] a primitive element c such that y and $\beta \cdot y \cdot \beta^{-1}$ are powers of c. As y is primitive, c

can be chosen as y. Accordingly,

$$y^j = \beta \cdot y \cdot \beta^{-1} \tag{3.1}$$

holds for some j. If β is a power of y, then $\beta \cdot y \cdot \beta^{-1} = y$, and the assertion of the lemma holds. Likewise if $j = 1$, then β and y commute. Since y is primitive, then β necessarily must be a power of y.

For a contradiction, therefore now assume that β is not a power of y and $j \neq 1$. W.l.o.g., we can assume that $j > 1$. First, assume now that y is *cyclically reduced* i.e., the first and last letter, a and b, respectively, of y are not mutually inverse ($a^{-1} \neq b$). Then for each $n > 0$, y^n is obtained from y by n-concatenation of y as a word (no reduction taking place). Likewise, either the last letter of β is different a^{-1} or the first letter of β^{-1} is different from b^{-1} because these two letters are mutually inverse. Assume that the former is the case. Then $\beta \cdot y$ is obtained by concatenation of β and y as words (no reduction taking place). By (3.1), $\beta \cdot y^n = y^{j \cdot n} \cdot \beta$. for every $n \geq 1$. Let $m > 0$ denote the length of β as a word. Since β can cancel only a suffix of $y^{j \cdot n}$ of length at most m, it follows, that the word $\beta\, y$ must be a prefix of the word y^{m+1}. Since β is not a power of y, the word y can be factored into $y = y'c$ for some non-empty suffix c such that $\beta = y^{j'} y'$, implying that $y'c = cy'$ and thus $yc = cy$ hold. As a consequence, $y = p^l$ for some $p \in \mathcal{F}_{\mathsf{A}}$ and $l > 1$ — in contradiction to the irreducibility of y.

If on the other hand, the first letter of β^{-1} is not the inverse of the last letter of y, then $y \cdot \beta^{-1}$ is obtained as the concatenation of y and β^{-1} as words. As a consequence, $y\beta^{-1}$ is a suffix of y^{m+1}, and we arrive at a contradiction.

We conclude that the statement of the lemma holds whenever y is cyclically reduced. Now assume that y is not yet cyclically reduced. Then we can find a maximal suffix r of y (considered as a word) such that $y = r^{-1} s r$ holds and s is cyclically reduced. Then s is also necessarily primitive. (If $s = c^n$, then $y = (r^{-1} c r)^n$). Then assertion (3.1) can be equivalently formulated as

$$s^j = (r \cdot \beta \cdot r^{-1}) \cdot y \cdot (r \cdot \beta \cdot r^{-1})^{-1}$$

We conclude that $r \cdot \beta \cdot r^{-1} = s^l$ for some $l \in \mathbb{Z}$. But then $\beta = (r^{-1} \cdot s \cdot r)^l = y^l$, and the claim of the lemma follows. □

Lemma 20. *Assume that x_1, x_2 and y_1, y_2 are distinct elements in $\mathcal{F}_{\mathsf{A}}$ and that*

$$x_i \cdot \alpha \cdot y_j \cdot \beta = \gamma \cdot y'_j \cdot \alpha' \cdot x'_i \cdot \beta' \tag{3.2}$$

holds for $i = 1, 2$ and $j = 1, 2$. Then there is some primitive element p and exponents $r, s \in \mathbb{Z}$ such that $x_1 \cdot \alpha = x_2 \cdot \alpha \cdot p^r$ and $y_1 = p^s \cdot y_2$.

Proof. By the assumption (3.2),

$$\begin{aligned}\gamma &= (x_1 \cdot \alpha \cdot y_j \cdot \beta) \cdot (y'_j \cdot \alpha' \cdot x'_1 \cdot \beta')^{-1} \\ &= (x_2 \cdot \alpha \cdot y_j \cdot \beta) \cdot (y'_j \cdot \alpha' \cdot x'_2 \cdot \beta')^{-1}\end{aligned}$$

for all $j = 1, 2$. Thus,

$$\begin{aligned}x_1 \cdot \alpha \cdot y_j \cdot \beta\beta'^{-1} x'^{-1}_1 \alpha'^{-1} y'^{-1}_j &= x_2 \cdot \alpha \cdot y_j \cdot \beta \cdot \beta'^{-1} \cdot x'^{-1}_2 \cdot \alpha'^{-1} \cdot y'^{-1}_j \quad \text{implying} \\ y_j^{-1} \cdot \alpha^{-1} \cdot x_2^{-1} \cdot x_1 \cdot \alpha \cdot y_j &= \beta \cdot \beta'^{-1} \cdot x'^{-1}_2 \cdot x'_1 \cdot \beta' \cdot \beta^{-1}\end{aligned}$$

for $j = 1, 2$. Hence,

$$\begin{aligned}y_1^{-1} \cdot \alpha^{-1} \cdot x_2^{-1} \cdot x_1 \cdot \alpha \cdot y_1 &= y_2^{-1} \cdot \alpha^{-1} \cdot x_2^{-1} \cdot x_1 \cdot \alpha \cdot y_2 \quad \text{implying} \\ (x_2 \cdot \alpha)^{-1} x_1 \cdot \alpha &= (y_1 \cdot y_2^{-1}) \cdot ((x_2 \cdot \alpha)^{-1} \cdot x_1 \cdot \alpha) \cdot (y_1 \cdot y_2^{-1})^{-1}\end{aligned}$$

Since x_1 is different from x_2, also $x_1 \cdot \alpha$ is different from $x_2 \cdot \alpha$. Let p denote a primitive root of $(x_2 \cdot \alpha)^{-1} \cdot x_1 \cdot \alpha$. Then by Lemma 19,

$$\begin{aligned}x_1 \cdot \alpha &= x_2 \cdot \alpha \cdot p^r \\ y_1 &= p^s \cdot y_2\end{aligned}$$

for suitable exponents $r, s \in \mathbb{Z}$. □

As the elements of $\mathcal{F}_{\mathsf{A}}$ are words, they can be represented by *straight-line* programs (SLPs). An SLP is a context-free grammar where each non-terminal occurs as the left-hand side of exactly one rule. We briefly recall basic complexity results for operations on elements of $\mathcal{F}_{\mathsf{A}}$ when represented as SLPs [Loh14].

Lemma 21. *Let U, V be SLPs representing words $w_1, w_2 \in \{a, a^{-1} \mid a \in \mathsf{A}\}$, respectively. Then the following computations and decision problems can be realized in polynomial time*

- *compute an SLP for w_1^{-};*
- *compute the primitive root of w_1 if $w_1 \neq \varepsilon$;*
- *compute an SLP for $w = w_1$ with w reduced;*
- *decide whether $w_1 = w_2$;*
- *decide whether it exists $g \in \mathcal{F}_{\mathsf{A}}$, such that $w_1 \in g \cdot \langle w_2 \rangle$ and compute an SLP for such g.*

Linear Tree Transducer. In the following, we introduce *deterministic linear tree transducers* which produce outputs in the free group $\mathcal{F}_{\mathsf{A}}$. For convenience, we follow the approach in [SMK18] where only *total* deterministic transducers are considered — but equivalence is relative w.r.t. some *top-down deterministic* domain automaton D. We use Σ to denote a finite *ranked* alphabet. $\mathcal{T}_\Sigma$ denotes the set of all trees (or terms) over Σ. The *depth* $\mathsf{depth}(t)$ of a tree $t \in \mathcal{T}_\Sigma$ equals 0, if $t = f()$ for some $f \in \Sigma$ of rank 0, and otherwise, $\mathsf{depth}(t) = 1 + \max\{\mathsf{depth}(t_i) \mid i = 1, \ldots, m\}$ for $t = f(t_1, \ldots, t_m)$. A top-down deterministic automaton (DTA) D is a tuple $(H, \Sigma, \delta_D, h_0)$ where H is a finite set of states, Σ is a finite ranked alphabet, $\delta_D : H \times \Sigma \to H^*$ is a partial function where $\delta_D(h, f) \in H^k$ if the rank of f equals k, and h_0 is the start state of D. For every $h \in H$, we define the set $\mathsf{dom}(h) \subseteq \mathcal{T}_\Sigma$ by $f(t_1, \ldots, t_m) \in \mathsf{dom}(h)$ iff $\delta_D(h, f) = h_1 \ldots h_m$ and $t_i \in \mathsf{dom}(h_i)$ for all $i = 1, \ldots, k$. D is called *reduced* if $\mathsf{dom}(h) \neq \emptyset$ for all $h \in H$. The *language* $\mathcal{L}(D)$ accepted by D is the set $\mathsf{dom}(h_0)$. We remark that for every DTA D with $\mathcal{L}(D) \neq \emptyset$, a reduced DTA D' can be constructed in polynomial time with $\mathcal{L}(D) = \mathcal{L}(D')$. Therefore, we subsequently assume w.l.o.g. that each DTA D is reduced.

A (total deterministic) *linear tree transducer* with output in $\mathcal{F}_\mathsf{A}$ (LT for short) is a tuple $M = (\Sigma, \mathsf{A}, Q, S, R)$ where Σ is the ranked alphabet for the input trees, A is the finite (unranked) output alphabet, Q is the set of *states*, S is the *axiom* of the form u_0 or $u_0 \cdot q_0(x_0) \cdot u_1$ with $u_0, u_1 \in \mathcal{F}_\mathsf{A}$ and $q_0 \in Q$, and R is the set of *rules* which contains for each state $q \in Q$ and each input symbol $f \in \Sigma$, one rule of the form

$$q(f(x_1, \ldots, x_m)) \to u_0 \cdot q_1(x_{\sigma(1)}) \cdot \ldots \cdot u_{n-1} \cdot q_n(x_{\sigma(n)}) \cdot u_n \tag{3.3}$$

Here, m is the rank of f, $n \leq m$, $u_0, \ldots, u_n \in \mathcal{F}_\mathsf{A}$ and σ is an injective mapping from $\{1, \ldots, n\}$ to $\{1, \ldots, m\}$. The *semantics* of a state q is the function $[\![q]\!] : \mathcal{T}_\Sigma \to \mathcal{F}_\mathsf{A}$ defined by

$$[\![q]\!](f(t_1, \ldots, t_m)) = u_0 \cdot [\![q_1]\!](t_{\sigma(1)}) \cdot \ldots \cdot u_{n-1} \cdot [\![q_n]\!](t_{\sigma(n)}) \cdot u_n$$

if there is a rule of the form (3.3) in R. Then the *translation* of M is the function $[\![M]\!] : \mathcal{T}_\Sigma \to \mathcal{F}_\mathsf{A}$ defined by $[\![M]\!](t) = u_0$ if the axiom of M equals u_0, and $[\![M]\!](t) = u_0 \cdot [\![q]\!](t) \cdot u_1$ if the axiom of M is given by $u_0 \cdot q(x_0) \cdot u_1$.

Example 16. *Let $\mathsf{A} = \{a, b\}$. As a running example we consider the LT M with input alphabet $\Sigma = \{f^2, g^1, k^0\}$ where the superscripts indicate the rank of the input symbols. M has axiom $q_0(x_0)$ and the following rules*

$$\begin{array}{lll}
q_0(f(x_1, x_2)) \to q_1(x_2) b q_2(x_1) & q_0(g(x_1)) \to q_0(x_1) & q_0(k) \to \varepsilon \\
q_1(f(x_1, x_2)) \to q_0(x_1) q_0(x_2) & q_1(g(x_1)) \to ab q_1(x_1) & q_1(k) \to a \\
q_2(f(x_1, x_2)) \to q_0(x_1) q_0(x_2) & q_2(g(x_1)) \to ab q_2(x_1) & q_2(k) \to ab
\end{array}$$

Two LTs M, M' are *equivalent* relative to the DTA D iff their translations coincide on all input trees accepted by D, i.e., $[\![M]\!](t) = [\![M']\!](t)$ for all $t \in \mathcal{L}(D)$.

To relate the computations of the LT M and the domain automaton D, we introduce the following notion. A mapping $\iota : Q \to H$ from the set of states of M to the set of states of D is called *compatible* if either the set of states of M is empty (and thus the axiom of M consists of an element of $\mathcal{F}_\mathsf{A}$ only), or the following holds:

1. $\iota(q_0) = h_0$;
2. If $\iota(q) = h$, $\delta_D(h, f) = h_1 \dots h_m$, and there is a rule in M of the form (3.3) then $\iota(q_i) = h_{\sigma(i)}$ for all $i = 1, \dots, n$;
3. If $\iota(q) = h$ and $\delta_D(h, f)$ is undefined for some $f \in \Sigma$ of rank $m \geq 0$, then M has the rule $q(f(x_1, \dots, x_m)) \to \bot$ for some dedicated symbol $\bot$ which does not belong to A.

Lemma 22. *For an* LT *M and a* DTA *$D = (H, \Sigma, \delta_D, h_0)$, an* LT *$M'$ with a set of states Q' together with a mapping $\iota : Q' \to H$ can be constructed in polynomial time such that the following holds:*

1. M and M' are equivalent relative to D;

2. ι is compatible.

Proof. In case that the axiom of M is in $\mathcal{F}_{\mathsf{A}}$, we obtain M' from M using the axiom of M and using empty sets of states and rules, respectively. Assume therefore that the axiom of M is of the form $u_0 \cdot q_0(x_0) \cdot u_1$. Then LT M' is constructed as follows. The set Q' of states of M' consists of pairs $\langle q, h\rangle$, $q \in Q, h \in H$ where $\iota(\langle q, h\rangle) = h$. In particular, $\langle q_0, h_0\rangle \in Q'$. As the axiom of M' we then use $u_0 \cdot \langle q_0, h_0\rangle(x_0) \cdot u_1$. For a state $\langle q, h\rangle \in Q'$, consider each input symbol $f \in \Sigma$. Let $m \geq 0$ denote the rank of f. If $\delta_D(h, f)$ is not defined, M' has the rule

$$\langle q, h\rangle(f(x_1, \dots, x_m)) \to \bot$$

Otherwise, let $\delta_D(h, f) = h_1 \dots h_m$, and assume that M has a rule of the form (3.3). Then we add the states $\langle q_i, h_{\sigma(i)}\rangle$ to Q' together with the rule

$$\langle q, h\rangle(f(x_1, \dots, x_m)) \to u_0 \cdot \langle q_1, h_{\sigma(1)}\rangle(x_{\sigma(1)}) \cdot \dots \cdot u_{n-1} \cdot \langle q_n, h_{\sigma(n)}\rangle(x_{\sigma(n)}) \cdot u_n$$

By construction, the mapping ι is compatible. We verify for each state $\langle q, h\rangle$ of M' and each input tree $t \in \mathsf{dom}(h)$ that $[\![q]\!](t) = [\![\langle q, h\rangle]\!](t)$ holds. This proof is by induction on the structure of t. From that, the equivalence of M and M' relative to D follows. □

Example 17. *Let* LT *M be defined as in Example 16. Consider* DTA *D with start state h_0 and the transition function $\delta_D = \{(h_0, f) \mapsto h_1 h_1, (h_1, g) \mapsto h_1, (h_1, h) \to \varepsilon\}$. According to Lemma 22,* LT *M' for M then is defined as follows. M' has axiom $\langle q_0, h_0\rangle(x_0)$ and the rules*

$$\begin{array}{ll} \langle q_0, h_0\rangle(f(x_1, x_2)) \to \langle q_1, h_1\rangle(x_2)\, b\, \langle q_2, h_1\rangle(x_1) & \\ \langle q_1, h_1\rangle(g(x_1)) \to ab\, \langle q_1, h_1\rangle(x_1) & \langle q_1, h_1\rangle(k) \to a \\ \langle q_2, h_1\rangle(g(x_1)) \to ab\, \langle q_2, h_1\rangle(x_1) & \langle q_2, h_1\rangle(k) \to ab \end{array}$$

The rules that have left-hand sides $\langle q_0, h_0\rangle(g(x_1))$, $\langle q_0, h_0\rangle(h)$, $\langle q_1, h_1\rangle(f(x_1, x_2))$, $\langle q_2, h_1\rangle(f(x_1, x_2))$, all have right-hand-sides $\perp$. The compatible map ι is then given by $\iota = \{\langle q_0, h_0\rangle \mapsto h_0, \langle q_1, h_1\rangle \mapsto h_1, \langle q_2, h_1\rangle \mapsto h_1\}$. For convenience, we denote the pairs $\langle q_0, h_0\rangle$, $\langle q_1, h_1\rangle$, $\langle q_2, h_1\rangle$ with q_0, q_1, q_2, respectively.

Subsequently, we w.l.o.g. assume that each LT M with corresponding DTA D for its domain, comes with a compatible map ι. Moreover, we define for each state q of M, the set $\mathcal{L}(q) = \{[\![q]\!](t) \mid t \in \mathsf{dom}(\iota(q))\}$ of all outputs produced by state q (on inputs in $\mathsf{dom}(\iota(q))$), and $\mathcal{L}^{(i)}(q) = \{[\![q]\!](t) \mid t \in \mathsf{dom}(\iota(q)), \mathsf{depth}(t) < i\}$ for $i \geq 0$. Beyond the availability of a compatible map, we also require that all states of M are *non-trivial* (relative to D). Here, a state q of M is called *trivial* if $\mathcal{L}(q)$ contains a single element only. Otherwise, it is called *non-trivial*. This property will be established in Theorem 2.

