Theory of Computing Systems manuscript No.
(will be inserted by the editor)

Grammar-based Compression of Unranked Trees

Adrià Gascón · Markus Lohrey · Sebastian Maneth · Carl Philipp Reh · Kurt Sieber

Received: date / Accepted: date

Abstract We introduce forest straight-line programs (FSLPs for short) as a compressed representation of unranked ordered node-labelled trees. FSLPs are based on the operations of forest algebra and generalize tree straight-line programs. We compare the succinctness of FSLPs with two other compression schemes for unranked trees: top dags and tree straight-line programs of first-child/next sibling encodings. Efficient translations between these formalisms are provided. Finally, we show that equality of unranked trees in the setting where certain symbols are associative and/or commutative can be tested in polynomial time. This generalizes previous results for testing isomorphism of compressed unordered ranked trees. An extended abstract of this paper appeared in [14].

Keywords grammar-based tree compression · top dags · equality testing · forest

CR Subject Classification E.4 Data compaction and compression

Adrià Gascón
Warwick University and Alan Turing Institute, UK
E-mail: agascon@inf.ed.ac.uk

Markus Lohrey
Universität Siegen, Germany
E-mail: lohrey@eti.uni-siegen.de

Sebastian Maneth
Universität Bremen, Germany
E-mail: sebastian.maneth@gmail.com

Carl Philipp Reh
Universität Siegen, Germany
Tel.: +49-271-740-3233
E-mail: reh@eti.uni-siegen.de

Kurt Sieber
Universität Siegen, Germany
E-mail: sieber@informatik.uni-siegen.de

Mathematics Subject Classification (2000) 68Q42; 68P30

1 Introduction

Grammar-based compression represents an object succinctly by means of a small context-free grammar. In many grammar-based compression formalisms such a grammar can be exponentially smaller than the object. Several string compressors use this idea and represent an input string w by a context-free grammar that produces only w [8]; such grammars are also known as straight-line programs—in this paper we use the term string straight-line program, SSLP for short. One of the biggest advantages of grammar-based compressors is that many algorithmic problems allow for efficient algorithms that work directly on the grammar and do not need prior decompression, see the survey [20] for references. One of the most well-known and fundamental such problems is testing equality of the strings produced by two string straight-line programs. Polynomial time solutions to this problem were discovered, in different contexts by different groups of people [15,17,27,28].

Grammar-based compression has been generalized from strings to ordered ranked node-labelled trees, by means of linear context-free tree grammars generating exactly one tree [7]. Such grammars are also known as tree straight-line programs, TSLPs for short, see [21] for a survey. Equality of the trees produced by two TSLPs can also be checked in polynomial time: one constructs SSLPs for the pre-order traversals of the trees, and then applies the above mentioned result for SSLPs, see [7]. The tree case becomes more complex when *unordered* ranked trees are considered. Such trees can be represented using TSLPs, by simply ignoring the order of children in the produced tree. Checking isomorphism of unordered ranked trees generated by TSLPs was recently shown to be solvable in polynomial time [23]. The solution transforms the TSLPs so that they generate canonical representations of the original trees and then checks equality of these canonical forms.

The aforementioned result for ranked trees cannot be applied to *unranked* trees (where the number of children of a node is not bounded), which arise for instance in XML document trees. This is unfortunate, because (i) grammar-based compression is particularly effective for XML document trees (see [22]), and (ii) XML document trees can often be considered unordered (one speaks of "data-centric XML", see e.g. [1,4,6,29,30]), allowing even stronger grammar-based compressions [24].

In this paper we introduce a generalization of TSLPs and SSLPs that allows to produce ordered unranked node-labelled trees and forests (i.e., ordered sequences of trees) that we call *forest straight-line programs*, FSLPs for short. In contrast to TSLPs, FSLPs can compress very wide and flat trees. For instance, the tree $f(a,\ldots,a)$ with n many a's is not compressible with TSLPs but can be produced by an FSLP of size $\Theta(\log n)$. FSLPs are based on the operations of horizontal and vertical forest composition from forest algebras [5].

In the following Sections 1.1 and 1.2 we explain the the main contributions of this paper.

1.1 Comparison with other formalisms

We compare the succinctness of FSLPs with two other grammar-based formalisms for compressing unranked node-labelled ordered trees: TSLPs for "first-child/next-sibling" (fcns) encodings and top dags. The fcns-encoding is the standard way of transforming an unranked tree into a binary tree. Then the resulting binary tree can be succinctly represented by a TSLP. This approach was used to apply the TreeRePair-compressor from [22] to unranked trees. We prove that FSLPs and TSLPs for fcns-encodings are equally succinct up to constant multiplicative factors and that one can change between both representations in linear time (Propositions 3 and 4).

Top dags are another formalism for compressing unranked trees [3]. Top dags use horizontal and vertical merge operations for tree construction, which are very similar to the horizontal and vertical concatenation operations from FSLPs. Whereas a top dag can be transformed in linear time into an equivalent FSLP with a constant multiplicative blow-up (Proposition 1), the reverse transformation (from an FSLP to a top dag) needs time $\mathcal{O}(\sigma \cdot n)$ and involves a multiplicative blow-up of size $\mathcal{O}(\sigma)$ where σ is the number of node labels of the tree (Proposition 2). A simple example (Example 6) shows that this σ-factor is unavoidable. The reason for the σ-factor is a technical restriction in the definition of top dags: In contrast to FSLPs, top dags only allow sharing of common subtrees but not of common subforests. Hence, sharing between (large) subtrees which only differ in their root labels may be impossible at all (as illustrated by Example 6), and this leads to the blow-up by a factor of σ in comparison to FSLPs. The impossibility of sharing subforests would also complicate the technical details of our main algorithmic results for FSLPs (in particular Proposition 4 and Theorem 3 which is discussed below) for which we make heavy use of a particular normal form for FSLPs that exploits the sharing of proper subforests. We therefore believe that at least for our purposes, FSLPs are a more adequate formalism than top dags.

1.2 Testing equality modulo associativity and commutativity

Our main algorithmic result for FSLPs can be formulated as follows: Fix a set Σ of node labels and take a subset $\mathcal{C} \subseteq \Sigma$ of "commutative" node labels and a subset $\mathcal{A} \subseteq \Sigma$ of "associative" node labels. This means that for all $a \in \mathcal{A}$, $c \in \mathcal{C}$ and all trees $t_1, t_2, \ldots, t_n$ we do not distinguish

(i) between $c(t_1, \ldots, t_n)$ and $c(t_{\xi(1)}, \ldots, t_{\xi(n)})$ for any permutation ξ (commutativity), and

(ii) between $a(t_1, \ldots, t_n)$ and $a(t_1, \ldots, t_{i-1}, a(t_i, \ldots, t_{j-1}), t_j, \ldots, t_n)$ for any indices i, j with $1 \leq i \leq j \leq n+1$ (associativity).

We then show that for two given FSLPs F_1 and F_2 that produce trees t_1 and t_2 (of possible exponential size), one can check in polynomial time whether t_1 and t_2 are equal modulo commutativity and associativity (Theorem 3). Note that unordered tree isomorphism corresponds to the case $\mathcal{C} = \Sigma$ and $\mathcal{A} = \emptyset$ (in particular we generalize the result from [23] for ranked unordered trees). Theorem 3 also holds if the trees t_1 and t_2 are given by top dags or TSLPs for the fcns-encodings, since these formalisms can be transformed efficiently into FSLPs. Theorem 3 also shows the usefulness of FSLPs even if one is only interested in say binary trees, which are represented by TSLPs. The law of associativity will yield very wide and flat trees that are no longer compressible with TSLPs but are still compressible with FSLPs.

1.3 Related work

In the recent paper [12] a general balancing result is shown, which also applies to FSLPs: From a given FSLP of size n that produces a forest of size N one can construct in linear time an equivalent FSLP of size $\mathcal{O}(n)$ and height $\mathcal{O}(\log N)$ [12, Corollary 11.5]. A corresponding result for top dags is shown as well [12, Corollary 12.3].

2 Straight-line programs over algebras

We will produce strings, trees and forests by algebraic expressions over certain algebras. These expressions will be compressed by directed acyclic graphs. In this section, we introduce the general framework, which will be reused several times in this paper.

An algebraic structure is a tuple $\mathcal{A} = (A, f_1, \ldots, f_k)$ where A is the universe and every $f_i \colon A^{n_i} \to A$ is an operation of a certain arity n_i. In this paper, the arity of all operations will be at most two. If $n_i = 0$, then f_i is called a constant. Moreover, it will be convenient to allow partial operations for the f_i. We will make use of such partial functions but we will not construct expressions that evaluate to undefined. Algebraic expressions over $\mathcal{A}$ are defined in the usual way: if $e_1, \ldots, e_{n_i}$ are algebraic expressions over $\mathcal{A}$, then also $f_i(e_1, \ldots, e_{n_i})$ is an algebraic expressions over $\mathcal{A}$. For an algebraic expression e, $[\![e]\!] \in A$ denotes the element to which e evaluates (it can be undefined).

A *straight-line program* (SLP for short) over $\mathcal{A}$ is a tuple $P = (V, S, \rho)$, where V is a set of *variables*, $S \in V$ is the *start variable*, and ρ maps every variable $A \in V$ to an expression of the form $f_i(A_1, \ldots, A_{n_i})$ (the so called *right-hand side* of A) such that $A_1, \ldots, A_{n_i} \in V$ and the edge relation $E(P) = \{(B, A) \in V \times V \mid B \text{ occurs in } \rho(A)\}$ is acyclic. This allows to define for every variable $A \in V$ its value $[\![A]\!]_P$ inductively by $[\![A]\!]_P = f_i([\![A_1]\!]_P, \ldots, [\![A_{n_i}]\!]_P)$ if $\rho(A) = f_i(A_1, \ldots, A_{n_i})$. Since the f_i can be partially defined, the value of a variable can be undefined. The SLP P will be called *valid* if all values $[\![A]\!]_P$ $(A \in V)$ are defined. We will only construct SLPs that are valid. Moreover, for each of our algebras, it will be easy to check whether an SLP is valid. The value of P is $[\![P]\!] = [\![S]\!]_P$. Usually, we prove properties of an SLP P by induction on the finite partial order $\leq_P = E(P)^*$.

It will be convenient to allow for the right-hand sides $\rho(A)$ algebraic expressions over $\mathcal{A}$, where the variables from V can appear as atomic expressions. By introducing additional variables, we can transform such an SLP into an equivalent SLP of the original form. We define the size $|P|$ of an SLP P as the total number of occurrences of operations $f_1, \ldots, f_k$ in all right-hand sides (which is the number of variables if all right-hand sides have the standard form $f_i(A_1, \ldots, A_{n_i})$).

Sometimes it is useful to view an SLP $P = (V, S, \rho)$ as a directed acyclic graph (dag) $(V, E(P))$, together with the distinguished output node S, and the node labelling that associates the label f_i with the node $A \in V$ if $\rho(A) = f_i(A_1, \ldots, A_{n_i})$. Note that the outgoing edges $(A, A_1), \ldots, (A, A_{n_i})$ have to be ordered since f_i is in general not commutative and that multi-edges have to be allowed. Such dags are also known as algebraic circuits in the literature (see [2] for the special case of algebraic circuits over fields and [19] for arbitrary algebraic structures).