3.2 Deciding Equivalence

In the first step, we show that equivalence relative to the DTA D of *same-ordered* LTs is decidable. For a DTA D, consider the LTs M and M' with compatible mappings ι and ι', respectively. M and M' are *same-ordered* relative to D if they process their input trees in the same order. We define set of pairs $\langle q, q'\rangle$ of *co-reachable* states of M and M'. Let $u_0 \cdot q_0(x_1) \cdot u_1$ and $u'_0 \cdot q'_0(x_1) \cdot u'_1$ be the axioms of M and M', respectively, where $\iota(q_0) = \iota'(q'_0)$ is the start state of D. Then the pair $\langle q_0, q'_0\rangle$ is co-reachable. Let $\langle q, q'\rangle$ be a pair of co-reachable states. Then $\iota(q) = \iota'(q')$ should hold. For $f \in \Sigma$, assume that $\delta_D(\iota(q), f)$ is defined. Let

$$\begin{array}{lcl} q(f(x_1, \ldots, x_m)) & \rightarrow & u_0 \cdot q_1(x_{\sigma(1)}) \cdot u_1 \cdot \ldots \cdot u_{n-1} \cdot q_n(x_{\sigma(n)}) \cdot u_n \\ q'(f(x_1, \ldots, x_m)) & \rightarrow & u'_0 \cdot q'_1(x_{\sigma'(1)}) \cdot u'_1 \cdot \ldots \cdot u'_{n-1} \cdot q'_n(x_{\sigma'(n')}) \cdot u'_{n'} \end{array} \tag{3.4}$$

be the rules of q, q' for f, respectively. Then $\langle q_j, q'_{j'}\rangle$ is co-reachable whenever $\sigma(j) = \sigma'(j')$ holds. In particular, we then have $\iota(q_j) = \iota'(q'_{j'})$.

The pair $\langle q, q'\rangle$ of co-reachable states is called same-ordered, if for each corresponding pair of rules (3.4), $n = n'$ and $\sigma = \sigma'$. Finally, M and M' are *same-ordered* if for every co-reachable pair $\langle q, q'\rangle$ of states of M, M', and every $f \in \Sigma$, each pair of rules (3.4) is same-ordered whenever $\delta_D(\iota(q), f)$ is defined.

Given that the LTs M and M' are same-ordered relative to D, we can represent the set of pairs of runs of M and M' on input trees by means of a single context-free grammar G. The set of nonterminals of G consists of a distinct start nonterminal S together with all co-reachable pairs $\langle q, q'\rangle$ of states q, q' of M, M', respectively. The set of terminal symbols T of G is given by $\{a, a^{-1}, \bar{a}, \bar{a}^{-1} \mid a \in \mathsf{A}\}$ for fresh distinct symbols $\bar{a}, \bar{a}^{-1}, a \in \mathsf{A}$. Let $\langle q, q'\rangle$ be a co-reachable pair of states of M, M', and $f \in \Sigma$ such that $\delta_D(\iota(q), f)$ is defined. For each corresponding pair of rules

(3.4), G receives the rule

$$\langle q, q' \rangle \to u_0 \cdot \bar{u}'_0 \cdot \langle q_1, q'_1 \rangle \cdot u_1 \cdot \bar{u}'_1 \cdot \ldots \cdot u_{n-1} \cdot \bar{u}'_{n-1} \cdot \langle q_n, q'_n \rangle \cdot u_n \cdot \bar{u}'_n$$

where $\bar{u}'_i$ is obtained from u'_i by replacing each output symbol $a \in \mathsf{A}$ with its barred copy $\bar{a}$ as well as each inverse a^{-1} with its barred copy $\bar{a}^{-1}$. For the axioms $u_0 q(x_1) u_1$ and $u'_0 q'(x_1) u'_1$ of M, M', respectively, we introduce the rule $S \to u_0 \cdot \bar{u}'_0 \cdot \langle q, q' \rangle \cdot u_1 \cdot \bar{u}'_1$ where again $\bar{u}'_i$ are the barred copies of u'_i. We define morphisms $f, g : T^* \to \mathcal{F}_\mathsf{A}$ by

$$\begin{array}{lll} f(a) = a & f(a^{-1}) = a^{-1} & f(\bar{a}) = f(\bar{a}^{-1}) = \varepsilon \\ g(\bar{a}) = a & g(\bar{a}^{-1}) = a^{-1} & g(a) = g(a^{-1}) = \varepsilon \end{array}$$

for $a \in \mathsf{A}$. Then M and M' are equivalent relative to D iff $g(w) = f(w)$ for all $w \in \mathcal{L}(G)$. Combining Plandowski's polynomial construction of a test set for a context-free language to check morphism equivalence over finitely generated free groups [Pla94, Theorem 6], with Lohrey's polynomial algorithm for checking equivalence of SLPs over the free group [Loh04], we deduce that the equivalence of the morphisms f and g on all words generated by the context-free grammar G, is decidable in polynomial time. Consequently, we obtain:

Corollary 4. *Equivalence of same-ordered LTs relative to a DTA D is decidable in polynomial time.* □

Next, we observe that for every LT M with compatible map ι and non-trivial states only, a *canonical* ordering can be established.

Definition 5. *We call M* ordered *(relative to D) if for all rules of the form* (3.3), *with $\mathcal{L}(q_i) \cdot u_i \cdot \ldots \cdot u_{j-1} \cdot \mathcal{L}(q_j) \subseteq v \cdot \langle p \rangle$, $p \in \mathcal{F}_\mathsf{A}$ the ordering $\sigma(i) < \ldots < \sigma(j)$ holds.*

We show that two ordered LTs, when they are equivalent, are necessarily *same-ordered*. The proof of this claim is split in two parts. First, we prove that the *set* of indices of subtrees processed by equivalent co-reachable states are identical and second, that the order is the same.

Lemma 23. *Let M, M' be LTs with compatible maps ι and ι', respectively, and non-trivial states only so that M and M' are equivalent relative to the DTA D. Let $\langle q, q' \rangle$ be a pair of co-reachable states of M and M'. Assume that $\delta_D(\iota(q), f)$ is defined for some $f \in \Sigma$ and consider the corresponding pair of rules* (3.4). *Then the following holds:*

1. *$\{\sigma(1), \ldots, \sigma(n)\} = \{\sigma'(1), \ldots, \sigma'(n')\}$;*
2. *$\sigma = \sigma'$.*

Proof. Since $\langle q, q' \rangle$ is a co-reachable pair of states, there are elements $\alpha, \alpha', \beta, \beta' \in \mathcal{F}_{\mathsf{A}}$ such that

$$\alpha \cdot [\![q]\!](t) \cdot \beta = \alpha' \cdot [\![q']\!](t) \cdot \beta'$$

holds for all $t \in \mathsf{dom}(\iota(q))$. Consider the first statement. Assume for a contradiction that $q_k(x_j)$ occurs on the right-hand side of the rule for q but x_j does not occur on the right-hand side of the rule for q'. Then, there are input trees $t = f(t_1, \ldots, t_m)$ and $t' = f(t'_1, \ldots, t'_m)$, both in $\mathsf{dom}(\iota(q))$, such that $[\![q_k]\!](t_j) \neq [\![q_k]\!](t'_j)$ and $t_i = t'_i$ for all $i \neq j$. Moreover, there are $\mu_1, \mu_2 \in \mathcal{F}_{\mathsf{A}}$ s.t.

$$\alpha \cdot [\![q]\!](t) \cdot \beta = \alpha \cdot \mu_1 \cdot [\![q_k]\!](t_j) \cdot \mu_2 \cdot \beta \neq \alpha \cdot \mu_1 \cdot [\![q_k]\!](t'_j) \cdot \mu_2 \cdot \beta = \alpha \cdot [\![q]\!](t') \cdot \beta$$

But then,

$$\alpha \cdot [\![q]\!](t) \cdot \beta = \alpha' \cdot [\![q']\!](t) \cdot \beta' = \alpha' \cdot [\![q']\!](t') \cdot \beta' = \alpha \cdot [\![q]\!](t') \cdot \beta$$

— a contradiction. By an analogous argument for some x_j only occurring in the right-hand side of the rule for q' the first statement follows.

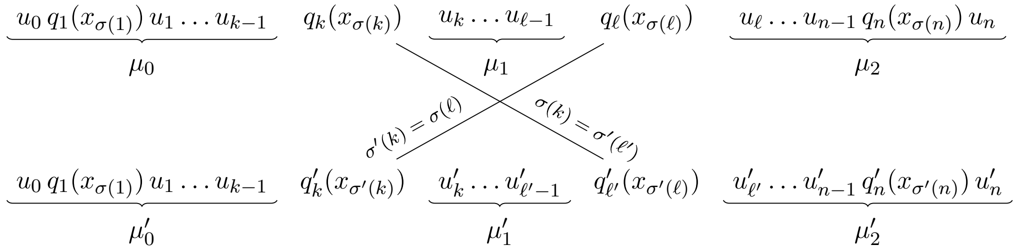
Consider the second statement. Assume for contradiction that the mappings σ and σ' in the corresponding rules (3.4) differ. Let k denote the minimal index so that $\sigma(k) \neq \sigma'(k)$. W.l.o.g., we assume that $\sigma'(k) < \sigma(k)$. By the first statement, $n = n'$ and $\{\sigma(1), \ldots, \sigma(n)\} = \{\sigma'(1), \ldots, \sigma'(n)\}$. Then there are $\ell, \ell' > k$ such that

$$\sigma'(k) = \sigma(\ell) < \sigma(k) = \sigma'(\ell')$$

Let $t = f(t_1, \ldots, t_n) \in \mathsf{dom}(\iota(q))$ be an input tree. For that we obtain

$$\begin{aligned} \mu_0 &:= u_0 \cdot [\![q_1]\!](t_{\sigma(1)}) \cdot \ldots \cdot u_{k-1} & \mu'_0 &:= u'_0 \cdot [\![q'_1]\!](t_{\sigma'(1)}) \cdot \ldots \cdot u'_{k-1} \\ \mu_1 &:= u_k \cdot [\![q_{k+1}]\!](t_{\sigma(k+1)}) \cdot \ldots \cdot u_{\ell-1} & \mu'_1 &:= u'_k \cdot [\![q'_{k+1}]\!](t_{\sigma'(k+1)}) \cdot \ldots \cdot u'_{\ell'-1} \\ \mu_2 &:= u_\ell \cdot [\![q_\ell]\!](t_{\sigma(\ell)}) \cdot \ldots \cdot u_n & \mu'_2 &:= u'_{\ell'} \cdot [\![q'_{\ell'}]\!](t_{\sigma'(\ell')}) \cdot \ldots \cdot u'_n \end{aligned}$$

as illustrated in the following picture.


$\underbrace{u_0\, q_1(x_{\sigma(1)})\, u_1 \ldots u_{k-1}}_{\mu_0}$
$q_k(x_{\sigma(k)})$
$\underbrace{u_k \ldots u_{\ell-1}}_{\mu_1}$
$q_\ell(x_{\sigma(\ell)})$
$\underbrace{u_\ell \ldots u_{n-1}\, q_n(x_{\sigma(n)})\, u_n}_{\mu_2}$
$\sigma'(k) = \sigma(\ell)$
$\sigma(k) = \sigma'(\ell')$
$\underbrace{u_0\, q_1(x_{\sigma(1)})\, u_1 \ldots u_{k-1}}_{\mu'_0}$
$q'_k(x_{\sigma'(k)})$
$\underbrace{u'_k \ldots u'_{\ell'-1}}_{\mu'_1}$
$q'_{\ell'}(x_{\sigma'(\ell)})$
$\underbrace{u'_{\ell'} \ldots u'_{n-1}\, q'_n(x_{\sigma'(n)})\, u'_n}_{\mu'_2}$

Then for all input trees $t' \in \mathsf{dom}(\iota(q_k))$, $t'' \in \mathsf{dom}(\iota(q'_k))$,

$$\alpha \cdot \mu_0 \cdot [\![q_k]\!](t') \cdot \mu_1 \cdot [\![q_\ell]\!](t'') \cdot \mu_2 \cdot \beta = \alpha' \cdot \mu'_0 \cdot [\![q'_k]\!](t'') \cdot \mu'_1 \cdot [\![q'_{\ell'}]\!](t') \cdot \mu'_2 \cdot \beta'$$

Let $\gamma' = \mu_0^{-1} \cdot \alpha^{-1} \cdot \alpha' \cdot \mu_0'$. Then

$$[\![q_k]\!](t') \cdot \mu_1 \cdot [\![q_\ell]\!](t'') \cdot \mu_2 \cdot \beta = \gamma' \cdot [\![q_k']\!](t'') \cdot \mu_1' \cdot [\![q_{\ell'}']\!](t') \cdot \mu_2' \cdot \beta'$$

By Lemma 20, we obtain that for all $w_1, w_2 \in \mathcal{L}(q_k)$ and $v_1, v_2 \in \mathcal{L}(q_\ell)$, $w_2^{-1} \cdot w_1 \in \mu_1 \cdot \langle p \rangle \cdot \mu_1^{-1}$ and $v_1 \cdot v_2^{-1} \in \langle p \rangle$ for some primitive p.

If $\ell = k+1$, i.e., there is no further state between $q_k(x_{\sigma(k)})$ and $q_\ell(x_{\sigma(\ell)})$, then $\mu_1 = u_k$, $\mathcal{L}(q_k) \subseteq w \cdot u_k \cdot \langle p \rangle \cdot u_k^{-1}$ and $\mathcal{L}(q_\ell) \subseteq \langle p \rangle \cdot v$ for some fixed $w \in \mathcal{L}(q_k)$ and $v \in \mathcal{L}(q_\ell)$. As $\sigma(k) > \sigma'(k) = \sigma(\ell)$, this contradicts M being ordered.
For the case that there is at least one occurrence of a state between $q_k(x_{\sigma(k)})$ and $q_\ell(x_{\sigma(\ell)})$, we show that for all $\alpha_1, \alpha_2 \in u_k \cdot \mathcal{L}(q_{k+1}) \cdot \ldots \cdot u_{\ell-1} =: \hat{L}$, $\alpha_1^{-1} \cdot \alpha_2 \in \langle p \rangle$ holds. We fix $w_1, w_2 \in \mathcal{L}(q_k)$ and $v_1, v_2 \in \mathcal{L}(q_\ell)$ with $w_1 \neq w_2$ and $v_1 \neq v_2$. For every $\alpha \in \hat{L}$, we find by Lemma 20, primitive p_α and exponent $r_\alpha \in \mathbb{Z}$ such that $v_1 \cdot v_2^{-1} = p_\alpha^{r_\alpha}$ holds. Since p_α is primitive, this means that $p_\alpha = p$ or $p_\alpha = p^{-1}$. Furthermore, there must be some exponent r'_α such that $w_1^{-1} \cdot w_2 = \alpha \cdot p^{r'_\alpha} \cdot \alpha^{-1}$. For $\alpha_1, \alpha_2 \in \hat{L}$, we therefore have that

$$p^{r'_{\alpha_1}} = (\alpha_1^{-1} \cdot \alpha_2) \cdot p^{r'_{\alpha_2}} \cdot (\alpha_1^{-1} \cdot \alpha_2)^{-1}$$

Therefore by Lemma 19, $\alpha_1^{-1} \cdot \alpha_2 \in \langle p \rangle$. Let us fix some $w_k \in \mathcal{L}(q_k)$, $\alpha \in \hat{L} = u_k \cdot \mathcal{L}(q_{k+1}) \cdot \ldots \cdot u_{\ell-1}$, and $w_l \in \mathcal{L}(q_l)$. Then $\mathcal{L}(q_k) \subseteq w_k \cdot \alpha \cdot \langle p \rangle \cdot \alpha^{-1}$, $\hat{L} \subseteq \alpha \cdot \langle p \rangle$ and $\mathcal{L}(q_l) \subseteq \langle p \rangle \cdot w_l$. Therefore,

$$\mathcal{L}(q_k) \cdot u_k \cdot \ldots \cdot \mathcal{L}(q_\ell) \subseteq w_k \cdot \alpha \cdot \langle p \rangle \cdot \alpha^{-1} \cdot \alpha \cdot \langle p \rangle \cdot \langle p \rangle \cdot w_l = w_k \cdot \alpha \cdot \langle p \rangle \cdot w_l$$

As $\sigma(k) > \sigma'(k) = \sigma(\ell)$, this again contradicts M being ordered. □

As equivalence of same-ordered and therefore ordered LTs is decidable in polynomial time it remains to show that every LT can be ordered in polynomial time. For that, we rely on the following characterization.

Lemma 24. *Assume that $L_1, \ldots, L_n$ are neither empty nor singleton subsets of $\mathcal{F}_A$ and $u_1, \ldots, u_{n-1} \in \mathcal{F}_A$. Then there are $v_1, \ldots, v_n \in \mathcal{F}_A$ such that*

$$L_1 \cdot u_1 \cdot \ldots \cdot L_{n-1} \cdot u_{n-1} \cdot L_n \subseteq v \cdot \langle p \rangle \tag{3.5}$$

holds if and only if for $i = 1, \ldots, n$, $L_i \subseteq v_i \cdot \langle p_i \rangle$ with

$$\begin{array}{rcll} p_n & = & p & \\ p_i & = & (u_i \cdot v_{i+1}) \cdot p_{i+1} \cdot (u_i \cdot v_{i+1})^{-1} & \text{for } i < n \end{array}$$

and

$$v^{-1} \cdot v_1 \cdot u_1 \cdot \ldots \cdot v_{n-1} \cdot u_{n-1} \cdot v_n \in \langle p \rangle \tag{3.6}$$

Proof. Let $s_1 = \varepsilon$. For $i = 2, \dots, n$ we fix some word $s_i \in L_1 \cdot u_1 \cdot L_2 \cdot \dots \cdot L_{i-1} \cdot u_{i-1}$. Likewise, let $t_n = \varepsilon$ and for $i = 1, \dots, n-1$ fix some word $t_i \in u_i \cdot L_{i+1} \cdot \dots \cdot L_n$, and define $v_i = s_i^{-1} \cdot v \cdot t_i^{-1}$.