2.1 String straight-line programs

A widely studied type of SLPs are SLPs over a free monoid $(\Sigma^*, \cdot, \varepsilon, (a)_{a\in\Sigma})$, where $\cdot$ is the concatenation operator (which, as usual, is not written explicitly in expressions) and the empty string ε and every alphabet symbol $a \in \Sigma$ are added as constants. We use the term *string straight-line programs* (SSLPs for short) for these SLPs. If we want to emphasize the alphabet Σ, we speak of an SSLP over Σ. In many papers, SSLPs are just called straight-line programs; see [20] for a survey. Occasionally we consider SSLPs without a start variable S and then write (V, ρ).

Example 1 Consider the SSLP $G = (\{S, A, B, C\}, S, \rho)$ over the alphabet $\{a, b\}$ with $\rho(S) = AAB$, $\rho(A) = CBB$, $\rho(B) = CaC$, $\rho(C) = b$. We have $[\![B]\!]_G = bab$, $[\![A]\!]_G = bbabbab$, and $[\![G]\!] = bbabbabbbabbabbab$. The size of G is 8 (six concatenation operators are used in the right-hand sides, and there are two occurrences of constants).

In the next two sections, we introduce two types of algebras for trees and forests.

3 Forest algebras and forest straight-line programs

3.1 Trees and forests

Let us fix a finite set Σ of node labels for the rest of the paper. We consider Σ-labelled rooted ordered trees, where "ordered" means that the children of a node are totally ordered. Every node has a label from Σ. Note that we make no rank assumption: the number of children of a node (also called its degree) is not determined by its node label. The set of nodes (resp. edges) of t is denoted by $V(t)$ (resp., $E(t)$). A *forest* is a (possibly empty) sequence of trees. The size $|f|$ of a forest is the total number of nodes in f. The set of all Σ-labelled forests is denoted by $\mathcal{F}_0(\Sigma)$ and the set of all Σ-labelled trees is denoted by $\mathcal{T}_0(\Sigma)$. As usual, we can identify trees with expressions built up from symbols in Σ and parentheses. Formally, $\mathcal{F}_0(\Sigma)$ and $\mathcal{T}_0(\Sigma)$ can be inductively defined as the following sets of strings over the alphabet $\Sigma \cup \{(,)\}$.

- If $t_1, \ldots, t_n$ are Σ-labelled trees with $n \geq 0$, then the string $t_1 t_2 \cdots t_n$ is a Σ-labelled forest (in particular, the empty string ε is a Σ-labelled forest).
- If f is a Σ-labelled forest and $a \in \Sigma$, then $a(f)$ is a Σ-labelled tree (where the singleton tree $a()$ is usually written as a).

Let us fix a distinguished symbol $x \notin \Sigma$ for the rest of the paper (called the parameter). The set of forests $f \in \mathcal{F}_0(\Sigma \cup \{x\})$ such that x has a unique occurrence in f and this occurrence is at a leaf node is denoted by $\mathcal{F}_1(\Sigma)$. Similarly, the set of trees $t \in \mathcal{T}_0(\Sigma \cup \{x\})$ such that x has a unique occurrence in t and this occurrence is at a leaf node is denoted by $\mathcal{T}_1(\Sigma)$. Elements of $\mathcal{T}_1(\Sigma)$ (resp., $\mathcal{F}_1(\Sigma)$) are called tree contexts (resp., forest contexts). We finally define $\mathcal{F}(\Sigma) = \mathcal{F}_0(\Sigma) \cup \mathcal{F}_1(\Sigma)$ and $\mathcal{T}(\Sigma) = \mathcal{T}_0(\Sigma) \cup \mathcal{T}_1(\Sigma)$. Following [5], we define the *forest algebra* $\mathsf{FA}(\Sigma) = (\mathcal{F}(\Sigma), \boxdot, \boxbar, (a)_{a \in \Sigma}, \varepsilon, x)$ as follows:

- $\boxdot$ is the horizontal concatenation operator: for forests $f_1, f_2 \in \mathcal{F}(\Sigma)$, $f_1 \boxdot f_2$ is defined if $f_1 \in \mathcal{F}_0(\Sigma)$ or $f_2 \in \mathcal{F}_0(\Sigma)$ and in this case we set $f_1 \boxdot f_2 = f_1 f_2$ (i.e., we concatenate the corresponding sequences of trees).
- $\boxbar$ is the vertical concatenation operator: for forests $f_1, f_2 \in \mathcal{F}(\Sigma)$, $f_1 \boxbar f_2$ is defined if $f_1 \in \mathcal{F}_1(\Sigma)$ and in this case $f_1 \boxbar f_2$ is obtained by replacing in f_1 the unique occurrence of the parameter x by the forest f_2.
- Every $a \in \Sigma$ is identified with the unary function $a \colon \mathcal{F}(\Sigma) \to \mathcal{T}(\Sigma)$ that produces $a(f)$ when applied to $f \in \mathcal{F}(\Sigma)$.
- $\varepsilon \in \mathcal{F}_0(\Sigma)$ and $x \in \mathcal{F}_1(\Sigma)$ are constants of the forest algebra.

For better readability, we also write $f\langle g \rangle$ instead of $f \boxbar g$, fg instead of $f \boxdot g$, and a instead of $a(\varepsilon)$.