First assume that the inclusion (3.5) holds. Let $p'_i = t_i \cdot p \cdot t_i^{-1}$. Then for all i, $s_i \cdot L_i \cdot t_i \subseteq v \cdot \langle p \rangle$, and therefore

$$L_i \subseteq s_i^{-1} \cdot v \cdot \langle p \rangle \cdot t_i^{-1} = s_i^{-1} \cdot v \cdot t_i^{-1} \cdot t_i \cdot \langle p \rangle \cdot t_i^{-1} = v_i \langle p'_i \rangle$$

We claim that $p'_i = p_i$ for all $i = 1, \dots, n$. We proceed by induction on $n - i$. As $t_n = \varepsilon$, we have that $p'_n = p = p_n$. For $i < n$, we can rewrite $t_i = u_i \cdot w_{i+1} \cdot t_{i+1}$ where $w_{i+1} \in L_{i+1}$ and thus is of the form $v_{i+1} \cdot p_{i+1}^{k_{i+1}}$ for some exponent k_{i+1}.

$$\begin{array}{rcll} p'_i & = & t_i \cdot p \cdot t_i^{-1} & \\ & = & u_i \cdot w_{i+1} \cdot t_{i+1} \cdot p \cdot t_{i+1}^{-1} \cdot w_{i+1}^{-1} \cdot u_i^{-1} & \\ & = & u_i \cdot w_{i+1} \cdot p_{i+1} \cdot w_{i+1}^{-1} \cdot u_i^{-1} & \text{by I.H.} \\ & = & u_i \cdot v_{i+1} \cdot p_{i+1} \cdot v_{i+1}^{-1} \cdot u_i^{-1} & \\ & = & p_i & \end{array}$$

It remains to prove the inclusion (3.6). Since $w_i \in L_i$, we have by (3.5) that $v^{-1} \cdot w_1 \cdot u_1 \cdot \dots w_n \cdot u_n \in \langle p \rangle$ holds. Now we calculate:

$$\begin{array}{rcl} v^{-1} \cdot w_1 \cdot u_1 \cdot \dots u_{n-1} \cdot w_n & = & v^{-1} \cdot v_1 \cdot p_1^{k_1} \cdot u_1 \cdot \dots \cdot u_{n-1} \cdot v_n \cdot p_n^{k_n} \\ & = & v^{-1} \cdot v_1 \cdot u_1 \cdot v_2 \cdot p_2^{k_1+k_2} \cdot u_2 \cdot \dots \cdot u_{n-1} \cdot v_n \cdot p_n^{k_n} \\ & \dots & \\ & = & v^{-1} \cdot v_1 \cdot u_1 \cdot \dots v_{n-1} \cdot u_{n-1} \cdot v_n \cdot p_n^{k} \end{array}$$

where $k = k_1 + \dots + k_n$. Since $p_n = p$, the claim follows.

The other direction of the claim of the lemma follows directly:

$$\begin{array}{rl} & L_1 u_1 \dots L_{n-1} u_{n-1} L_n \\ \subseteq & v_1 \cdot \langle p_1 \rangle \cdot u_1 \cdot \dots \cdot v_{n-1} \cdot \langle p_{n-1} \rangle \cdot u_{n-1} \cdot v_n \cdot \langle p_n \rangle \\ = & v_1 \cdot u_1 \cdot v_2 \cdot \langle p_2 \rangle \cdot \langle p_2 \rangle \cdot u_2 \cdot \dots \cdot v_{n-1} \cdot \langle p_{n-1} \rangle \cdot u_{n-1} \cdot v_n \cdot \langle p_n \rangle \\ = & v_1 \cdot u_1 \cdot v_2 \cdot \langle p_2 \rangle \cdot u_2 \cdot \dots \cdot v_{n-1} \cdot \langle p_{n-1} \rangle \cdot u_{n-1} \cdot v_n \cdot \langle p_n \rangle \\ \dots & \\ = & v_1 \cdot u_1 \cdot v_2 \cdot \dots \cdot u_{n-1} \cdot v_n \cdot \langle p_n \rangle \\ = & v_1 \cdot u_1 \cdot v_2 \cdot \dots \cdot u_{n-1} \cdot v_n \cdot \langle p \rangle \\ \subseteq & v \cdot \langle p \rangle \end{array}$$

where the last inclusion follows from (3.6). □

Let us call a non-empty, non-singleton language $L \subseteq \mathcal{F}_{\mathsf{A}}$ *periodic*, if $L \subseteq v \cdot \langle p \rangle$ for some $v, p \in \mathcal{F}_{\mathsf{A}}$. Lemma 24 then implies that if a concatenation of languages and elements from $\mathcal{F}_{\mathsf{A}}$ is periodic, then so must be all non-singleton component languages. In fact, the languages in the composition can then be arbitrarily permuted.

Corollary 5. *Assume for non-empty, nonsingleton languages $L_1, \dots, L_n \subseteq \mathcal{F}_{\mathsf{A}}$ and $u_1, \dots, u_{n-1} \in \mathcal{F}_{\mathsf{A}}$ that property* (3.5) *holds. Then for every permutation π, there are elements $u_{\pi,0}, \dots, u_{\pi,n} \in \mathcal{F}_{\mathsf{A}}$ such that*

$$L_1 \cdot u_1 \cdot \ldots \cdot L_{n-1} \cdot u_{n-1} \cdot L_n = u_{\pi,0} \cdot L_{\pi(1)} \cdot u_{\pi,1} \cdot \ldots \cdot u_{\pi_n-1} \cdot L_{\pi(n)} \cdot u_{\pi,n}$$

Proof. For $i = 1, \dots, n$, let v_i and p_i be defined as in Lemma 24. Then for all i, $L_i \subseteq v_i \cdot \langle p_i \rangle$. Moreover, the languages L'_i defined by $L'_n = v_n^{-1} \cdot L_n$ and for $i < n$,

$$L'_i = (u_i \cdot v_{i+1} \cdot \ldots \cdot u_{n-1} \cdot v_n)^{-1} \cdot (v_i^{-1} \cdot L_i) \cdot (u_i \cdot v_{i+1} \cdot \ldots \cdot u_{n-1} \cdot v_n)$$

all are subsets of $\langle p \rangle$. Therefore their compositions can arbitrarily be permuted. At the same time,

$$L_1 \cdot u_1 \cdot \ldots \cdot L_{n-1} \cdot u_{n-1} \cdot L_n = v_1 \cdot u_1 \cdot \ldots \cdot v_{n-1} \cdot u_{n-1} \cdot v_n \cdot L'_1 \cdot \ldots L'_n$$

From that, the corollary follows. □

Example 18. *We reconsider* LT *M' and* DTA *B from Example 17. We observe that $\mathcal{L}(q_1) \subseteq a\langle ba \rangle$, $\mathcal{L}(q_2) \subseteq \langle ab \rangle$, and thus $\mathcal{L}(q_0) = \mathcal{L}(q_1) \cdot b \cdot \mathcal{L}(q_2) \subseteq \langle ab \rangle$. Accordingly, the rule for state q_0 and input symbol f is not ordered. Following the notation of Corollary 5, we find $v_1 = a$, $u_1 = b$ and $v_2 = \varepsilon$, and the rule for q_0 and f can be reordered to*

$$q_0(f(x_1, x_2)) \to ab \cdot q_2(x_1) \cdot b^{-1}a^{-1} \cdot q_1(x_2) \cdot b$$

This example shows major improvements compared to the construction in [BP16] which we recap in Section 3.3, see Example 19. Since we have inverses at hand, only local *changes must be applied to the sub-sequence $q_1(x_2) \cdot b \cdot q_2(x_1)$. In contrast to the construction in [BP16], neither auxiliary states nor further changes to the rules of q_1 and q_2 are required.*

By Corollary 5, the order of occurrences of terms $q_k(x_{\sigma(k)})$ can be permuted in every sub-sequence $q_i(x_{\sigma(i)}) \cdot u_i \cdot \ldots \cdot u_{j-1} \cdot q_j(x_{\sigma(j)})$ where $\mathcal{L}(q_i) \cdot u_i \cdot \ldots \cdot u_{j-1} \cdot \mathcal{L}(q_j) \subseteq u \cdot \langle p \rangle$ is periodic, to satisfy the requirements of an ordered LT. A sufficient condition for that is, according to Lemma 24, that $\mathcal{L}(q_k)$ is periodic for each q_k occurring in that sub-sequence. Therefore we will determine the subset of *all* states q where $\mathcal{L}(q)$ is periodic, and if so elements v_q, p_q such that $\mathcal{L}(q) \subseteq v_q \cdot \langle p_q \rangle$. In order to do so we compute an *abstraction* of the sets $\mathcal{L}(q)$ by means of a complete lattice which both reports constant values and also captures periodicity.

Let $\mathcal{D} = 2^{\mathcal{F}_{\mathsf{A}}}$ denote the complete lattice of subsets of the free group $\mathcal{F}_{\mathsf{A}}$. We define a *projection* $\alpha : \mathcal{D} \to \mathcal{D}$ by $\alpha(\emptyset) = \emptyset$, $\alpha(\{g\}) = \{g\}$, and for languages L with at least two elements,

$$\alpha(L) = \begin{cases} g \cdot \langle p \rangle & \text{if } L \subseteq g \cdot \langle p \rangle \text{ and } p \text{ is primitive} \\ \mathcal{F}_{\mathsf{A}} & \text{otherwise} \end{cases}$$

The projection α is a *closure* operator, i.e., is a monotonic function with $L \subseteq \alpha(L)$, and $\alpha(\alpha(L)) = \alpha(L)$. The image of α can be considered as an *abstract* complete lattice $\mathcal{D}^\sharp$, partially ordered by subset inclusion. Thereby, the abstraction α commutes with least upper bounds as well as with the group operation. For that, we define *abstract* versions $\sqcup, \star : (\mathcal{D}^\sharp)^2 \to \mathcal{D}^\sharp$ of set union and the group operation by

$$A_1 \sqcup A_2 = \alpha(A_1 \cup A_2) \qquad A_1 \star A_2 = \alpha(A_1 \cdot A_2)$$

In fact, "$\sqcup$" is the least upper bound operation for $\mathcal{D}^\sharp$. The two abstract operators can also be more explicitly defined by:

$$\begin{array}{lcl}
\emptyset \sqcup L & = & L \sqcup \emptyset \quad = \quad L \\
\mathcal{F}_\mathsf{A} \sqcup L & = & L \sqcup \mathcal{F}_\mathsf{A} \quad = \quad \mathcal{F}_\mathsf{A} \\
\{g_1\} \sqcup \{g_2\} & = & \begin{cases} \{g_1\} & \text{if } g_1 = g_2 \\ g_1 \cdot \langle p \rangle & \text{if } g_1 \neq g_2, p \text{ primitive root of } g_1^{-1} \cdot g_2 \end{cases} \\
\{g_1\} \sqcup g_2 \cdot \langle p \rangle & = & g_2 \cdot \langle p \rangle \sqcup \{g_1\} \quad = \quad \begin{cases} g_2 \cdot \langle p \rangle & \text{if } g_1 \in g_2 \cdot \langle p \rangle \\ \mathcal{F}_\mathsf{A} & \text{otherwise} \end{cases} \\
g_1 \cdot \langle p_1 \rangle \sqcup g_2 \cdot \langle p_2 \rangle & = & \begin{cases} g_1 \cdot \langle p_1 \rangle & \text{if } p_2 \in \langle p_1 \rangle \text{ and } g_2^{-1} \cdot g_1 \in \langle p_1 \rangle \\ \mathcal{F}_\mathsf{A} & \text{otherwise} \end{cases}
\end{array}$$

Likewise, the *abstract product* operator "$\star$" can explicitly be defined by:

$$\begin{array}{lcl}
\emptyset \star L & = & L \star \emptyset \quad = \quad \emptyset \\
\mathcal{F}_\mathsf{A} \star L & = & L \star \mathcal{F}_\mathsf{A} \quad = \quad F_\mathsf{A} \qquad \text{for } L \neq \emptyset \\
\{g_1\} \star \{g_2\} & = & \{g_1 \cdot g_2\} \\
\{g_1\} \star g_2 \cdot \langle p \rangle & = & (g_1 \cdot g_2) \cdot \langle p \rangle \\
g_1 \cdot \langle p \rangle \star \{g_2\} & = & (g_1 \cdot g_2) \cdot \langle g_2^{-1} \cdot p \cdot g_2 \rangle \\
g_1 \cdot \langle p_1 \rangle \star g_2 \cdot \langle p_2 \rangle & = & \begin{cases} (g_1 \cdot g_2) \cdot \langle p_2 \rangle & \text{if } g_2^{-1} \cdot p_1 \cdot g_2 \in \langle p_2 \rangle \\ \mathcal{F}_\mathsf{A} & \text{otherwise} \end{cases}
\end{array}$$

Lemma 25. *For all subsets $L_1, L_2 \subseteq \mathcal{F}_A$,*

$$\begin{array}{lcl}
\alpha(L_1 \cup L_2) & = & \alpha(L_1) \sqcup \alpha(L_2) \\
\alpha(L_1 \cdot L_2) & = & \alpha(L_1) \star \alpha(L_2)
\end{array}$$

Proof. As $\emptyset \cup L = L \cup \emptyset = \emptyset$, it follows that $\alpha(\emptyset \cup L) = \alpha(L \cup \emptyset) = \alpha(\emptyset) = \emptyset = \emptyset \sqcup L' = L' \sqcup \emptyset = \alpha(\emptyset) \sqcup \alpha(L) = \alpha(L) \sqcup \alpha(\emptyset)$.

Assume that $\alpha(L_1) = \mathcal{F}_\mathsf{A}$. Let L_2 be some language, then $\alpha(L_1 \cup L_2) = \alpha(L_2 \cup L_1) = \mathcal{F}_\mathsf{A}$ and $\alpha(L_1) \sqcup \alpha(L_2) = \mathcal{F}_\mathsf{A} \sqcup \alpha(L_2) = \mathcal{F}_\mathsf{A} = \alpha(L_2) \sqcup \mathcal{F}_\mathsf{A} = \alpha(L_2) \sqcup \alpha(L_1)$. The case where $\alpha(L_2) = \mathcal{F}_\mathsf{A}$ is analogous.

For $\alpha(L_1) = \{g_1\}$, $\alpha(L_2) = \{g_2\}$, both languages are singleton, and we obtain that $\{g_1\} \cup \{g_2\} = \{g_1\}$ if and only if $g_1 = g_2$. Accordingly, $\alpha(L_1 \cup L_2) = \alpha(\{g_1\}) =$

$\{g_1\} = \alpha(\{g_1\}) \sqcup \alpha(\{g_2\})$. If $g_1 \neq g_2$ then $\{g_1\} \cup \{g_2\} \subseteq g_1 \cdot \langle g_1^{-1} \cdot g_2\rangle$ and $\alpha(L_1 \cup L_2) = g_1 \cdot \langle p\rangle$ with p the primitive root of $g_1^{-1} g_2$. Therefore, $\alpha(L_1 \cup L_2) = g_1 \cdot \langle p\rangle = \alpha(\{g_1\}) \sqcup \alpha(\{g_2\})$.

Assume that $\alpha(L_1) = \{g_1\}$ and $\alpha(L_2) = g_2 \cdot \langle p_2\rangle$ for some primitive p_2. If $g_1 \in g_2 \cdot \langle p_2\rangle$, then $\alpha(L_1 \cup L_2) = g_2 \cdot \langle p_2\rangle = \alpha(L_1) \sqcup \alpha(L_2)$. Otherwise, i.e., if $g_1 \notin g_2 \cdot \langle p_2\rangle$, then $L_1 \cup L_2$ is not contained in $g \cdot \langle p\rangle$ for any p (since p_2 was chosen primitive), and therefore, $\alpha(L_1 \cup L_2) = \mathcal{F}_{\mathsf{A}} = \alpha(L_1) \sqcup \alpha(L_2)$. A similar argument applies if $\alpha(L_2) = \{g_1\}$, and $\alpha(L_1) = g_2 \cdot \langle p_2\rangle$.

Assume that $\alpha(L_1) = g_1 \cdot \langle p_1\rangle$ and $\alpha(L_2) = g_2 \cdot \langle p_2\rangle$ for some primitive p_1, p_2. If $p_2 \in \langle p_1\rangle$ as well as $g_2^{-1} \cdot g_1 \in \langle p_1\rangle$, then $g_1 \cdot \langle p_1\rangle = g_2 \cdot \langle p_2\rangle$ (due to primitivity of p_1, p_2). Moreover, $\alpha(L_1 \cup L_2) = g_1 \cdot \langle p_1\rangle = \alpha(L_1) \sqcup \alpha(L_2)$. Otherwise, i.e., if $p_2 \notin \langle p_1\rangle$ or $g_2^{-1} \cdot g_1 \notin \langle p_1\rangle$, then $L_1 \cup L_2$ cannot be subset of $g \cdot \langle p\rangle$ for any $g, p \in \mathcal{F}_{\mathsf{A}}$. Therefore, $\alpha(L_1 \cup L_2) = \mathcal{F}_{\mathsf{A}} = \alpha(L_1) \sqcup \alpha(L_2)$.

For the concatenation with the empty set and the product operator we find $\alpha(\emptyset \cdot L) = \alpha(L \cdot \emptyset) = \alpha(\emptyset) = \emptyset = \alpha(\emptyset) \star \alpha(L) = \alpha(L) \star \alpha(\emptyset)$.

Assume that $\alpha(L_1) = \mathcal{F}_{\mathsf{A}}$. Then $L_1 \not\subseteq g \cdot \langle p\rangle$ for any $g, p \in \mathcal{F}_{\mathsf{A}}$. Assume that $L_2 \subseteq \mathcal{F}_{\mathsf{A}}$ is nonempty. Then by Lemma 24, $L_1 \cdot L_2$ and $L_2 \cdot L_1$ cannot be contained in $g' \cdot \langle p'\rangle$ for any g', p'. Therefore, $\alpha(L_1 \cdot L_2) = \alpha(L_2 \cdot L_1) = \mathcal{F}_{\mathsf{A}} = \alpha(L_1) \star \alpha(L_2) = \alpha(L_2) \star \alpha(L_1)$.

For $\alpha(L_1) = \{g_1\}, \alpha(L_2) = \{g_2\}$, both languages are singletons, and we obtain $\alpha(L_1 \cdot L_2) = \{g_1 \cdot g_2\} = \{g_1\} \star \{g_2\} = \alpha(L_1) \star \alpha(L_2)$.

Now assume that $\alpha(L_1) = \{g_1\}$ and $\alpha(L_2) = g_2\langle p_2\rangle$. Then $L_1 = \{g_1\}$, while $L_1 \cdot L_2$ is not a singleton language, but contained in $g_1 \cdot g_2 \cdot \langle p_2\rangle$. Therefore, $\alpha(L_1 \cdot L_2) = g_1 \cdot g_2 \cdot \langle p_2\rangle = \alpha(L_1) \star \alpha(L_2)$. Likewise, if $\alpha(L_1) = g_1 \cdot \langle p_1\rangle$ and $\alpha(L_2) = \{g_2\}$, then $L_2 = \{g_2\}$, and $L_1 \cdot L_2$ is a non-singleton language contained in $g_1 \cdot g_2 \cdot \langle g_2^{-1} p_1 g_2\rangle$. Therefore, $\alpha(L_1 \cdot L_2) = g_1 \cdot g_2 \cdot \langle g_2^{-1} p_1 g_2\rangle = \alpha(L_1) \star \alpha(L_2)$.

Finally, let $\alpha(L_1) = g_1 \cdot \langle p_1\rangle$ and $\alpha(L_2) = g_2 \cdot \langle p_2\rangle$ be both ultimately periodic languages. By Lemma 24, $L_1 \cdot L_2$ is ultimately periodic if and only if $g_2^{-1} \cdot p_1 \cdot g_2 \in \langle p_2\rangle$. Thus if $L_1 \cdot L_2$ is ultimately periodic, then $\alpha(L_1 \cdot L_2) = g_1 \cdot g_2 \cdot \langle p_2\rangle = \alpha(L_1) \star \alpha(L_2)$. Otherwise, $L_1 \cdot L_2 \not\subseteq g \cdot \langle p\rangle$ for any $g, p \in \mathcal{F}_{\mathsf{A}}$, and therefore $\alpha(L_1 \cdot L_2) = \mathcal{F}_{\mathsf{A}} = \alpha(L_1) \star \alpha(L_2)$. □

We conclude that α in fact represents a *precise* abstract interpretation in the sense of [Mül06]. Accordingly, we obtain:

Lemma 26. *For every LT M and DTA D with compatible map ι, the sets $\alpha(\mathcal{L}(q))$ with q a state of M can be computed in polynomial time.*

Proof. We introduce one unknown X_q for every state q of M, and one constraint for each rule of M of the form (3.3) where $\delta(\iota(q), f)$ is defined in D. This constraint is given by:

$$X_q \sqsupseteq u_0 \star X_{q_1} \star \ldots \star u_{n-1} \star X_{q_n} \star u_n \tag{3.7}$$

As the right-hand sides of the constraints (3.7) all represent monotonic functions, the given system of constraints has a *least* solution. In order to obtain this solution, we consider for each state q of M, the sequence $X_q^{(i)}, i \geq 0$ of values in $\mathcal{D}^\sharp$ where $X_q^{(0)} = \emptyset$, and for $i > 0$, we set $X_q^{(i)}$ as the least upper bound of the values obtained from the constraints with left-hand side X_q of the form (3.7) by replacing the unknowns X_{q_j} on the right-hand side with the values $X_{q_j}^{(i-1)}$. By induction on $i \geq 0$, we verify that for all states q of M,

$$X_q^{(i)} = \alpha(\mathcal{L}^{(i)}(q))$$

holds. Note that the induction step thereby, relies on Lemma 25.
As each strictly increasing chain of elements in $\mathcal{D}^\sharp$ consists of at most four elements, we have that the least solution of the constraint system is attained after at most $3 \cdot N$ iterations, if N is the number of states of M, i.e., for each state q of M, $X_q^{(3N)} = X_q^{(i)}$ for all $i \geq 3N$. The elements of $\mathcal{D}^\sharp$ can be represented by SLPs where the operations $\star$ and $\sqcup$ run in polynomial time, cf. Lemma 21. Since each iteration requires only a polynomial number of operations $\star$ and $\sqcup$, the statement of the lemma follows. □

We now exploit the information provided by the $\alpha(\mathcal{L}(q))$ to remove trivial states as well as order sub-sequences of right-hand sides which are periodic.

Theorem 2. *Let D be a DTA such that $\mathcal{L}(D) \neq \emptyset$. For every LT M with compatible map ι, an LT M' with compatible map ι' can be constructed in polynomial time such that*

1. *M and M' are equivalent relative to D;*
2. *M' has no trivial states;*
3. *M' is ordered.*

Proof. By Lemma 26, we can, in polynomial time, determine for every state q of M, the value $\alpha(\mathcal{L}(q))$. We use this information to remove from M all trivial states. W.l.o.g., assume that the axiom of M is given by $u_0 \cdot q_0(x_0) \cdot u_1$. If the state q_0 occurring in the axiom of M is trivial with $\mathcal{L}(q_0) = \{v\}$, then M' has no states or rules, but the axiom $u_0 \cdot v \cdot u_1$.

Therefore now assume that q_0 is non-trivial. We then construct an LT M' whose set of states Q' consists of all *non-trivial* states q of M where the compatible map ι' of M' is obtained from ι by restriction to Q'. Since $\mathcal{L}(M) \neq \emptyset$, the state of M occurring in the axiom is non-trivial. Accordingly, the axiom of M is also used as axiom for M'. Consider a non-trivial state q of M and $f \in \Sigma$. If $\delta(\iota(q), f)$ is not defined then M' has the rule $q(f(x_1, \ldots, x_m) \to \bot$. Assume that $\delta(\iota(q), f)$ is defined and M has a rule of the form (3.3). Then M' has the rule

$$q(f(x_1, \ldots, x_m)) \to u_0 \cdot g_1 \cdot \ldots \cdot u_{n-1} \cdot g_n \cdot u_n$$

where for $i = 1, \ldots, n$, g_i equals $q_i(x_{\sigma(i)})$ if q_i is non-trivial, and equals the single word in $\mathcal{L}(q_i)$ otherwise. Obviously, M and M' are equivalent relative to D where M' now has no trivial states, while for every non-trivial state q, the semantics of q in M and M' are the same relative to D. Our goal now is to equivalently rewrite the right-hand side of each rule of M' so that the result is ordered. For each state q of the LT we determine whether there are $v, p \in \mathcal{F}_\mathsf{A}$ such that $\mathcal{L}(q) \subseteq v \cdot \langle p \rangle$, cf. Lemma 26. So consider a rule of M' of the form (3.3). By means of the values $\alpha(\mathcal{L}(q_i))$, $i = 1, \ldots, n$, together with the abstract operation "$\star$", we can determine maximal intervals $[i, j]$ such that $\mathcal{L}(q_i) \cdot u_i \cdot \ldots \cdot u_{j-1} \cdot \mathcal{L}(q_j)$ is periodic, i.e., $\alpha(\mathcal{L}(q_i)) \star u_i \cdot \ldots \star u_{j-1} \star \alpha(\mathcal{L}(q_j)) \subseteq v \cdot \langle p \rangle$ for some $v, p \in \mathcal{F}_\mathsf{A}$. We remark that these maximal intervals are necessarily disjoint. By Corollary 5, for every permutation $\pi : [i, j] \to [i, j]$, elements $u', u'_i, \ldots, u'_j, u'' \in \mathcal{F}_\mathsf{A}$ can be found so that $q_i(x_{\sigma(i)}) \cdot u_i \cdot \ldots \cdot u_{j-1} \cdot q_j(x_{\sigma(j)})$ is equivalent to

$$u' \cdot q_{\pi(i)}(x_{\sigma(\pi(i))}) \cdot u'_i \cdot \ldots \cdot u'_{j-1} \cdot q_{\pi(j)}(x_{\sigma(\pi(j))}) \cdot u''$$

In particular, this is true for the permutation π with $\sigma(\pi(i)) < \ldots < \sigma(\pi(j))$. Accordingly, we rewrite the unordered interval such that it is ordered.

Assuming that all group elements are represented as SLPs, the overall construction runs in polynomial time. □

In summary, we arrive at the main theorem of this chapter.

Theorem 3. *The equivalence of LTs relative to some DTA D can be decided in polynomial time.*

Proof. Assume we are given LTs M, M' with compatible maps (relative to D). By Theorem 2, we may w.l.o.g. assume that M and M' both have no trivial states and are ordered. It can be checked in polynomial time whether or not M and M' are same-ordered. If they are not, then by Lemma 23, they cannot be equivalent relative to D. Therefore now assume that M and M' are same-ordered. Then their equivalence relative to D is decidable in polynomial time by Corollary 4. Altogether we thus obtain a polynomial decision procedure for equivalence of LTs relative to some DTA D. □

3.3 Ordered Form for Linear Tree-to-Word Transducers

We consider linear tree-to-*word* transducers (LTW for short) over the free monoid $\langle \mathsf{A}^*, \cdot, \varepsilon \rangle$ in this section. The rules and therefore the outputs of a tree-to-word transducer contain only words $w \in \mathsf{A}^*$, no inverses $a^{-1} \in \mathsf{A}^{-1}$ occur. Thus, this is a special case of the before considered linear tree transducers LTs. We remind, that we use $\cdot$ to denote concatenation followed by reduction. Even if we consider only output over the free monoid this will be used to build conjugates of words. For example, $w_1^{-1} \cdot w \cdot w_1$ yields the conjugate $w_2 w_1$ of $w = w_1 w_2$. We showed in Theorem 2 that for every LT M an equivalent ordered LT M' can be constructed in polynomial time. This construction can introduce inverses in the rules of the ordered LT M' even if the initial LT M is an LTW. In fact, the LT M from the running example in Section 3.2 is an LTW, cf. Example 17. However, the equivalent ordered LT M' constructed in Example 18 is *not* an LTW since the reordering introduces inverses. We recap the underlying idea of the ordering in the following example.

Example 19. *Recall the LTW M from Section 3.2 with axiom $q_0(x_0)$ and the following rules*

$$\begin{array}{ll} q_0(f(x_1,x_2)) \to q_1(x_2)\, b\, q_2(x_1) & \\ q_1(g(x_1)) \to ab\, q_1(x_1) & q_1(k) \to a \\ q_2(g(x_1)) \to ab\, q_2(x_1) & q_2(k) \to ab \end{array}$$

The top rule with left-hand side $q_0(f(x_1,x_2))$ is not ordered as $\mathcal{L}(q) = \mathcal{L}(q_1)b\mathcal{L}(q_2) \subseteq (ab)^*$ with $\mathcal{L}(q_1) \subseteq a(ba)^*$, $\mathcal{L}(q_2) \subseteq (ab)^*$ but the recursive call on subtree x_2 is left of the recursive call on the subtree x_1. The underlying idea of the ordering is that if the states q_1 and q_2 produce languages over the same period then the recursive calls can be permuted. With inverses at hand this we do not have to change the rules of q_1 and q_2; we replace the rule with left-hand side $q_0(f(x_1,x_2))$ by

$$q_0(f(x_1,x_2)) \to ab\, q_2(x_1) \cdot b^{-1}a^{-1} \cdot q_1(x_2)\, b$$

Then q, q_2 and $b^{-1}a^{-1}q_1 b$ produce all languages over the same period ab. We observe two concepts for that inverses are introduced:

- *The inverse a^{-1} is used to remove the constant prefix a of the language $\mathcal{L}(q_1) \subseteq a(ab)^*$.*
- *The inverse b^{-1} together with the additional constant b after the recursive call of q_1 shifts the period of the language $a^{-1} \cdot \mathcal{L}(q_1) \subseteq \langle ba \rangle$, i.e., $b^{-1}a^{-1} \cdot \mathcal{L}(q_1)\, b \subseteq \langle ab \rangle$.*

Without inverses a new state q_1^{ab}, $\mathcal{L}(q_1^{ab}) \subseteq (ab)^*$ with corresponding rules has to be introduced such that $[\![q_1^{ab}]\!](t) = b^{-1}a^{-1} \cdot q_1(t)\, b$ for all $t \in \mathsf{dom}(\iota(q_1))$. Then, the LTW M can be ordered as follows.

$$\begin{array}{ll} q_0(f(x_1,x_2)) \to ab\, q_2(x_1)\, q_1^{ab}(x_2) & \\ q_1^{ab}(g(x_1)) \to ab\, q_1(x_1) & q_1^{ab}(k) \to \varepsilon \\ q_2(g(x_1)) \to ab\, q_2(x_1) & q_2(k) \to ab \end{array}$$

In Lemma 24 the languages of the states occurring in an interval of the form $\mathcal{L}(q_i)u_i \ldots u_{j-1}\mathcal{L}(q_j) \subseteq v\langle p \rangle$ were characterized. Now, we have to distinguish for the periodic languages of the form $v\langle p\rangle$ two cases, i.e., languages of the form $v \cdot \langle p\rangle$ and $\langle p\rangle \cdot v$. In the free group we could cover both cases with the form $v \cdot \langle p\rangle$ as $v \cdot \langle v^{-1} \cdot p \cdot v\rangle = \langle p\rangle \cdot v$. We call a language $L \subseteq \mathsf{A}^*$ *ultimately periodic* if $L \subseteq vp^*$ or $L \subseteq p^*\, v$ with $v, p \in \mathsf{A}^*$. Accordingly, we call a state q of an LTW producing an ultimately periodic language, $\mathcal{L}(q) \subseteq vp^*$ or $\mathcal{L}(q) \subseteq p^*v$, ultimately periodic. Let q be an ultimately periodic state with $\mathcal{L}(q) \subseteq vp^*$. Then q and q' are *conjugates w.r.t. w*, $w \overset{p}{\sqsubseteq} p^\omega$, if $[\![q]\!](t) = vw[\![q']\!](t) \cdot w^{-1}$, for all $t \in \mathsf{dom}(\iota(q))$.[1] We show that for each LTW an equivalent LTW that is ordered can be constructed in polynomial time. The crucial operation thereby is to compute, given an ultimately periodic state q, a state q^{p_w} such that q and q^{p_w} are conjugates w.r.t. some $w \overset{p}{\sqsubseteq} p^\omega$. For an ultimately periodic state q, $\mathcal{L}(q) \subseteq vp^*$ and $w \overset{p}{\sqsubseteq} p^\omega$ we give a procedure to determine a state q' that is a conjugate to q w.r.t. w in Algorithm 1. Note, that the constructions consider the case that a state is ultimately periodic of the form vp^*. The case that a state is ultimately periodic of the form p^*v can be handled by taking the reversed state q^r. For a word $w = w_0 \dots w_n \in \mathsf{A}^*$, $w_i \in \mathsf{A}$, we denote by w^r the reversed word $w_n \dots w_0$. The reversed state q^r of state q is obtained as follows. For each rule $q(f(x_1, \dots, x_m)) \to u_0 q_1(x_{\sigma(1)}) \dots u_{n-1} q_n(x_{\sigma(n)}) u_n$ we get the rule $q^r(f(x_1, \dots, x_m)) \to u_n^r q_n^r(x_{\sigma(n)}) u_{n-1}^r \dots u_1^r q_1^r(x_{\sigma(1)}) u_0^r$ with q_i^r the reversed state of q_i. By induction we have $[\![q^r]\!](t) = ([\![q]\!](t))^r$ for all $t \in \mathsf{dom}(\iota(q))$. Thus, if q is quasi-periodic with $\mathcal{L}(q) \subseteq p^*v$ ($v \overset{s}{\not\sqsubseteq} p$) then we can construct q^r in polynomial time with $\mathcal{L}(q^r) \subseteq v^r(p^r)^*$. We remind that if no explicit DTA D and corresponding map ι for a LTW is given then we implicitly assume these. This shortens notation

[1] We use $\overset{p}{\sqsubseteq}$ to denote the prefix relation, i.e., $v \overset{p}{\sqsubseteq} w$ if $w = vv'$ for some $v' \in \mathsf{A}^*$ and $\overset{s}{\sqsubseteq}$ to denote the suffix relation, i.e, $v \overset{s}{\sqsubseteq} w$ if $w = v'v$ for some $v' \in \mathsf{A}^*$. Additionally, $w^\omega = www\dots$ denotes an infinite word. For a detailed definition see Section 2.1.

and is frequently used in this section.

```
Input  : DTA $B$, LTW $M$ with state $q$ such that $\mathcal{L}(q) \subseteq vp^*$, and $w \overset{p}{\sqsubset} p^\omega$
Output: LTW $M'$ such that $M'$ and $M$ are equivalent relative to $D$. $M'$
         contains state $q^{p_w}$ with $p_w$ the primitive root of $w^{-1}pw$ and
         $[\![q^{p_w}]\!](t) = w^{-1}v^{-1} \cdot [\![q]\!](t)w$ for all $t \in \mathsf{dom}(\iota(q))$
// For all states we determined whether they produce languages of the
 form $vp^*$ or $p^*v$, see Lemma 26
Let $M$' be a copy of $M$
for each state $\dot{q}$ in $M'$ with $\mathcal{L}(\dot{q}) \subseteq \dot{p}^*$ ($\dot{p}$ primitive) do
    Rename state $\dot{q}$ and therefore each corresponding rule and each
     recursive call on a right-hand side of a rule by $\dot{q}^{\dot{p}}$ in $M'$;
end
shiftPeriod($q$,$w$);
Function shiftPeriod($q$,$w$)
    for each rule $q(f(x_1, \ldots, x_m)) \to u_0 q_1(x_{\sigma(1)}) \ldots u_{n-1} q_n(x_{\sigma(n)}) u_n$ in $M'$
     with $f \in \mathsf{dom}(\iota(q))$ do
        Set $v, p$ such that $\mathcal{L}(q) \subseteq vp^*$ ($p$ primitive and $v$ minimal);
        Set $v_i, p_i$ such that $\mathcal{L}(q_i) \subseteq v_i p_i^*$ ($p_i$ primitive and $v_i$ minimal);
        Add a rule $q^{p_w}(f(x_1, \ldots, x_m)) \to$
         $w^{-1}v^{-1} \cdot u_0 v_1 \ldots u_{n-1} v_n u_n w \, q_1^{p_w}(x_{\sigma(1)}) q_2^{p_w}(x_{\sigma(2)}) \ldots q_n^{p_w}(x_{\sigma(n)})$ to
         $M'$;
        for each $q_i^{p_w}$ occuring on the right-hand side of the before added
         rule do
            if $q_i^{p_w}$ does not already exist in $M'$ then
                add shiftPeriod($q_i$,$u_i v_{i+1} \ldots u_{n-1} v_n u_n w$) to $M'$;
            end
        end
    end
    for each rule $q(g(x_1, \ldots, x_m)) \to \perp$ do
        Add a rule $q^{p_w}(g(x_1, \ldots, x_m)) \to \perp$ to $M'$;
    end
end
```

Algorithm 1: Shift period of an ultimately periodic state q with $\mathcal{L}(q) \subseteq vp^*$.