As an example for the vertical concatenation, consider the forests $f = aa(bxb)$ and $g = c(dd)c(d)$, where we have $f \boxbar g = f\langle g \rangle = aa(bc(dd)c(d)b)$.

3.2 First-child/next-sibling encoding

The first-child/next-sibling encoding transforms a forest over some alphabet Σ into a binary tree over $\Sigma \uplus \{\bot\}$, where every node labelled with $a \in \Sigma$ has degree 2 and every node labelled with $\bot$ has degree 0. We define $\mathrm{fcns} \colon \mathcal{F}_0(\Sigma) \to \mathcal{T}_0(\Sigma \uplus \{\bot\})$ inductively by:

- $\mathrm{fcns}(\varepsilon) = \bot$,
- $\mathrm{fcns}(a(f)g) = a(\mathrm{fcns}(f)\, \mathrm{fcns}(g))$ for $f, g \in \mathcal{F}_0(\Sigma)$, $a \in \Sigma$.

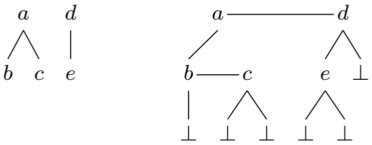

Fig. 1 Forest f on the left and fcns(f) on the right from Example 2.

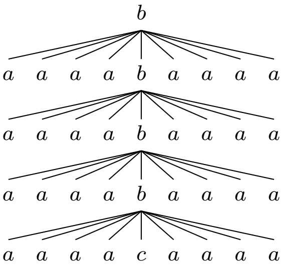

Fig. 2 Forest $[\![F]\!]$ for $n = 2$ from Example 3.

Thus, the left (resp., right) child of a node in fcns(f) is the first child (resp., right sibling) of the node in f or a $\perp$-labelled leaf if it does not exist.

Example 2 If $f = a(bc)d(e)$ then

$$\begin{aligned}\mathrm{fcns}(f) &= \mathrm{fcns}(a(bc)d(e)) = a(\mathrm{fcns}(bc)\,\mathrm{fcns}(d(e)))\\ &= a(b(\perp \mathrm{fcns}(c))d(\mathrm{fcns}(e)\perp)) = a(b(\perp c(\perp\perp))d(e(\perp\perp)\perp)),\end{aligned}$$

see also Figure 1.

3.3 Forest straight-line programs

A *forest straight-line program* over Σ, FSLP for short, is a valid straight-line program over the algebra $\mathsf{FA}(\Sigma)$ such that $[\![F]\!] \in \mathcal{F}_0(\Sigma)$. Iterated vertical and horizontal concatenations allow to generate forests, whose depth and width is exponential in the FSLP size. For an FSLP $F = (V, S, \rho)$ and $i \in \{0, 1\}$ we define $V_i = \{A \in V \mid [\![A]\!]_F \in \mathcal{F}_i(\Sigma)\}$.

Example 3 Let $n \in \mathbb{N}$ and let $F = (S, V, \rho)$ be the FSLP over $\{a, b, c\}$ with $V_0 = \{S, A_0, \ldots, A_n\}$, $V_1 = \{B_0, \ldots, B_n\}$ where ρ is defined by $\rho(A_0) = a$, $\rho(B_0) = b(A_n x A_n)$, $\rho(A_i) = A_{i-1}A_{i-1}$ and $\rho(B_i) = B_{i-1}\langle B_{i-1}\rangle$ for $1 \leq i \leq n$, and $\rho(S) = B_n\langle c\rangle$. Then we have $[\![F]\!] = b(a^{2^n} b(a^{2^n} \cdots b(a^{2^n} c\, a^{2^n}) \cdots a^{2^n})a^{2^n})$, where b occurs 2^n many times. This is a forest whose width and depth is exponential in the size of the FSLP F. See Figure 2 for $n = 2$.

Example 4 Consider the alphabet $\Sigma = \{a, b, c, d, e\}$. Let $n \geq 0$ be a natural number, and let $F = (V, S_1, \rho)$ be the FSLP with

$$\begin{aligned}
V_0 &= \{A_1, A_2, B, S_1\},\\
V_1 &= \{B_0, \ldots, B_n, C_0, \ldots, C_n\},\\
\rho(A_1) &= e(e(ab)c),\\
\rho(A_2) &= e(a\,e(bc)),\\
\rho(B_0) &= A_1 x A_2,\\
\rho(B_i) &= B_{i-1}\langle B_{i-1}\rangle \text{ for } 1 \leq i \leq n,\\
\rho(B) &= B_n\langle A_1\rangle,\\
\rho(C_0) &= d(xB),\\
\rho(C_i) &= C_{i-1}\langle C_{i-1}\rangle \text{ for } 1 \leq i \leq n, \text{ and}\\
\rho(S_1) &= C_n\langle B\rangle.
\end{aligned}$$

Note that, although F has size $\mathcal{O}(n)$, $[\![F]\!]$ has exponential width and depth, as it is the tree

$$\underbrace{d(d(\cdots d(d(}_{2^n \text{ many } d(}f\,\underbrace{f)f)\cdots f)f)}_{2^n \text{ many } f)},$$

where $f = [\![B]\!]_F$ is the forest $(e(e(ab)c))^{2^n+1}(e(a\,e(bc)))^{2^n}$; see Figure 3 for $n = 1$.