Lemma 27. *Let M be an LTW with an ultimately periodic state q, $\mathcal{L}(q) \subseteq vp^*$, and M' be the LTW obtained from Algorithm 1 with input M, q and $w \overset{p}{\sqsubset} p^\omega$. The following properties hold for Algorithm 1:*

- *The algorithm is correct, i.e., M and M' are equivalent and for all additionally introduced states q^p, $[\![q^p]\!](t) = w^{-1}v^{-1} \cdot [\![q]\!](t)w$ for all $t \in \mathsf{dom}(\iota(q))$.*
- *The algorithm runs in polynomial time and M' has polynomial size in M.*

Proof. Algorithm 1 is based on the observations in Lemma 24 and Corollary 5. By induction, in each recursive call of function `shiftPeriod` with parameters $\dot{q}, \dot{w}$ with $\mathcal{L}(\dot{q}) \subseteq \dot{v}\dot{p}^*$ and $\dot{w}^{-1}\dot{p}\dot{w} = p_w$ we have by Corollary 5 that

$$(u_i v_{i+1} \dots u_{n-1} v_{n-1} u_n)^{-1} \cdot p_i \cdot (u_i v_{i+1} \dots u_{n-1} v_{n-1} u_n) = \dot{p}$$

and thus

$$\dot{w}^{-1}(u_i v_{i+1} \dots u_{n-1} v_{n-1} u_n)^{-1} \cdot p_i \cdot (u_i v_{i+1} \dots u_{n-1} v_{n-1} u_n)\dot{w} = \dot{w}^{-1} \cdot \dot{p} \cdot \dot{w} = p_w$$

where $\dot{q}(f(x_1, \dots, x_m)) \to u_0 q_1(x_{\sigma(1)}) \dots u_{n-1} q_n(x_{\sigma(n)}) u_n$ is a rule in M' and $f \in \mathsf{dom}(\iota(q))$. Thus, each state $q_i^{p_w}$ occurring on the right-hand side of the rule added in function `shiftPeriod(q,w)` is already contained in M' or added in the next step. Furthermore, we show that $[\![q_i^{p_w}]\!](t) = w^{-1}v_i^{-1} \cdot [\![q_i]\!](t) \cdot w$ for all $t \in \mathsf{dom}(\iota(q))$. Let $\hat{q}(f(x_1, \dots, x_m)) \to u_0 q_1(x_{\sigma(1)}) \dots u_{n-1} q_n(x_{\sigma(n)}) u_n$ be a rule reachable from q. Then by induction there are $\hat{v}, \hat{p} \in \mathsf{A}^*$ such that $\mathcal{L}(\hat{q}) \subseteq \hat{v}\hat{p}^*$ ($\hat{p}$ primitive) and there is w such that $w^{-1} \cdot \hat{p} \cdot w = p$, cf. Lemma 24. Thus,

$$(u_i v_{i+1} \dots u_{n-1} v_n u_n)^{-1} \, v_i^{-1} \cdot \mathcal{L}(q_i) \cdot (u_i v_{i+1} \dots u_{n-1} v_n u_n) \subseteq \hat{p}^*$$

and therefore

$$w^{-1} \, (u_i v_{i+1} \dots u_{n-1} v_n u_n)^{-1} \, v_i^{-1} \cdot \mathcal{L}(q_i) \cdot (u_i v_{i+1} \dots u_{n-1} v_n u_n) \, w \subseteq w^{-1} \cdot \hat{p}^* \cdot w = p^*,$$

cf. Corollary 5.

We analyse the complexity of the algorithm. All operations in the algorithm run in polynomial time using SLPs and the abstractions $\alpha(\mathcal{L}(q_i))$, cf. Lemma 26. It therefore remains to estimate the increase in size of M'. Let k be the number of disjoint states reachable from the given state q with $\mathcal{L}(q) \subseteq vp^*$. Then in the worst case for each of these k states $q_1, \dots, q_k$ a new state q_i^p is introduced. Thus, the size of the transducer is at most doubled if all states are reachable from q. Note, however, that the size of the transducer might not be increased at all, cf. Example 19. □

With the polynomial time procedure given in Algorithm 1 at hand we strengthen Theorem 2 and show that for every LTW M an equivalent LT M' can be constructed in polynomial time that is ordered.

Theorem 4. *Let D be a DTA such that $\mathcal{L}(D) \neq \emptyset$. For every LTW M with compatible map ι, an LTW M' with compatible map ι' can be constructed in polynomial time such that*

1. *M and M' are equivalent relative to D;*
2. *M' has no trivial states;*
3. *M' is ordered.*

Proof. Using Lemma 26 we determine in polynomial time for every state q of M the value $\alpha(\mathcal{L}(q))$. With this information we remove all trivial states from M, such that we obtain in polynomial time an LTW M' with no trivial states and M and M' are equivalent relative to D, cf. Theorem 2. Thus, M' set of states Q' consists of all non-trivial states q of M and the compatible map ι' of M' is obtained from ι by restriction to Q'.

Consider a rule $q(f(x_1,\dots,x_m)) \to u_0q_1(x_{\sigma(1)})\dots u_{n-1}q_n(x_{\sigma(n)})u_n$ in M'. With Lemma 26 we determine maximal intervals $[i,j]$ such that $\mathcal{L}(q_i)u_i\dots u_{j-1}\mathcal{L}(q_j) \subseteq v_1p^*v_2$ with $v_1,p,v_2 \in \mathsf{A}^*$, i.e., $\alpha(\mathcal{L}(q_i)) \star u_i \cdot \dots \star u_{j-1} \star \alpha(\mathcal{L}(q_j)) \subseteq v \cdot \langle p' \rangle$ with $v,p' \in \mathcal{F}_{\mathsf{A}}$. These intervals are necessarily disjoint. Let $q_i(x_{\sigma(i)})u_i\dots u_{j-1}q_j(x_{\sigma(j)})$ be such a maximal interval on the right-hand side of a rule that is not ordered. We first assume that $\mathcal{L}(q_i)u_i\dots u_{j-1}\mathcal{L}(q_j) \subseteq vp^*$. Let $v_k,p_k \in \mathsf{A}^*$ be such that $\mathcal{L}(q_k) \subseteq v_kp_k^*$, $k \in \{i,\dots,j\}$. With $\texttt{shiftPeriod(}q_k\texttt{, }w_k\texttt{)}$ in Algorithm 1 we obtain states q_k^p with $w_k = u_kv_{k+1}\dots u_{j-1}v_j$ such that $[\![q_k]\!](t) = v_kw_k[\![q_k^{w_k}]\!](t)\cdot w_k^{-1}$. By Corollary 5 we have that

$$[\![q_i]\!](t_i)u_i\dots u_j[\![q_j]\!](t_j) = v_iu_i\dots v_{j-1}u_{j-1}v_j[\![q_i^p]\!](t_i)\cdot[\![q_j^p]\!](t_j)$$

for all $t_k \in \mathsf{dom}(\iota(q_k))$, $k \in \{i,\dots,j\}$. Additionally, $\mathcal{L}(q_i^p),\dots,\mathcal{L}(q_j^p) \subseteq p_j^*$ and thus the recursive calls of the states q_k^p can be arbitrarily permuted. Therefore, we rewrite the unordered interval $q_i(x_{\sigma(i)})u_i\dots u_{j-1}q_j(x_{\sigma(j)})$ by

$$v_iu_i\dots v_{j-1}u_{j-1}v_jq_{\pi(i)}^{w_i}(x_{\sigma(\pi(i))})\dots q_{\pi(j)}^{w_j}(x_{\sigma(\pi(j))})$$

where $\pi : [i,j] \to [i,j]$ is the permutation such that $\sigma(\pi(i)) < \dots < \sigma(\pi(j))$. For each additionally introduced state q^p we extend the compatible mapping ι' by $\iota'(q^p) = \iota'(q)$.

Assume now that the unordered interval $q_i(x_{\sigma(i)})u_i\dots u_{j-1}q_j(x_{\sigma(j)})$ on the right-hand side of a rule produces a language of the form $v'p^*v$ with v minimal and $v \overset{p}{\not\sqsubseteq} p$. As we removed all trivial states the constant suffix v is produced by state q_j, i.e., $\mathcal{L}(q_j) \subseteq p_j^*v$ and $\mathcal{L}(q_i)u_i\dots u_{j-2}\mathcal{L}(q_j)u_j \subseteq \dot{v}\dot{p}^*$. We therefore remove the constant suffix v from $\mathcal{L}(q_j)$ as follows. We build the reversed state q_j^r with $[\![q_j^r]\!](t) = ([\![q_j]\!](t))^r$ for all $t \in \mathsf{dom}(\iota(q_j))$, i.e., $\mathcal{L}(q_j^r) \subseteq v^r(p_j^r)^*$. Using Algorithm 1 we build state $(q_j^r)^{p_j}$ such that $v^r[\![(q_j^r)^{p_j}]\!](t) = [\![q_j^r]\!](t)$ for all $t \in \mathsf{dom}(\iota(q_j))$, thus $\mathcal{L}((q_j^r)^{p_j}) \subseteq p_j^*$. For state $(q_j^r)^{p_j}$ we build the reversed state and call it $q_j^{p_j}$. Then $[\![q_j^{p_j}]\!](t) = ([\![(q_j^r)^{p_j}]\!](t))^r$ for all $t \in \mathsf{dom}(\iota(q_j))$ and therefore

$$\begin{aligned}(v^r[\![(q_j^r)^{p_j}]\!](t))^r &= ([\![(q_j^r)^{p_j}]\!](t))^rv\\ &= [\![q_j^{p_j}]\!](t)v\\ &= ([\![q_j^r]\!](t))^r\\ &= [\![q_j]\!](t)\end{aligned}$$

I.e., $[\![q_j^{p_j}]\!](t)v = [\![q_j]\!](t)$ for all $t \in \mathsf{dom}(\iota(q_j))$ and therefore

$$\begin{array}{rl} & [\![q_i]\!](t_i)u_i \dots u_{j-2}[\![q_{j-1}]\!](t_{j-1})u_{j-1}[\![q_j]\!](t_j) \\ = & [\![q_i]\!](t_i)u_i \dots u_{j-2}[\![q_{j-1}]\!](t_{j-1})u_{j-1}[\![q_j^{p_j}]\!](t_j)v \end{array}$$

for all $t_k \in \mathsf{dom}(\iota(q_k))$. Thus, $\mathcal{L}(q_i)u_i \dots u_{j_1}\mathcal{L}(q_j^{p_j}) \subseteq v'p^*$ and we follow the before discussed case to order the interval.
In total, we arrive at an ordered LTW M' with M and M' are equivalent relative to D.

It remains to analyse the runtime of the rewriting of the unordered rules. As for each interval $[i, j]$ Algorithm 1 is called at most $j - i + 2$ times and each call increases the size of M' at most polynomially. In case reversed states have to be constructed the construction still runs in polynomial time and M' is only increased by the additional states produced by Algorithm 1 as the intermediate constructed reversed states are not contained in M'. In total, the overall procedure runs in polynomial time. Note, that some states $q_k^{w_k}$ may already exist in M'. This can be tested before calling Algorithm 1 such that no additional states are added in this case. Further optimizations might be possible; if all recursive calls of a state q_k on the right-hand sides of the rules are replaced then the state q_k can be removed from M', cf. Example 19. □

We give a comprehensive example on how to compute an ordered LTW. Especially, the example covers the following two cases, (i) the interval that has to be ordered is of the form $v'p^*v$ with $v \overset{p}{\not\sqsubseteq} p$ and therefore an additional step is needed to remove the constant suffix v; (ii) the size of the transducer increases due to the ordering.

Example 20. *We consider the ranked input alphabet $\{f^{(2)}, k^{(2)}, g^{(1)}, h^{(0)}\}$ where the superscripts indicate the rank. Let D be a DTA with start state h_0 and the transition function $\delta_D = \{(h_0, f) \mapsto h_1h_1, (h_0, k) \mapsto h_1h_1, (h_1, g) \mapsto h_1, (h_1, h) \mapsto \varepsilon\}$.*
LTW M has the axiom $q_0(x_0)$ and the following rules:

$$\begin{array}{rlrl} q_0(f(x_1,x_2)) & \to q_1(x_2)\, a\, q_2(x_1) & q_0(k(x_1,x_2)) & \to q_1(x_1)\, q_2(x_2) \\ q_1(g(x_1)) & \to q_1(x_1)\, ab & q_1(h) & \to a \\ q_2(g(x_1)) & \to ba\, q_2(x_1) & q_2(h) & \to d \end{array}$$

All omitted rules have right-hand side $\bot$. The compatible map ι is then given by $\iota = \{q_0 \mapsto h_0, q_1 \mapsto h_1, q_2 \mapsto h_2\}$.
Thus, the state q_0 of M produces either a language of the form $a(ab)^{k+\ell}d$ if the input tree is of the form $f(g^k(h), g^\ell(h))$ or a language of the form $a(ab)^k(ba)^\ell d$ if the input tree is of the form $k(g^k(h), g^\ell(h))$. As the former language is periodic of the form $ad \cdot \langle d^{-1} \cdot ab \cdot d\rangle$, respectively $a(ab)^*d$, the rule with left-hand side $q_0(f(x_1, x_2))$ is not ordered. Following the proof of Theorem 4 we first remove the constant suffix d

of the language $(ba)^*d$ produced by state q_2. We obtain state q_2^{ba} with the following rules

$$\begin{array}{lcl}
q_2^{ab}(g(x_1)) & \to & q_2^{ba}(x_1)\, ba\, d \cdot d^{-1} \\
 & \to & q_2^{ba}(x_1)\, ba \\
q_2^{ba}(h) & \to & d \cdot d^{-1} \\
 & \to & \varepsilon
\end{array}$$

For all $t \in \mathsf{dom}(\iota(q_2))$ the following holds, $[\![q_2]\!](t) = [\![q_2^{ba}]\!](t)\, d$ and we obtain the following set of rules

$$\begin{array}{rl@{\qquad}rl}
q_0(f(x_1,x_2)) & \to q_1(x_2)\, a\, q_2^{ab}(x_1)\, d & q_0(k(x_1,x_2)) & \to q_1(x_1)\, q_2^{ba}(x_2)\, d \\
q_1(g(x_1)) & \to q_1(x_1)\, ab & q_1(h) & \to a \\
q_2^{ab}(g(x_1)) & \to q_2^{ba}(x_1)\, ba & q_2^{ab}(h) & \to \varepsilon
\end{array}$$

Now, consider the interval $q_1(x_2)\, a\, q_2^{ba}(x_1)$ of the rule with left-hand side $q_0(f(x_1,x_2))$ with $\mathcal{L}(q_1\, a\, q_2^{ba}) \subseteq a(ab)^*\, a\, (ba)^* = aa\, (ba)^*$. For state q_1 with a output language of the form $a(ab)^*$ we build state q_1^{ba} with $\mathcal{L}(q_1^{ba}) \subseteq (ba)^*$. With Algorithm 1 and $w = a$ we obtain the following rules

$$\begin{array}{lcl}
q_1^{ba}(g(x_1)) & \to & a^{-1}a^{-1} \cdot a\, ab\, a\, q_1^{ba}(x_1) \\
 & \to & ba\, q_1^{ba}(x_1) \\
q_1^{ba}(h) & \to & a^{-1}a^{-1} \cdot a\, a \\
 & \to & \varepsilon
\end{array}$$

Thus, for all $t \in \mathsf{dom}(\iota(q_1))$, $[\![q_1^{ba}]\!](t) = a^{-1}a^{-1} \cdot [\![q_1]\!](t) \cdot a$ and the rule with left-hand side $q_0(f(x_1,x_2))$ is replaced by the ordered rule $q_0(f(x_1,x_2)) \to aa\, q_2^{ba}(x_1)\, q_1^{ba}(x_2)\, d$. In total, we arrive at the following rules

$$\begin{array}{rl@{\qquad}rl}
q_0(f(x_1,x_2)) & \to aa\, q_2^{ba}(x_1)\, q_1^{ba}(x_2)\, d & q_0(k(x_1,x_2)) & \to q_1(x_1)\, q_2^{ba}(x_2)\, d \\
q_1(g(x_1)) & \to q_1(x_1)\, ab & q_1(h) & \to a \\
q_1^{ab}(g(x_1)) & \to ba\, q_1^{ba}(x_1) & q_1^{ab}(h) & \to \varepsilon \\
q_2^{ab}(g(x_1)) & \to q_2^{ba}(x_1)\, ba & q_2^{ab}(h) & \to \varepsilon
\end{array}$$

We observe that state q_1 can not *be removed because of the recursive call of q_1 on the right-hand side of the rule with left-hand side $q_0(k(x_1,x_2))$. The size of M is therefore increased while building the ordered form.*

Equivalence of ordered LTWs without trivial states can be decided in polynomial time by first checking whether they are same-ordered (Lemma 23) and if they are same-ordered then checking equivalence via the morphism equivalence, cf. Corollary 4.

3.4 Summary

In this chapter we showed that equivalence of linear tree transducers with output in the free group can be decided in polynomial time. Therefore, we introduced an ordered form that guarantees that equivalent transducers process their input in the same order, i.e., they are same-ordered. Then, a context-free grammar (over the free group) simulating all parallel runs of two linear same-ordered tree transducers can be constructed and equivalence is decidable via morphism equivalence on context-free grammars. The decision procedure follows the same lines as in [BP16] where equivalence of linear tree-to-word transducers was shown to be decidable in polynomial time. But the result given here is stronger as it subsumes the result from [BP16]. Additionally, we showed that the use of inverses lead to easier and more direct proofs underlying the ordered form that is the analogon to the partial normal form in [BP16]. We recapped the results for linear tree-to-word transducer and adjusted them to the new ordered form in Section 3.3. Without the use of inverses additional states might be introduced such that the size of the equivalent transducer in ordered form is polynomially larger compared to the original transducer.

4 Balancedness

Structured text requires that pairs of opening and closing brackets are properly nested. This applies to text representing program code as well as to XML or HTML documents. Again, as introduced in Chapter 2 we call properly nested words over an alphabet $\mathsf{B} = \mathsf{A} \cup \overline{\mathsf{A}}$ of opening (A) and closing ($\overline{\mathsf{A}} = \{\overline{a} \mid a \in \mathsf{A}\}$) brackets *balanced*. Thus, a word $w \in \mathsf{B}^*$ is balanced if $\rho(w) = \varepsilon$. Balanced words need not necessarily be constructed in a structured way. Therefore deciding whether the set of words produced by some kind of text processor consists of balanced words only is a non-trivial problem. For the case of a single pair of brackets, i.e., $|\mathsf{A}| = 1$, and context-free languages, decidability of this problem has been settled by Knuth [Knu67] and a polynomial time algorithm is presented by Minamide and Tozawa [MT06]. Recently, these results were generalized to the output languages of monadic second-order logic (MSO) definable tree-to-word transductions [MS18]. In contrast, balancedness of non-determinsitc recursive program schemes was shown to be undecidable [Kob19]. The case when the alphabet B consists of *multiple* pairs of different kind of brackets ($|\mathsf{A}| > 1$), though, seems to be more intricate than the case of a single pair of brackets. Still, balancedness for context-free languages was shown to be decidable by Berstel and Boasson [BB02] where a polynomial time algorithm again has been provided by Tozawa and Minamide [TM07]. Whether or not these results can be generalized to MSO definable transductions as e.g. done by finite copying macro tree transducers with regular look-ahead, remains as an open problem. Here, we provide a first step to answering this question.

We consider deterministic 2-copy tree-to-word transducers (2-TW) which process their input at most twice by calling in their axioms at most two *linear* tree-to-word transducers on the input. The output languages of *linear* deterministic tree-to-word transducers are context-free, but this does not need to be the case for 2-TWs. 2-TWs form a subclass of MSO definable transductions which allows to specify transductions such as *prepending* an XML document with the list of its section headings, or *appending* such a document with the list of figure titles. For 2-TWs we show that balancedness is decidable in *polynomial time*. Let T_1 and T_2 be the two linear tree-to-word transducers called in the axiom of a 2-TW M, i.e., M performs the transduction $T_1(t)T_2(t)$ for an input tree t. Balancedness of M is decided by first checking that T_1 and $\overline{T_2}$ produces well-formed output only. Here, $\overline{T_2}$ denotes the transduction obtained if the output of T_2 on an input tree t is inverted. In a second step we check whether T_1 and $\overline{T_2}$ are equivalent *after* *reduction.* The output language of a linear tree-to-word transducer is context-free and an respective context-free grammar for this language can directly be read from the rules of the transducer. This allows us to use the polynomial decision procedure for well-formedness from Chapter 2 to decide whether T_1 and $\overline{T_2}$ produce only well-formed output. Then it remains to decide equivalence of T_1 and $\overline{T_2}$ considering the reduced outputs of the transducers. For example, if $T_1(t)$ produces the output $aba\overline{a}$ and $T_2(t)$ the output $ba\overline{a}\,\overline{bb}\,\overline{a}$ for an input tree t, then $aba\overline{a}$ and $\overline{ba\overline{a}\,\overline{bb}\,\overline{a}} = abba\overline{a}\,\overline{b}$ are not equivalent as words but equivalent after reduction, $\rho(aba\overline{a}\,) = ab = \rho(abba\overline{a}\,\overline{b}\,)$. However, as well-formedness of the (inverted) outputs of T_1 (T_2) is checked before equivalence of the outputs can be decided over the free group where inverted letters are interpreted as inverses. In general, the rewriting rules of the involutive monoid over A and the free group over A may lead to different reduced words, as in the involutive monoid $\overline{a}\,a$ does not reduce further while in the free group $a^{-1}a$ is equivalent to the empty word. We recap that equivalence of linear tree transducers over the free group is obtained via an ordered form. We show that this ordered form can be obtained for linear tree-to-word transducers where the output is interpreted over the involutive monoid. In fact, the canonical form of (ultimately) periodic states on that the ordering is based can be used to remove inverted letters from the rules of (ultimately) periodic states. Thus, we show that the rules of a state q with the output language $\mathcal{L}(q)$ and $\rho(\mathcal{L}(q))$ is ultimately periodic can be rewritten such that $\mathcal{L}(q) = \rho(\mathcal{L}(q))$.

Outline Section 4.1 introduces linear tree-to-word transducers that are interpreted over the involutive monoid and 2-TWs. In Section 4.2 we present the main result of this chapter – the reduction from balancedness of 2-TWs to well-formedness and equivalence of linear tree-to-word transducers. Additionally, we analyse in Section 4.3 how inverted letters in the rules of (ultimately) periodic states can be eliminated and thus an ordered linear tree-to-word transducer over the involutive monoid can be constructed.

4.1 2-copy Tree-to-Word Transducers

We define a (total deterministic) *linear* tree-to-word transducer (LTW_B for short) $M = (\Sigma, \mathsf{B}, Q, S, R)$ over the free involutive monoid generated by a finite alphabet A where $\overline{\mathsf{A}} = \{\overline{a} \mid a \in \mathsf{A}\}$ is the alphabet of inverted letters derived from A and $\mathsf{B} = \mathsf{A} \cup \overline{\mathsf{A}}$, cf. Section 2.1. The definition is analogously to LTs but the output alphabet is B and the output can thus be interpreted over the involutive monoid. As before we use Latin letters $u, v, \ldots$ to denote words over A and Greek letters $\alpha, \beta, \gamma, \ldots$ to denote words over B. Σ is a finite ranked input alphabet, B is the finite (unranked) output alphabet, Q is a finite set of states, the axiom S is of the form γ_0 or $\gamma_0 q(x_1)\gamma_1$ with $\gamma_0, \gamma_1 \in \mathsf{B}^*$ and R is a set of rules of the form

$$q(f(x_1, \ldots, x_m)) \to \gamma_0 q_1(x_{\sigma(1)})\gamma_1 \ldots q_n(s_{\sigma(n)})\gamma_n$$

with $q, q_i \in Q$, $f \in \Sigma$, $\gamma_i \in \mathsf{B}^*$, $n \leq m$ and σ an injective mapping from $\{1, \ldots, n\}$ to $\{1, \ldots, m\}$. As we consider total deterministic transducers there is exactly one rule for each pair $q \in Q$ and $f \in \Sigma$. Again, we consider an $\mathsf{LTW_B}$ together with a top-down deterministic domain automaton D and a compatible map ι, cf. Section 3.1. A 2-*copy tree-to-word transducer* (2-TW) is a tuple $N = (\Sigma, \mathsf{B}, Q, S, R)$ that is defined in the same way as an $\mathsf{LTW_B}$ but the axiom S is of the form γ_0 or of the form $\gamma_0 q_1(x_1)\gamma_1 q_2(x_1)\gamma_2$, with $\gamma_i \in \mathsf{B}^*$. We define the semantics $[\![q]\!] : \mathcal{T}_\Sigma \to \mathsf{B}^*$ of a state q with rule $q(f(t_1, \ldots, t_m)) \to \gamma_0 q_1(t_{\sigma(1)})\gamma_1 \ldots \gamma_{n-1} q_n(t_{\sigma(n)})\gamma_n$ inductively by

$$[\![q]\!](f(t_1, \ldots, t_m)) = \gamma_0 [\![q_1]\!](t_{\sigma(1)})\gamma_1 \cdots \gamma_{n-1}[\![q_n]\!](t_{\sigma(n)})\gamma_n$$

The semantics $[\![M]\!]$ of an $\mathsf{LTW_B}$ M with axiom γ_0 is given by $\rho(\gamma_0)$; if the axiom is of the form $\gamma_0 q(x_1)\gamma_1$ it is defined by $\gamma_0 [\![q]\!](t)\gamma_1$ for all $t \in \mathcal{T}_\Sigma$; while the semantics $[\![N]\!]$ of a 2-TW N with axiom γ_0 is again given by γ_0 and for axiom $\gamma_0 q_1(x_1)\gamma_1 q_2(x_1)\gamma_2$ it is defined by $\gamma_0 [\![q_1]\!](t)\gamma_1 [\![q_2]\!](t)\gamma_2$ for all $t \in \mathcal{T}_\Sigma$. For a state q we define the output language $\mathcal{L}(q) = \{[\![q]\!](t) \mid t \in \mathcal{T}_\Sigma\}$; For a 2-TW M we let $\mathcal{L}(M) = \{[\![M]\!](t) \mid t \in \mathcal{T}_\Sigma\}$.

Additionally, we may assume w.l.o.g. that all states q of an $\mathsf{LTW_B}$ are *non-trivial* after reduction, i.e., $\rho(\mathcal{L}(q))$ contains at least two words. We call a 2-TW M *balanced* if $\rho(\mathcal{L}(M)) = \{\varepsilon\}$. We say an $\mathsf{LTW_B}$ M is *well-formed* if $\rho(\mathcal{L}(M)) \subseteq \mathsf{A}^*$. To shorten notation we say that state q fulfills some property if the reduced output language $\rho(\mathcal{L}(q))$ fulfills this property. For example, q is balanced (well-formed, ultimately periodic, ...) if $\rho(\mathcal{L}(q))$ is balanced (well-formed, ultimately periodic, ...). We use $\overline{q}$ to denote the inverse transduction of q which is obtained from a copy of the transitions reachable from q by involution of the right-hand side of each rule. For example, given a rule $q(f(x_1, \ldots, x_m)) \to \gamma_0 q_1(x_{\sigma(1)})\gamma_1 \ldots \gamma_{n-1} q_n(x_{\sigma(n)})\gamma_n$ we obtain $\overline{q}\,(f(x_1, \ldots, x_m)) \to \overline{\gamma_n}\,\overline{q_n}\,(x_{\sigma(n)})\overline{\gamma_{n-1}} \ldots \overline{\gamma_1}\,\overline{q_1}\,(x_{\sigma(1)})\overline{\gamma_0}$. As a consequence, $[\![\overline{q}\,]\!](t) = \overline{[\![q]\!](t)}$ for all $t \in \mathcal{T}_\Sigma$, and thus, $\mathcal{L}(\overline{q}\,) = \overline{\mathcal{L}(q)}$. We say that two states q, q' are *equivalent* iff for all $t \in \mathcal{T}_\Sigma$, $\rho([\![q]\!](t)) = \rho([\![q']\!](t))$. Accordingly, two 2-TWs M, M' are equivalent iff for all $t \in \mathcal{T}_\Sigma$, $\rho([\![M]\!](t)) = \rho([\![M']\!](t))$.

4.2 Balancedness of 2-TWs

Let M denote a 2-TW. W.l.o.g., we assume that the axiom of M is of the form $q_1(x_1)q_2(x_1)$ for two states q_1, q_2. If this is not yet the case, an equivalent 2-TW with this property can be constructed in polynomial time. We reduce balancedness of M to decision problems for *linear* tree-to-word transducers alone.

Proposition 1. *The 2-TW M is balanced iff the following two properties hold:*

- *Both* $\mathcal{L}(q_1)$ *and* $\overline{\mathcal{L}(q_2)}$ *are well-formed;*
- q_1 *and* $\overline{q_2}$ *are equivalent.*

Proof. Assume first that M with axiom $q_1(x_1)q_2(x_1)$ is balanced, i.e., $\rho(\mathcal{L}(M)) = \varepsilon$. Then for all w', w'' with $w = w'w'' \in \mathcal{L}(M)$, $\rho(w') = u \in \mathsf{A}^*$ and $\rho(w'') = \overline{u}$. Thus, both $\mathcal{L}(q_1)$ and $\overline{\mathcal{L}(q_2)}$ consist of well-formed words only. Assume for a contradiction that q_1 and $\overline{q_2}$ are not equivalent. Then there is some $t \in \mathcal{T}_\Sigma$ such that $[\![q_1]\!](t) \overset{\rho}{\neq} [\![\overline{q_2}]\!](t)$. Let $\rho([\![q_1]\!](t)) = u \in \mathsf{A}^*$ and $\rho([\![\overline{q_2}]\!](t)) = \overline{[\![q_2]\!](t)} = v$ with $v \in \mathsf{A}^*$ and $u \neq v$. Then $\rho([\![q_1]\!](t)[\![q_2]\!](t)) = \rho(u\overline{v}) \neq \varepsilon$ as $u \neq v$, $u, v \in \mathsf{A}^*$. Since M is balanced, this is not possible.

Now, assume that $\mathcal{L}(q_1)$ and $\overline{\mathcal{L}(q_2)}$ are well-formed, i.e., for all $t \in \mathcal{T}_\Sigma$, $\rho([\![q_1]\!](t)) \in \mathsf{A}^*$ and $\rho([\![q_2]\!](t)) \in \overline{\mathsf{A}}^*$. Additionally assume that q_1 and $\overline{q_2}$ are equivalent, i.e., for all $t \in \mathcal{T}_\Sigma$, $[\![q_1]\!](t) \overset{\rho}{=} [\![\overline{q_2}]\!](t) \overset{\rho}{=} \overline{[\![q_2]\!](t)}$. Therefore for all $t \in \mathcal{T}_\Sigma$, $[\![q_2]\!](t) \overset{\rho}{=} \overline{[\![q_1]\!](t)}$ and hence,

$$\rho([\![q_1]\!](t)[\![q_2]\!](t)) = \rho([\![q_1]\!](t)\overline{[\![q_1]\!](t)}) = \varepsilon$$

Therefore, the 2-TW M must be balanced. □

The output languages of states q_1 and $\overline{q_2}$ are generated by means of context-free grammars of polynomial size. Let $\mathcal{L}(G)$ denote the language produced by a context-free grammar G and q be a state of an $\mathsf{LTW_B}$. Then the rules for a CFG G with $\mathcal{L}(G) = \mathcal{L}(q)$ can be directly read from the rules of state q.

Example 21. *Consider* $\mathsf{LTW_B}$ M *with input alphabet* $\Sigma = \{f^{(2)}, g^{(0)}\}$ *(the superscript denotes the rank), output alphabet* $\mathsf{B} = \{a, \overline{a}\}$*, axiom* $q_3(x_1)$ *and rules*

$$\begin{array}{ll} q_3(f(x_1,x_2)) \to aq_2(x_1)q_2(x_2)\overline{a} & q_2(g) \to \varepsilon \\ q_2(f(x_1,x_2)) \to aq_1(x_1)q_1(x_2)\overline{a} & q_2(g) \to \varepsilon \\ q_1(f(x_1,x_2)) \to q_3(x_1)q_3(x_2) & q_1(g) \to aa \end{array}$$

We obtain a CFG *producing exactly the output language of* M *by nondeterministically guessing the input symbol, i.e. the state* q_i *becomes the nonterminal* W_i*. The axiom of this* CFG *is then* W_3*, and as rules we obtain*

$$W_3 \to aW_2W_2\overline{a} \mid \varepsilon \quad W_2 \to aW_1W_1\overline{a} \mid \varepsilon \quad W_1 \to W_3W_3 \mid aa$$

Note that the rules of M *and the associated* CFG *use a form of iterated squaring, i.e.* $W_3 \to^2 W_3^4$*, that allows to encode potentially exponentially large outputs within the rules (see also Example 13). In general, words thus have to be stored in compressed form as* SLP*s [Loh12].*

Therefore, Theorem 1 of Section 2.3 implies that well-formedness of q_1, $\overline{q_2}$ can be decided in polynomial time. Accordingly, it remains to consider the equivalence problem for well-formed $\mathsf{LTW_B}$s. Since the two transducers in question are well-formed, they are equivalent as $\mathsf{LTW_B}$s iff they are equivalent when their outputs are considered over the free group $\mathcal{F}_\mathsf{A}$ where inverted letters are interpreted as inverses. In the free group $\mathcal{F}_\mathsf{A}$, we have that $a^{-1}a$ reduces to ε — which does not hold in our rewriting system. If sets $\mathcal{L}(q_1), \mathcal{L}(\overline{q_2})$ of outputs for q_1 and $\overline{q_2}$, however, are well-formed, it follows for all $u \in \mathcal{L}(q_1), v \in \mathcal{L}(\overline{q_2})$ that $\rho(u\overline{v}) = \rho(\rho(u)\,\rho(\overline{v}))$ cannot contain $\overline{a}\,a$. Therefore, $\rho(u\overline{v}) = \varepsilon$ iff uv^{-1} is equivalent to ε over the free group $\mathcal{F}_\mathsf{A}$. In Theorem 3 we have proven that equivalence of LTs is decidable in polynomial time. Thus, we obtain our main theorem.