Now consider a second FSLP $F' = (V', S_2, \rho')$ over Σ with

$$\begin{aligned}
V_0' &= \{D, E_0, \ldots, E_n, E, S_2\},\\
V_1' &= \{F_0, \ldots, F_n\},\\
\rho(D) &= e(abc),\\
\rho(E_0) &= DD,\\
\rho(E_i) &= E_{i-1}E_{i-1} \text{ for } 1 \leq i \leq n,\\
\rho(E) &= E_n D,\\
\rho(F_0) &= d(Ex),\\
\rho(F_i) &= F_{i-1}\langle F_{i-1}\rangle \text{ for } 1 \leq i \leq n, \text{ and}\\
\rho(S_2) &= F_n\langle E\rangle.
\end{aligned}$$

Then $[\![F']\!]$ is the tree

$$\underbrace{d(f'd(f'\cdots d(f'd(f'}_{2^n \text{ many } d(f'}\,f'))\cdots)),$$

where $f' = [\![E]\!]_{F'}$ is the forest $e(abc)^{2^{n+1}+1}$; see Figure 3 for $n = 1$.

Note that if we consider e as associative then f and f' represent the same forest. If in addition we consider d as commutative then the FSLPs F and F' in fact represent the same unranked tree. Our main contribution is a polynomial time algorithm for performing this kind of equivalence check.

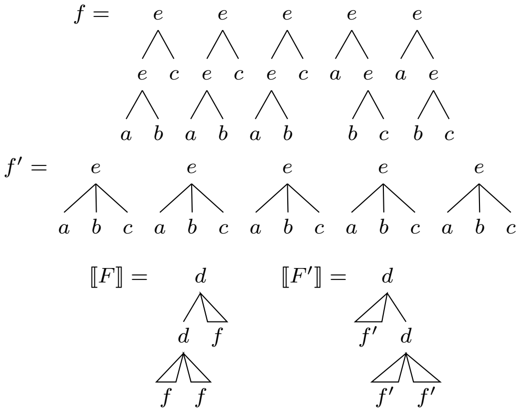

Fig. 3 Example 4 for $n = 1$.

FSLPs generalize *tree straight-line programs* (TSLPs for short) that have been used for the compression of ranked trees before, see e.g. [21]. In this work, we only use TSLPs for binary trees. A TSLP over Σ can then be defined as an FSLP $T = (V, S, \rho)$ such that for every $A \in V$, $\rho(A)$ has the form a, $a(BC)$, $a(xB)$, $a(Bx)$, or $B\langle C\rangle$ with $a \in \Sigma$, $B, C \in V$. TSLPs can be used to compress the fcns-encoding of an unranked tree; see also [22]. It is not hard to see that an FSLP F that produces a binary tree can be transformed into a TSLP T such that $[\![F]\!] = [\![T]\!]$ and $|T| \in \mathcal{O}(|F|)$. This is an easy corollary of our normal form for FSLPs that we introduce in Section 3.5 (see also the proof of Proposition 3). For the construction of the normal form, we first need a technical result about SSLP factorizations that we show in the next section.

3.4 Factorization of SSLPs

Let Σ be an alphabet, let $\Sigma_1 \subseteq \Sigma$ and $\Sigma_2 = \Sigma \setminus \Sigma_1$. Then every string $w \in \Sigma^*$ has a unique factorization $w = v_0 a_1 v_1 \cdots a_n v_n$ with $n \geq 0$, $a_i \in \Sigma_1$ and $v_0, v_i \in \Sigma_2^*$ for $1 \leq i \leq n$, which we call the *Σ_1-factorization of w*. Let $G = (V, \rho)$ and $G' = (V', \rho')$ be SSLPs over Σ. We call G' a *Σ_1-factorization of G* if $[\![A]\!]_G = [\![A]\!]_{G'}$ for all $A \in V$, and there are sets $\mathcal{U}, \mathcal{L}$ of (*upper* and *lower*) variables such that $V' = V \uplus \mathcal{U} \uplus \mathcal{L}$ and

$$\begin{aligned}\rho'(V) &\subseteq \mathcal{L} \cup \mathcal{L}\Sigma_1\mathcal{L} \cup \mathcal{L}\mathcal{U}\Sigma_1\mathcal{L},\\ \rho'(\mathcal{U}) &\subseteq \Sigma_1\mathcal{L} \cup \mathcal{U}\mathcal{U},\\ \rho'(\mathcal{L}) &\subseteq \{\varepsilon\} \cup \Sigma_2 \cup \mathcal{L}\mathcal{L}.\end{aligned}$$

Note that the partition $V' = V \uplus \mathcal{U} \uplus \mathcal{L}$ is uniquely determined by V' and ρ. Moreover, $[\![A]\!]_{G'} \in \Sigma_2^*$ for every $A \in \mathcal{L}$ and $[\![A]\!]_{G'} \in (\Sigma_1\Sigma_2^*)^*$ for every $A \in \mathcal{U}$. This implies that G' describes the Σ_1-factorization $w = v_0 a_1 v_1 \cdots a_n v_n$ for every string $w = [\![A]\!]_{G'} = [\![A]\!]_G$ ($A \in V$) in the following sense: If $\rho'(A) =$

$B \in \mathcal{L}$, then $n = 0$ and $[\![B]\!]_{G'} = v_0$. If $\rho'(A) = BaC \in \mathcal{L}\Sigma_1\mathcal{L}$, then $n = 1$, $[\![B]\!]_{G'} = v_0$, $a = a_1$ and $[\![C]\!]_{G'} = v_1$. Finally, if $\rho'(A) = BCaD \in \mathcal{L}\mathcal{U}\Sigma_1\mathcal{L}$ then $n \geq 2$, $[\![B]\!]_{G'} = v_0$, $a = a_n$, $[\![D]\!]_{G'} = v_n$ and there are variables C_i, D_i with $[\![C]\!]_{G'} = [\![C_1]\!]_{G'} \cdots [\![C_{n-1}]\!]_{G'}$, $\rho(C_i) = a_i D_i$ and $[\![D_i]\!]_{G'} = v_i$ for $1 \leq i < n$.

Lemma 1 *Let $G = (V, \rho)$ be an SSLP over Σ and let $\Sigma_1 \subseteq \Sigma$. We can compute in linear time a Σ_1-factorization of G of size $\mathcal{O}(|G|)$.*

Proof We may assume w.l.o.g. that $\rho(V) \subseteq VV \cup \Sigma$. For every string $w \in \Sigma^*$ with Σ_1-factorization $w = v_0 a_1 v_1 \cdots a_n v_n$ we define $w_\ell, w_m, w_r \in \Sigma^*$ and $\sigma_w \in \Sigma_1 \cup \{\varepsilon\}$ as follows:

- If $n = 0$ then $w_\ell = v_0$ and $w_m = w_r = \sigma_w = \varepsilon$.
- If $n > 0$ then $w_\ell = v_0$, $w_m = a_1 v_1 \cdots a_{n-1} v_{n-1}$, $\sigma_w = a_n$ and $w_r = v_n$.

Note that in both cases $w = w_\ell w_m \sigma_w w_r$ and $w_\ell, w_m, \sigma_w, w_r$ satisfy the following equations:

- If $w = \varepsilon$ then $w_\ell = w_m = \sigma_w = w_r = \varepsilon$.
- If $w = a \in \Sigma_1$ then $\sigma_w = a$ and $w_\ell = w_m = w_r = \varepsilon$.
- If $w = b \in \Sigma_2$ then $w_\ell = b$ and $w_m = \sigma_w = w_r = \varepsilon$.
- If $w = uv$ with $u, v \in \Sigma^*$ then
  - if $\sigma_u = \varepsilon$ then also $u_m = u_r = \varepsilon$, hence $w_\ell = u_\ell v_\ell$, $w_m = v_m$, $\sigma_w = \sigma_v$ and $w_r = v_r$,
  - if $\sigma_u \in \Sigma_1$ and $\sigma_v = \varepsilon$ then also $v_m = v_r = \varepsilon$, hence $w_\ell = u_\ell$, $w_m = u_m$, $\sigma_w = \sigma_u$ and $w_r = u_r v_\ell$,
  - if $\sigma_u, \sigma_v \in \Sigma_1$ then $w_\ell = u_\ell$, $w_m = u_m \sigma_u u_r v_\ell v_m$, $\sigma_w = \sigma_v$ and $w_r = v_r$.

We use these equations as a guideline for the construction of a Σ_1-factorization $G' = (V \uplus \mathcal{U} \uplus \mathcal{L}, \rho')$ of G. Take new variables $A_\ell, A_m, A_r, U_{BC}, U'_{BC}, L_{BC} \notin V$ and let

$$\mathcal{U} = \{A_m \mid A \in V\} \cup \{U_{BC}, U'_{BC} \mid BC \in \rho(V)\},$$
$$\mathcal{L} = \{A_\ell, A_r \mid A \in V\} \cup \{L_{BC} \mid BC \in \rho(V)\}.$$

For every $A \in V$ we define $\sigma_A \in \Sigma_1 \cup \{\varepsilon\}$ and the right-hand sides of the new variables by the following induction on $\leq_G$:

- If $\rho(A) = a \in \Sigma_1$ then $\sigma_A = a$ and $\rho'(A_\ell) = \rho'(A_m) = \rho'(A_r) = \varepsilon$.
- If $\rho(A) = b \in \Sigma_2$ then $\rho'(A_\ell) = b$ and $\rho'(A_m) = \sigma_A = \rho'(A_r) = \varepsilon$.
- If $\rho(A) = BC$ then
  - if $\sigma_B = \varepsilon$ then $\rho'(A_\ell) = B_\ell C_\ell$, $\rho'(A_m) = \rho'(C_m)$, $\sigma_A = \sigma_C$ and $\rho'(A_r) = \rho'(C_r)$,
  - if $\sigma_B \in \Sigma_1$ and $\sigma_C = \varepsilon$ then $\rho'(A_\ell) = \rho'(B_\ell)$, $\rho'(A_m) = \rho'(B_m)$, $\sigma_A = \sigma_B$ and $\rho'(A_r) = B_r C_\ell$,
  - if $\sigma_B, \sigma_C \in \Sigma_1$ then
    - $\rho'(A_\ell) = \rho'(B_\ell)$,
    - $\rho'(A_m) = \rho'(U'_{BC})$ if $\rho'(B_m) = \varepsilon$, otherwise $\rho'(A_m) = B_m U'_{BC}$,
    - $\rho'(U'_{BC}) = \rho'(U_{BC})$ if $\rho'(C_m) = \varepsilon$, otherwise $\rho'(U'_{BC}) = U_{BC} C_m$,
    - $\rho'(U_{BC}) = \sigma_B L_{BC}$,

- $\rho'(L_{BC}) = B_r C_\ell$,
- $\sigma_A = \sigma_C$,
- $\rho'(A_r) = \rho'(C_r)$.

Finally we remove every variable A_m with $\rho'(A_m) = \varepsilon$ from $\mathcal{U}$ and define $\rho'(A)$ for every $A \in V$ as follows: If $\sigma_A = \varepsilon$ then $\rho'(A) = A_\ell \in \mathcal{L}$. If $\sigma_A \in \Sigma_1$ and $A_m \notin \mathcal{U}$ then $\rho'(A) = A_\ell \sigma_A A_r \in \mathcal{L}\Sigma_1\mathcal{L}$. Otherwise $\rho'(A) = A_\ell A_m \sigma_A A_r \in \mathcal{L}\mathcal{U}\Sigma_1\mathcal{L}$.