Theorem 5. *Balancedness of* 2-*TWs is decidable in polynomial time.*

4.3 Eliminating Inverted Letters in Well-formed $\mathsf{LTW_B}$s

We reduced balancedness of 2-TWs to well-formedness and equivalence of well-formed $\mathsf{LTW_B}$s in Section 4.2. Given the well-formedness of the $\mathsf{LTW_B}$s equivalence can be decided over the free group. In this section we present an alternative approach [LLS19] that we discovered before showing that equivalence of LTs is decidable in polynomial time. The underlying idea is the same – we order the $\mathsf{LTW_B}$s such that they are same-ordered and then decide equivalence of same-ordered $\mathsf{LTW_B}$s (over the free group). A state q of an $\mathsf{LTW_B}$is called ultimately periodic if $\rho(\mathcal{L}(q)) \subseteq vp^*$ or $\rho(\mathcal{L}(q)) \subseteq p^*v$, $v, p \in \mathsf{A}^*$. The key observation is that all inverted letters occurring in the output of ultimately periodic states can be eliminated. Thus, we can rewrite the rules of an ultimately periodic state q such that no inverted letters occur on the right-hand sides, i.e., $\rho(\mathcal{L}(q)) = \mathcal{L}(q))$. This rewriting is analogously to the canonical form $q(f(x_1, \ldots, x_m)) \to p^i q_1(x_{\sigma(1)}) \ldots q_n(x_{\sigma(n)})$ used in Algorithm 1 for LTWs.
The procedure of removing inverted letters can analogously applied to context-free grammars since the output of linear tree transducers produce context-free languages and there is a direct correspondence between these two models, cf. Example 21. This is of special interest as it is still an open problem whether there is a polynomial time transformation from a well-formed context-free grammar G over B to a context-free grammar G' over A such that $\rho(\mathcal{L}(G)) = \mathcal{L}(G)$ and the derivations are in bijection.
Let M be an $\mathsf{LTW_B}$ that is well-formed and q be a state of M. Then q does not have to be well-formed but bounded well-formed, i.e., there is r_q such that $r_q\mathcal{L}(q)$ is well-formed. For example, an $\mathsf{LTW_B}$ with axiom $aq(x_0)$ and state q producing the language $\overline{a}\,(ba)^n\overline{a}$, $n \in \mathbb{N}$ is well-formed while q is only bounded well-formed with $r_q = a$.

Definition 6. *We call a well-formed $\mathsf{LTW_B}$ M* suffix-empty *if for all states q in M, $\rho(\mathcal{L}(q)) \in \mathsf{A}^*$ and $\mathsf{lcs}(\rho(\mathcal{L}(q))) = \varepsilon$.*

We show that the rules of a well-formed $\mathsf{LTW_B}$ M can be adjusted such that *every state* produces a well-formed language with empty longest common suffix.

Lemma 28. *Let M be a well-formed $\mathsf{LTW_B}$ and D be a DTA. Then an equivalent $\mathsf{LTW_B}$ M' w.r.t. D can be constructed in polynomial time such that M' is suffix-empty.*

Proof. Let q be a state of a well-formed $\mathsf{LTW_B}$ M. Then $\mathcal{L}(q)$ is bounded well-formed and we can construct in polynomial time a CFG G such that $\mathcal{L}(G) = \mathcal{L}(q)$. From G we compute (SLPs) for the minimal word $r_q \in \mathsf{A}^*$ such that $r_q\mathcal{L}(q)$ is well-formed Lemma 4) and s_q the reduced longest common suffix of $\rho(r_q\mathcal{L}(q))$ (Lemma 11). For every rule $q(f(x_1,\dots,x_m)) \to \gamma_0 q_1(x_{\sigma(1)})\gamma_1 \dots q_n(x_{\sigma(n)})\gamma_n$ we add a rule

$$q'(f(x_1,\dots,x_m)) \to r_q\gamma_0\overline{r_{q_1}}\, q_1'(x_{\sigma(1)})s_{q_1}\gamma_1 \dots \overline{r_{q_n}}\, q_n'(x_{\sigma(n)})s_{q_n}\gamma_n\overline{s_q}$$

to M'. Let $S = \gamma_0 q(x_1)\gamma_1$ be the axiom in M, then we add the axiom $S' = \gamma_0\overline{r_q}\, q'(x_1)s_q\gamma_1$ to M'.

Let q be a state in M. We prove by induction over the size of the input tree that for all $t \in \mathsf{dom}(\iota(q))$, $\llbracket q'\rrbracket(t) = r_q\llbracket q\rrbracket(t)\overline{s_q}$. For the base case let $t = h \in \Sigma^{(0)}$ and $q(h) \to \gamma_0$ be the corresponding rule in M. Then $\llbracket q'\rrbracket(h) = r_q\gamma_0\overline{s_q} = r_q\llbracket q\rrbracket(h)\overline{s_q}$. Let $f(x_1,\dots,x_m) \in \mathsf{dom}(\iota(q))$ and $q(f(x_1,\dots,x_m)) \to \gamma_0 q_1(x_{\sigma(1)})\gamma_1 \dots q_n(x_{\sigma(n)})\gamma_n$ be the corresponding rule in M. Then

$$\begin{array}{cl}
 & \llbracket q'\rrbracket(f(x_1,\dots,x_m)) \\
= & r_q\gamma_0\overline{r_{q_1}}\,\llbracket q_1'\rrbracket(x_{\sigma(1)})s_{q_1}\gamma_1 \dots \overline{r_{q_n}}\,\llbracket q_n'\rrbracket(x_{\sigma(n)})s_{q_n}\gamma_n\overline{s_q} \\
= & r_q\gamma_0\overline{r_{q_1}}\, r_{q_1}\llbracket q_1\rrbracket(x_{\sigma(1)})\overline{s_{q_1}}\, s_{q_1}\gamma_1 \dots \overline{r_{q_n}}\, r_{q_n}\llbracket q_n\rrbracket(x_{\sigma(n)})\overline{s_{q_n}}\, s_{q_n}\gamma_n\overline{s_q} \\
\overset{\rho}{=} & r_q\gamma_0\llbracket q_1\rrbracket(x_{\sigma(1)})\gamma_1 \dots \llbracket q_n\rrbracket(x_{\sigma(n)})\gamma_n\overline{s_q} \\
= & r_q\llbracket q\rrbracket(f(x_1,\dots,x_m))\overline{s_q}
\end{array}$$

Let $S = \gamma_0 q(x_1)\gamma_1$ be the axiom in M. Then for all $t \in \mathsf{dom}(\iota(q))$, $\llbracket M'\rrbracket = \gamma_0\overline{r_q}\,\llbracket q'\rrbracket(t)s_q\gamma_1 = \gamma_0\overline{r_q}\, r_q\llbracket q\rrbracket(t)\overline{s_q}\, s_q\gamma_1 = \gamma_0\llbracket q\rrbracket(t)\gamma_1 = \llbracket M\rrbracket$. Thus, M and M' are equivalent w.r.t. D. From the construction it directly follows that for all states q in M', $\rho(\mathcal{L}(q)) \subseteq \mathsf{A}^*$ and $\mathsf{lcs}(\rho(\mathcal{L}(q))) = \varepsilon$. □

Example 22. *Consider a well-formed $\mathsf{LTW_B}$ with axiom $q(x_1)$, states q, q' and the rules*

$$\begin{array}{lcllcl}
q(f(x_1)) & \to & ab\, q'(x_1) & q'(f(x_1)) & \to & \overline{ab}\, q(x_1)\, ab \\
q(g) & \to & ab & q'(g) & \to & ab
\end{array}$$

We omit a corresponding DTA. Then $r_q = \varepsilon$, $r_{q'} = ab$ are the minimal words such that $r_q\mathcal{L}(q)$ and $r_{q'}\mathcal{L}(q')$, respectively, are well-formed and $s_q = ab$ and $s_{q'} =$

abab are the longest common suffixes of $\rho(r\mathcal{L}(q))$ and $\rho(r'\mathcal{L}(q'))$, respectively. We therefore obtain

$$\begin{array}{lll@{\qquad}lll}
q(f(x_1)) & \to \varepsilon\, ab\, \overline{ab}\; q'(x_1)\, abab\, \overline{ab} & & q'(f(x_1)) & \to ab\, \overline{ab}\; \overline{\varepsilon}\; q(x_1)\, ab\, ab\, \overline{abab} \\
 & \to q'(x_1)\, ab & & & \to q(x_1) \\
q(g) & \to \varepsilon\, ab\, \overline{ab} & & q'(g) & \to ab\, ab\, \overline{abab} \\
 & \to \varepsilon & & & \to \varepsilon
\end{array}$$

The semantics did not change through the rewriting, but $\rho(\mathcal{L}(q)), \rho(\mathcal{L}(q')) \subseteq \mathsf{A}^*$ and $\mathsf{lcs}(\rho(\mathcal{L}(q))) = \mathsf{lcs}(\rho(\mathcal{L}(q'))) = \varepsilon$.

Let M be a well-formed and suffix-empty $\mathsf{LTW_B}$. We observe that M does not contain any ultimately periodic state q of the form $\rho(\mathcal{L}(q)) \subseteq p^*v$ with $v \neq \varepsilon$ as then $\mathsf{lcs}(\rho(\mathcal{L}(q))) \neq \varepsilon$, a contradiction to the suffix-empty property of M. Let q be an ultimately periodic state of M with $\rho(\mathcal{L}(q)) \subseteq vp^*$. We show that if the rules of q are in the canonical form $q(f(x_1, \dots, x_m)) \to \rho(\gamma_0)\, q_1(x_{\sigma(1)}) \dots q_n(x_{\sigma(n)})$, i.e., first all (reduced) output is produced and then the consecutive recursive calls on the subtrees follow (cf. Algorithm 1), then no inverted letters occur in the output.

Lemma 29. *Let M be a well-formed $\mathsf{LTW_B}$ and D be a DTA. Then an $\mathsf{LTW_B}$ M' can be constructed in polynomial time such that for all ultimately periodic states q, $\rho(\mathcal{L}(q)) = \mathcal{L}(q)$, i.e., q does not produce any inverted letter in the output and M and M' are equivalent w.r.t. D.*

Proof. W.l.o.g. we assume that M is suffix-empty. Therefore, every ultimately periodic state q in M is of the form $\rho(\mathcal{L}(q)) \subseteq vp^*$ with $v \overset{s}{\not\preceq} p^i$ for any i. Let M' be a copy of M. Then we basically apply Algorithm 1 to *every* ultimately periodic states of M' (cf. Theorem 4) such that every rule of an ultimately periodic state q is in the canonical form

$$q(f(x_1, \dots, x_m)) \to \gamma_0\, q_1(x_{\sigma(1)}) \dots q_n(x_{\sigma(n)})$$

where we discuss in the following how inverted letters are handled as Algorithm 1 processes LTWs. Let $q(f(x_1, \dots, x_m)) \to \gamma_0 q_1(x_{\sigma(1)})\gamma_1 \dots \gamma_{n-1} q_n(x_{\sigma(n)})\gamma_n$ be a rule of an ultimately periodic state q in M' with $\rho(\mathcal{L}(q)) \subseteq vp^*$ and $\rho(\mathcal{L}(q_i)) \subseteq v_i p_i^*$. Then $\rho(\gamma_0 v_1 \gamma_1 \dots \gamma_{n-1} v_n \gamma_n) \in \mathsf{A}^*$ as M' is suffix-empty. Thus, no inverted letters occur in the produced rules if the words are reduced. However, the parameter w for shifting the period can be instantiated with a word containing inverted letters (even after reduction) if in the algorithm `shiftPeriod(`$q_i, \gamma_i v_{i+1} \dots \gamma_{n-1} v_n \gamma_n w$ is recursively called as $\gamma_i v_{i+1} \dots \gamma_{n-1} v_n \gamma_n$ is not necessarily well-formed. As M' is suffix-empty, the languages $\rho(\mathcal{L}(q_1)), \dots, \rho(\mathcal{L}(q_n))$ are periodic, i.e., $\rho(\mathcal{L}(q_i)) \subseteq p_i^*$. Additionally, if $\rho(\mathcal{L}(q_1)) \subseteq v_1 p_1^*$ with $v_1 \neq \varepsilon$ then $\gamma_1 \mathcal{L}(q_2)\gamma_2 \dots \gamma_{n-1}\mathcal{L}(q_n)\gamma_n$ is well-formed. Thus, w.l.o.g. we assume that all $\rho(\mathcal{L}(q_i))$ are periodic and thus $\varepsilon \in \rho(\mathcal{L}(q_i))$. Let $\rho(\gamma_i v_{i+1} \dots \gamma_{n-1} v_n \gamma_n) = \overline{x}\, y$. Since the output of every rule is well-formed and M' is suffix-empty we have that

$$\gamma_0 v_1 \gamma_1 \dots \gamma_{i-1} \mathcal{L}(q_i) \overline{x}\, y \overset{\rho}{=} \gamma_0 v_1 \gamma_1 \dots \gamma_{i-1} \overline{x}\, x \mathcal{L}(q_i) \overline{x}\, y$$

Thus, $x\mathcal{L}(q_i)\overline{x}\, y \subseteq \mathsf{A}^*$ and $y^{-1} \cdot \rho(xp_i\overline{x}\,)y \in p^*$. As $(x^{-1} \cdot y)^{-1} \cdot p_i \cdot x^{-1} \cdot y = y^{-1} \cdot xp_i \cdot x^{-1} \cdot y = y^{-1} \cdot \rho(xp_i\overline{x}\,)y$ we can interpret the inverted letters as inverses in the free group over A and proceed with the algorithm. The described procedure runs in polynomial time and may increase the size of the $\mathsf{LTW_B}$ polynomially as discussed in Theorem 4. □

Example 23. *Consider the well-formed and suffix-empty* $\mathsf{LTW_B}$ *M with axiom $q(x_0)$ and rules*

$$\begin{array}{ll} \multicolumn{2}{c}{q(f(x_1,x_2)) \to ab\, q_1(x_2)\, \bar{b}\; q_2(x_1)\, \bar{a}\, ab} \\ q_1(g(x_1)) \to ab\, q_1(x_1)\, \bar{b}\, bab & q_1(h) \to \varepsilon \\ q_2(g(x_1)) \to q_2(x_1)\, ba & q_2(h) \to \varepsilon \end{array}$$

We omit a corresponding DTA. *We observe that $\rho(\mathcal{L}(q_1)) \subseteq (ab)^*$, $\rho(\mathcal{L}(q_2)) \subseteq (ba)^*$ and $\rho(\mathcal{L}(q)) \subseteq (ab)^*$. With Algorithm 1 we obtain for the top rule with left-hand side $q(f(x_1, x_2))$*

$$\begin{array}{ll} q^{ab}(f(x_1, x_)) & \to \rho(ab\bar{b}\,\bar{a}\, ab)\, q_1^{ab}(x_2)\, q_2^{ab}(x_1) \\ & \to ab\, q_1^{ab}(x_2)\, q_2^{ab}(x_1) \end{array}$$

Even if $\rho(\mathcal{L}(q_1)) \subseteq (ab)^*$ we bring the rules in the canonical form to eliminate all inverted letters on right-hand sides. The rules are constructed by a recursive call of function ***shiftPeriod*** *with parameter $w = b^{-1} \cdot a^{-1} \cdot ab = \varepsilon$ as inverted letters are interpreted as inverses in the free group over* A.

$$\begin{array}{ll} q_1^{ab}(g(x_1)) & \to \rho(ab\bar{b}\, bab)\, q_1^{ab}(x_1) \\ & \to abab\, q_1^{ab}(x_1) \\ q_1^{ab}(h) & \to \varepsilon \end{array}$$

To construct the rules for q_2^{ab} function ***shiftPeriod*** *with paramter $w = a^{-1} \cdot ab = b$.*

$$\begin{array}{ll} q_2^{ab}(g(x_1)) & \to b^{-1} \cdot bab\, q_2^{ab}(x_1) \\ & \to ab\, q_2^{ab}(x_1) \\ q_2^{ab}(h) & \to b^{-1} \cdot b \\ & \to \varepsilon \end{array}$$

In total we arrive at $\mathsf{LTW_B}$ *M' with axiom $q^{ab}(x_0)$ and rules*

$$\begin{array}{ll} \multicolumn{2}{c}{q^{ab}(f(x_1, x_2)) \to ab\, q_1^{ab}(x_2)\, q_2^{ab}(x_1)} \\ q_1^{ab}(g(x_1)) \to abab\, q_1^{ab}(x_1) & q_1^{ab}(h) \to \varepsilon \\ q_2^{ab}(g(x_1)) \to ab\, q_2^{ab}(x_1) & q_2^{ab}(h) \to \varepsilon \end{array}$$

with M' is equivalent to M.

Analogously to Theorem 4 we can order a well-formed and suffix-empty $\mathsf{LTW_B}$ M where for every ultimately period state q, $\rho(\mathcal{L}(q)) = \mathcal{L}(q)$ holds. Then, equivalent well-formed and ordered $\mathsf{LTW_B}$s are same-ordered and thus, equivalence can be decided in polynomial time by a reduction to the morphism equivalence problem over the free group (cf. Corollary 4).

4.4 Summary

In this chapter we showed that balancedness of 2-copy tree transducers is decidable in polynomial time. A 2-TW M calls in the axiom $(\gamma_0\, q_1(x_1)\, \gamma_1\, q_2(x_1)\, \gamma_2)$ two linear tree-to-word transducers on the input tree and performs therefore a *non-linear* transduction. We reduced the decision problem of balancedness of M to the question whether the output of q_1 and $\overline{q_2}$ is well-formed and the question whether q_1 and $\overline{q_2}$ are equivalent considering the output *after reduction*. Further, we showed that due to the previous check that q_1 and $\overline{q_2}$ are well-formed we can check q_1 and $\overline{q_2}$ for equivalence over the free group. Thus, Chapter 2 and Chapter 3 provide the results needed to decide balancedness of 2-TWs in polynomial time.