A straightforward induction on $\leq_G$ shows that $[\![A_\ell]\!]_{G'} = w_\ell$, $[\![A_m]\!]_{G'} = w_m$, $\sigma_A = \sigma_w$ and $[\![A_r]\!]_{G'} = w_r$ whenever $[\![A]\!]_G = w$. From this and the definition of the new right-hand sides $\rho'(A)$ we finally obtain

$$[\![A]\!]_{G'} = ([\![A_\ell]\!]_{G'})([\![A_m]\!]_{G'})\sigma_A([\![A_r]\!]_{G'}) = w_\ell w_m \sigma_w w_r = w.$$

This proves the lemma. □

3.5 Normal form FSLPs

In this subsection, we introduce a normal form for FSLPs that turns out to be crucial in the rest of the paper. An FSLP $F = (V, S, \rho)$ is in *normal form* if $V_0 = V_0^\top \uplus V_0^\perp$ and all right-hand sides have one of the following forms:

- $\rho(A) = \varepsilon$, where $A \in V_0^\top$,
- $\rho(A) = BC$, where $A \in V_0^\top, B, C \in V_0$,
- $\rho(A) = B\langle C\rangle$, where $B \in V_1$ and either $A, C \in V_0^\perp$ or $A, C \in V_1$,
- $\rho(A) = a(B)$, where $A \in V_0^\perp$, $a \in \Sigma$ and $B \in V_0$,
- $\rho(A) = a(BxC)$, where $A \in V_1$, $a \in \Sigma$ and $B, C \in V_0$.

Note that the partition $V_0 = V_0^\top \uplus V_0^\perp$ is uniquely determined by ρ. Also note that variables from V_1 produce tree contexts and variables from $V_0^\perp$ produce trees, whereas variables from $V_0^\top$ produce forests with arbitrarily many trees.

Let $F = (V, S, \rho)$ be a normal form FSLP. Every variable $A \in V_1$ produces a vertical concatenation of (possibly exponentially many) variables, whose right-hand sides have the form $a(BxC)$. This vertical concatenation is called the spine of A. Formally, we split V_1 into $V_1^\top = \{A \in V_1 \mid \exists B, C \in V_1 : \rho(A) = B\langle C\rangle\}$ and $V_1^\perp = V_1 \setminus V_1^\top$. We then define the *vertical SSLP* $F^{\text{◫}} = (V_1^\top, \rho_1)$ over $V_1^\perp$ with $\rho_1(A) = BC$ whenever $\rho(A) = B\langle C\rangle$. For every $A \in V_1$ the string $[\![A]\!]_{F^{\text{◫}}} \in (V_1^\perp)^*$ is called the *spine* of A, denoted by $\mathrm{spine}_F(A)$ or just $\mathrm{spine}(A)$ if F is clear from the context. We also define the *horizontal SSLP* $F^{\boxdot} = (V_0^\top, \rho_0)$ over $V_0^\perp$, where ρ_0 is the restriction of ρ to $V_0^\top$. For every $A \in V_0$ we denote the string $[\![A]\!]_{F^{\boxdot}} \in (V_0^\perp)^*$ by $\mathrm{hor}_F(A)$ or just $\mathrm{hor}(A)$ if F is clear from the context. Note that $\mathrm{spine}(A) = A$ (resp., $\mathrm{hor}(A) = A$) for every $A \in V_1^\perp$ (resp., $A \in V_0^\perp$).

The intuition behind the normal form can be explained as follows: Consider a tree context $t \in \mathcal{T}_1(\Sigma)\setminus\{x\}$. By decomposing t along the nodes on the unique path from the root to the x-labelled leaf, we can write t as a vertical concatenation of tree contexts $a_1(f_1 x g_1), \ldots, a_n(f_n x g_n)$ for forests $f_1, g_1, \ldots, f_n, g_n$

and symbols $a_1, \ldots, a_n$. In a normal form FSLP one would produce t by first deriving a vertical concatenation $A_1\langle \cdots \langle A_n \rangle \cdots \rangle$. Every A_i is then derived to $a_i(B_i x C_i)$, where B_i (resp., C_i) produces the forest f_i (resp., g_i). Computing an FSLP for this decomposition for a tree context that is already given by an FSLP is the main step in the proof of the normal form theorem below. Another insight is that proper forest contexts from $\mathcal{F}_1(\varSigma) \setminus \mathcal{T}_1(\varSigma)$ can be eliminated without significant size blow-up.

Theorem 1 *From a given FSLP F one can construct in linear time an FSLP F' in normal form such that $[\![F']\!] = [\![F]\!]$ and $|F'| \in \mathcal{O}(|F|)$.*

Proof To convert an FSLP to normal form, we first introduce a *weak normal form*, where all right-hand sides have one of the following forms:

- $\rho(A) = \varepsilon$, where $A \in V_0$,
- $\rho(A) = B\langle C \rangle$, where $A, C \in V_0$ and $B \in V_1$,
- $\rho(A) = B\langle C \rangle$, where $A, B, C \in V_1$,
- $\rho(A) = a(x)$, where $A \in V_1$, $a \in \varSigma$,
- $\rho(A) = BxC$, where $A \in V_1$, $B, C \in V_0$.