Additionally, we presented in Section 4.3 an approach to build the ordered form of linear well-formed tree-to-word transducers with output in the involutive monoid. The key observation thereby is that through the construction of the ordered form inverted letters are eliminated in (ultimately) periodic states. Since this approach can analogously be applied on context-free grammars this is one step in answering the question whether for a well-formed context-free grammar an equivalent grammar without inverses can be constructed (in polynomial time).

5 Conclusion

In this thesis we studied three different types of linear tree transducers with serialized output, (i) LTWs with output interpreted as words in the free monoid, (ii) $\mathsf{LTW_B}$s with output interpreted over the involutive monoid, and (iii) LTs with output interpreted in the free group. For $\mathsf{LTW_B}$s we considered the special case that the output is always well-formed, thus reduces to a word over the opening letters only. Chapter 3 and Section 4.3 contain the following core contributions:

- We showed that equivalence of linear tree transducers with output in the free group is decidable in polynomial time. This subsumes our previous result on equivalence of linear tree-to-word transducers. Additionally, we showed that equivalence of well-formed $\mathsf{LTW_B}$s can be decided in the free group by interpreting inverted letters as inverses.

- We introduced an ordered form for LTs that simplifies the partial normal form from [BP16] and showed that equivalent LTs in ordered form process their input in the same order, i.e., they are same-ordered.

- We gave a polynomial-time reduction from equivalence of same-ordered LTs to the morphism equivalence problem of context-free grammars over the free group by simulating all parallel runs of two LTs with a context-free grammar.

- We showed that the ordered form can be obtained for LTWs (cf. [BP16]) as well as for well-formed $\mathsf{LTW_B}$s. The technical challenge thereby is that no inverses as in the free group can be used which leads to more involved constructions to build the ordered form and a polynomial size increase of the obtained transducer. Additionally, in case of well-formed $\mathsf{LTW_B}$s we showed that the construction removes inverted letters from ultimately periodic states.

While equivalence of deterministic tree-to-word transducer was shown to be decidable [SMK15], the decision procedure given in [SMK15] is not based on a normal form. Therefore, the question arises whether a normal form for a larger class of tree transducers with serialized output can be given. Here, we observed that the use of inverses provided by the free group lead to simpler proofs. Therefore, it might be beneficial to consider the output in the free group also for non linear tree transducers to obtain a normal form.

In Chapter 2 we showed that well-formedness of context-free grammars is decidable in polynomial time. Therefore, we reduced the decision problem to the computation of the *reduced* longest common suffix of a context-free grammar. Chapter 2 contains the following core contributions:

- We gave basic characteristics of well-formed (context-free) languages and provided comprehensive computation rules for the longest common suffix.
- We defined a polynomial size representation of a language, the lcs-summary, to compute the longest common suffix of a language. The lcs-summary consists of the longest common suffix of the language and the maximal suffix extension which is the longest word to which the suffix can be extended if another language is concatenated from the left.
- We showed that the reduced longest common suffix of a context-free grammar G is the same as the reduced longest common suffix of the words with a derivation tree up to height $4N$, where N is the number of nonterminals in G. Therefore, we can compute an SLP representing the reduced longest common suffix of a context-free grammar in polynomial time.
- We provided easier and shorter proofs for the computation of the longest common prefix (suffix) of a context-free grammar where no reduction is taken into account [LPS18]. Further, we extended these results by analysing the reduced longest common suffix of linear well-formed languages of the form $\{\alpha\sigma_1 \ldots \sigma_k\gamma\tau_k \ldots \tau_1\beta \mid \alpha, \beta, \gamma, \sigma_i, \tau_i \in \mathsf{B}^*\}$ with B the set of opening and corresponding closing letters.

Unfortunately we could not directly use the results from [LPS18] on the computation of the longest common prefix (suffix) of a context-free grammar to compute the *reduced* longest common suffix of a context-free grammar. The reason for this is that we were not able to rewrite a context-free grammar G over B^* to an equivalent context-free grammar G' such that $\rho(L(G)) = L(G')$. Moreover, the question is whether G' would be of size polynomial in G. The proofs given in Section 2.4 to compute the reduced longest common suffix of simple well-formed languages indicate that it might only be possible to find a reduced equivalent grammar of exponential size.

Finally, we showed in Chapter 4 that balancedness of 2-copy tree transducers can be decided in polynomial time. The decision procedure combines the two results about well-formedness of context-free grammars and equivalence of linear tree transducers over the free group. It is a natural question whether the decision algorithms for balancedness of 2-TWs can be extended to tree transducer models where more than two linear tree-to-word transducers are called in the axiom. However, in case of two transducers it is clear that the opening letters remaining after reduction and produced by the first transducers have to be closed by the second transducer. Considering three transducers this property does not hold anymore and more nested dependencies between the different transducers arise. Therefore, the extension of balancedness to other similar transducer models remains as an open problem.

Acknowledgements

Vielen Dank, Helmut, für die Möglichkeit, diese Arbeit zu verfassen, die Einführung in dieses Thema und alle Unterstützung!

Vielen Dank, Michael L., für die vielen hilfreichen und unterstützenden thematischen Diskussionen sowie alle Gespräche über Gott und die Welt!

Vielen Dank, allen aktuellen und vorherigen Kollegen und besonders Stefan, für ein wunderbares Arbeitsumfeld und manches Korrekturlesen!

Vielen Dank, allen Freunden, meiner Familie und insbesondere meinen Eltern, für die Unterstützung und Hilfe vor allem am Ende dieser Arbeit, die mir die Fertigstellung (zu 99,9%) vor dem nächsten Großprojekt ermöglicht haben!

Vielen Dank, Rico, für *alles* – vor allem das gemeinsame Bewältigen und Überleben der letzten Monate und Wochen!

July 2020

Bibliography

[BB02] J. Berstel and L. Boasson. Formal properties of XML grammars and languages. *Acta Inf.*, 38(9):649–671, 2002.

[Boi16] Adrien Boiret. Normal form on linear tree-to-word transducers. In Adrian-Horia Dediu, Jan Janoušek, Carlos Martín-Vide, and Bianca Truthe, editors, *Language and Automata Theory and Applications: 10th International Conference, LATA 2016*, pages 439–451. LNCS 9618, Springer, 2016.

[BP16] 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, Proceedings*, volume 9840 of *Lecture Notes in Computer Science*, pages 355–367. Springer, 2016.

[BSQM13] Fabienne Braune, Nina Seemann, Daniel Quernheim, and Andreas Maletti. Shallow local multi-bottom-up tree transducers in statistical machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 1: Long Papers*, pages 811–821. The Association for Computer Linguistics, 2013.

[Cho56] Noam Chomsky. Three models for the description of language. *IRE Trans. Inf. Theory*, 2(3):113–124, 1956.

[Cho03] Christian Choffrut. Minimizing subsequential transducers: a survey. *Theor. Comput. Sci.*, 292(1):131–143, 2003.

[EM99] Joost Engelfriet and Sebastian Maneth. Macro tree transducers, attribute grammars, and MSO definable tree translations. *Inf. Comput.*, 154(1):34–91, 1999.

[EM03] Joost Engelfriet and Sebastian Maneth. Macro tree translations of linear size increase are MSO definable. *SIAM J. Comput.*, 32(4):950–1006, 2003.

[EM06] Joost Engelfriet and Sebastian Maneth. The equivalence problem for deterministic MSO tree transducers is decidable. *Information Processing Letters*, 100(5):206–212, 2006.

[EMS09] Joost Engelfriet, Sebastian Maneth, and Helmut Seidl. Deciding equivalence of top-down XML transformations in polynomial time. *J. Comput. Syst. Sci.*, 75(5):271–286, 2009.

[Eng80] Joost Engelfriet. *Some open questions and recent results on tree transducers and tree languages*, pages 241–286. Academic Press, 12 1980.

[Ési80] Zoltán Ésik. Decidability results concerning tree transducers I. *Acta Cybern.*, 5(1):1–20, 1980.

[Flo62] Robert W. Floyd. On ambiguity in phrase structure languages. *Commun. ACM*, 5(10):526, 1962.

[Fri11] Sylvia Friese. *On normalization and type checking for tree transducers.* PhD thesis, Technical University Munich, 2011.

[FSM10] Sylvia Friese, Helmut Seidl, and Sebastian Maneth. Minimization of deterministic bottom-up tree transducers. In Yuan Gao, Hanlin Lu, Shinnosuke Seki, and Sheng Yu, editors, *Developments in Language Theory, 14th International Conference, DLT 2010, London, ON, Canada, August 17-20, 2010, Proceedings*, volume 6224 of *Lecture Notes in Computer Science*, pages 185–196. Springer, 2010.

[FV98] Zoltán Fülöp and Heiko Vogler. *Syntax-Directed Semantics - Formal Models Based on Tree Transducers.* Monographs in Theoretical Computer Science. An EATCS Series. Springer, 1998.

[Gri68] Timothy V. Griffiths. The unsolvability of the equivalence problem for lambda-free nondeterministic generalized machines. *J. ACM*, 15(3):409–413, 1968.

[HMNI14] Shizuya Hakuta, Sebastian Maneth, Keisuke Nakano, and Hideya Iwasaki. Xquery streaming by forest transducers. In Isabel F. Cruz, Elena Ferrari, Yufei Tao, Elisa Bertino, and Goce Trajcevski, editors, *IEEE 30th International Conference on Data Engineering, Chicago, ICDE 2014, IL, USA, March 31 - April 4, 2014*, pages 952–963. IEEE Computer Society, 2014.

[HU69] John E. Hopcroft and Jeffrey D. Ullman. *Formal languages and their relation to automata.* Addison-Wesley series in computer science and information processing. Addison-Wesley, 1969.

[HU79] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley, 1979.

[Knu67] D. E. Knuth. A characterization of parenthesis languages. *Information and Control*, 11(3):269–289, 1967.

[Kob19] Naoki Kobayashi. Inclusion between the frontier language of a non-deterministic recursive program scheme and the dyck language is undecidable. *Theor. Comput. Sci.*, 777:409–416, 2019.

[KPR92] Juhani Karhumäki, Wojciech Plandowski, and Wojciech Rytter. Polynomial size test sets for context-free languages. In Werner Kuich, editor, *Automata, Languages and Programming, 19th International Colloquium, ICALP92, Vienna, Austria, July 13-17, 1992, Proceedings*, volume 623 of *Lecture Notes in Computer Science*, pages 53–64. Springer, 1992.

[KRS95] Marek Karpinski, Wojciech Rytter, and Ayumi Shinohara. Pattern-matching for strings with short descriptions. In Zvi Galil and Esko Ukkonen, editors, *Combinatorial Pattern Matching, 6th Annual Symposium, CPM 95, Espoo, Finland, July 5-7, 1995, Proceedings*, volume 937 of *Lecture Notes in Computer Science*, pages 205–214. Springer, 1995.

[LL06] Yury Lifshits and Markus Lohrey. Querying and embedding compressed texts. In Rastislav Kralovic and Pawel Urzyczyn, editors, *Mathematical Foundations of Computer Science 2006, 31st International Symposium, MFCS 2006, Stará Lesná, Slovakia, August 28-September 1, 2006, Proceedings*, volume 4162 of *Lecture Notes in Computer Science*, pages 681–692. Springer, 2006.

[LLN+11] Grégoire Laurence, Aurélien Lemay, Joachim Niehren, Sławek 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*, pages 354–365. LNCS 6638, Springer, 2011.

[LLN+14] Grégoire Laurence, Aurélien Lemay, Joachim Niehren, Sławek 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*, pages 490–502. LNCS 8370, Springer, 2014.

[LLS19] Raphaela Löbel, Michael Luttenberger, and Helmut Seidl. On the balancedness of tree-to-word transducers, 2019.

[LLS20a] Raphaela Löbel, Michael Luttenberger, and Helmut Seidl. Equivalence of linear tree transducers with output in the free group. In Natasa Jonoska and Dmytro Savchuk, editors, *Developments in Language Theory - 24th International Conference, DLT 2020, Proceedings*, volume 12086 of *Lecture Notes in Computer Science*, pages 207–221. Springer, 2020.

[LLS20b] Raphaela Löbel, Michael Luttenberger, and Helmut Seidl. On the balancedness of tree-to-word transducers. In Natasa Jonoska and Dmytro Savchuk, editors, *Developments in Language Theory - 24th International Conference, DLT 2020, Proceedings*, volume 12086 of *Lecture Notes in Computer Science*, pages 222–236. Springer, 2020.

[Loh04] Markus Lohrey. Word problems on compressed words. In Josep Díaz, Juhani Karhumäki, Arto Lepistö, and Donald Sannella, editors, *Automata, Languages and Programming: 31st International Colloquium, ICALP 2004, Turku, Finland, July 12-16, 2004. Proceedings*, volume 3142 of *Lecture Notes in Computer Science*, pages 906–918. Springer, 2004.

[Loh06] Markus Lohrey. Word problems and membership problems on compressed words. *SIAM J. Comput.*, 35(5):1210–1240, 2006.

[Loh12] Markus Lohrey. Algorithmics on SLP-compressed strings: A survey. *Groups Complexity Cryptology*, 4(2), 2012.

[Loh14] Markus Lohrey. *The Compressed Word Problem for Groups.* Springer Briefs in Mathematics. Springer, 2014.

[LPS18] Michael Luttenberger, Raphaela Palenta, and Helmut Seidl. Computing the longest common prefix of a context-free language in polynomial time. In Rolf Niedermeier and Brigitte Vallée, editors, *35th Symposium on Theoretical Aspects of Computer Science, STACS 2018*, volume 96 of *LIPIcs*, pages 48:1–48:13. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.

[LS15] Roger C. Lyndon and Paul E. Schupp. *Combinatorial Group Theory.* Classics in Mathematics. Springer, 2015.

[Mal17] Andreas Maletti. Survey: Finite-state technology in natural language processing. *Theor. Comput. Sci.*, 679:2–17, 2017.

[MBPS05] Sebastian Maneth, Alexandru Berlea, Thomas Perst, and Helmut Seidl. XML type checking with macro tree transducers. In Chen Li, editor, *Proceedings of the Twenty-fourth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 13-15, 2005, Baltimore, Maryland, USA*, pages 283–294. ACM, 2005.

[MGHK09] Andreas Maletti, Jonathan Graehl, Mark Hopkins, and Kevin Knight. The power of extended top-down tree transducers. *SIAM J. Comput.*, 39(2):410–430, 2009.

[MN99] Sebastian Maneth and Frank Neven. Structured document transformations based on XSL. In Richard C. H. Connor and Alberto O. Mendelzon, editors, *Research Issues in Structured and Semistructured Database Programming, 7th International Workshop on Database Programming Languages, DBPL'99, Kinloch Rannoch, Scotland, UK, September 1-3, 1999, Revised Papers*, volume 1949 of *Lecture Notes in Computer Science*, pages 80–98. Springer, 1999.

[Moh00] Mehryar Mohri. Minimization algorithms for sequential transducers. *Theor. Comput. Sci.*, 234(1-2):177–201, 2000.

[MS18] Sebastian Maneth and Helmut Seidl. Balancedness of MSO transductions in polynomial time. *Inf. Process. Lett.*, 133:26–32, 2018.

[MT06] Y. Minamide and A. Tozawa. XML validation for context-free grammars. In *APLAS*, pages 357–373. LNCS 4279, Springer, 2006.

[Mül06] Markus Müller-Olm. *Variations on Constants - Flow Analysis of Sequential and Parallel Programs*, volume 3800 of *Lecture Notes in Computer Science*. Springer, 2006.

[Pla94] Wojciech Plandowski. Testing equivalence of morphisms on context-free languages. In *Algorithms - ESA '94, Second Annual European Symposium*, pages 460–470. LNCS 855, Springer, 1994.

[PR99] Wojciech Plandowski and Wojciech Rytter. Complexity of language recognition problems for compressed words. In Juhani Karhumäki, Hermann A. Maurer, Gheorghe Paun, and Grzegorz Rozenberg, editors, *Jewels are Forever, Contributions on Theoretical Computer Science in Honor of Arto Salomaa*, pages 262–272. Springer, 1999.

[Ryt03] Wojciech Rytter. Application of lempel-ziv factorization to the approximation of grammar-based compression. *Theor. Comput. Sci.*, 302(1-3):211–222, 2003.

[Sak] Jacques Sakarovitch. *Elements of Automata Theory.* Cambridge University Press.

[SLLN09] Slawomir Staworko, Grégoire Laurence, Aurélien Lemay, and Joachim Niehren. Equivalence of deterministic nested word to word transducers. In Miroslaw Kutylowski, Witold Charatonik, and Maciej Gebala, editors, *Fundamentals of Computation Theory, 17th International Symposium, FCT 2009, Wroclaw, Poland, September 2-4, 2009. Proceedings*, volume 5699 of *Lecture Notes in Computer Science*, pages 310–322. Springer, 2009.

[SMK15] Helmut Seidl, Sebastian Maneth, and Gregor Kemper. Equivalence of deterministic top-down tree-to-string transducers is decidable. In Venkatesan Guruswami, editor, *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 943–962. IEEE Computer Society, 2015.

[SMK18] Helmut Seidl, Sebastian Maneth, and Gregor Kemper. Equivalence of deterministic top-down tree-to-string transducers is decidable. *J. ACM*, 65(4):21:1–21:30, 2018.

[TM07] Akihiko Tozawa and Yasuhiko Minamide. Complexity results on balanced context-free languages. In Helmut Seidl, editor, *Foundations of Software Science and Computational Structures, 10th International Conference, FOSSACS 2007, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2007, Proceedings*, volume 4423 of *Lecture Notes in Computer Science*, pages 346–360. Springer, 2007.

[Voi05] Janis Voigtländer. *Tree transducer composition as program transformation.* PhD thesis, Dresden University of Technology, 2005.

[Wad90] Philip Wadler. Deforestation: Transforming programs to eliminate trees. *Theor. Comput. Sci.*, 73(2):231–248, 1990.

[Zac79] Z. Zachar. The solvability of the equivalence problem for deterministic frontier-to-root tree transducers. *Acta Cybern.*, 4(2):167–177, 1979.