Converting an FSLP into weak normal form is straightforward: By splitting up right-hand sides, we can assume that all right-hand sides have the form $\varepsilon, x, a(x), BC$, or $B\langle C \rangle$ for $a \in \varSigma$, $B, C \in V$. This transformation does not increase the size of the FSLP. Right-hand sides of the form $\rho(A) = BC$, where w.l.o.g. $B \in V_0$, can be replaced by $\rho(A) = B'\langle C \rangle$ and $\rho(B') = Bx$, where B' is a new variable.

We may now assume that $F = (V, S, \rho)$ is in weak normal form. Like we did with FSLPs in normal form, we split V_1 into $V_1^\top = \{A \in V_1 \mid \exists B, C \in V_1 : \rho(A) = B\langle C \rangle\}$ and $V_1^\perp = V_1 \setminus V_1^\top$ and define its vertical SSLP as the SSLP $F^{\boxbar} = (V_1^\top, \rho_1)$ over $V_1^\perp$ with $\rho_1(A) = BC$ whenever $\rho(A) = B\langle C \rangle$.

Let $\mathcal{V} = \{A \in V_1^\perp \mid \rho(A)$ has the form $a(x)\}$ and $\mathcal{H} = V_1^\perp \setminus \mathcal{V}$. Thus, $\rho(A)$ has the form BxC for $A \in \mathcal{H}$. The idea of the construction is to consider maximal factors of the form $A_0 A_1 \cdots A_n$ with $A_0 \in \mathcal{V}$ and $A_1, \ldots, A_n \in \mathcal{H}$ in $[\![A]\!]_{F^{\boxbar}}$ (for some $A \in V_1^\top$). In the FSLP F, such a factor corresponds to an iterated vertical concatenation $A_0\langle A_1 \langle \cdots \langle A_n \rangle \cdots \rangle \rangle$. Assume that $\rho(A_0) = a(x)$ and $\rho(A_i) = B_i x C_i$ for $1 \leq i \leq n$. Then, $A_0\langle A_1 \langle \cdots \langle A_n \rangle \cdots \rangle \rangle$ can be rewritten into $a(B_1 B_2 \cdots B_n x C_n \cdots C_2 C_1)$. We will introduce additional variables in order to produce the horizontal concatenations $B_1 B_2 \cdots B_n$ and $C_n \cdots C_2 C_1$ and a variable with right-hand side $a(BxC)$. Note that the latter form of right-hand sides is allowed in normal form FSLPs.

At this point, $\mathcal{V}$-factorizations turn out to be useful. The maximal factors $A_0 \cdots A_n$ are explicitly generated by a $\mathcal{V}$-factorization of the vertical SSLP $F^{\boxbar}$. By Lemma 1 we can compute in linear time a $\mathcal{V}$-factorization $G = (V_1^\top \uplus \mathcal{U} \uplus \mathcal{L}, \rho_G)$ of $F^{\boxbar}$ with $|G| \in \mathcal{O}(|F^{\boxbar}|) \leq \mathcal{O}(|F|)$. From F and G we obtain the FSLP $F' = (V', S, \rho')$ where $V_0' = V_0 \uplus \{A_\ell, A_r \mid A \in \mathcal{L}\}$ with new variables A_ℓ, A_r, $V_1' = \mathcal{U}$, and ρ' is defined by:

1. If $A \in \mathcal{L}$ with $\rho_G(A) = \varepsilon$ then $\rho'(A_\ell) = \rho'(A_r) = \varepsilon$.

2. If $A \in \mathcal{L}$ with $\rho_G(A) = B \in \mathcal{H}$ and $\rho(B) = CxD$ then $\rho'(A_\ell) = C$ and $\rho'(A_r) = D$.
3. If $A \in \mathcal{L}$ with $\rho_G(A) = BC \in \mathcal{L}\mathcal{L}$ then $\rho'(A_\ell) = B_\ell C_\ell$ and $\rho'(A_r) = C_r B_r$.
4. If $A \in \mathcal{U}$ with $\rho_G(A) = BC \in \mathcal{V}\mathcal{L}$ and $\rho(B) = a(x)$ then $\rho'(A) = a(C_\ell x C_r)$.
5. If $A \in \mathcal{U}$ with $\rho_G(A) = BC \in \mathcal{U}\mathcal{U}$ then $\rho'(A) = B\langle C\rangle$.
6. If $A \in V_0$ with $\rho(A) = \varepsilon$ then $\rho'(A) = \varepsilon$.
7. If $A \in V_0$ with $\rho(A) = B\langle A_0\rangle$, $B \in V_1^\perp$ and $\rho(B) = a(x)$ then $\rho'(A) = a(A_0)$.
8. If $A \in V_0$ with $\rho(A) = B\langle A_0\rangle$, $B \in V_1^\perp$ and $\rho(B) = CxD$ then $\rho'(A) = CA_0D$.
9. If $A \in V_0$ with $\rho(A) = B\langle A_0\rangle$, $B \in V_1^\top$ and $\rho_G(B) = C \in \mathcal{L}$ then $\rho'(A) = C_\ell A_0 C_r$.
10. If $A \in V_0$ with $\rho(A) = B\langle A_0\rangle$, $B \in V_1^\top$, $\rho_G(B) = CDE \in \mathcal{L}\mathcal{V}\mathcal{L}$ and $\rho(D) = a(x)$ then $\rho'(A) = C_\ell a(E_\ell A_0 E_r) C_r$.
11. If $A \in V_0$ with $\rho(A) = B\langle A_0\rangle$, $B \in V_1^\top$, $\rho_G(B) = CDD'E \in \mathcal{L}\mathcal{U}\mathcal{V}\mathcal{L}$ and $\rho(D') = a(x)$ then $\rho'(A) = C_\ell D\langle a(E_\ell A_0 E_r)\rangle C_r$.

Note that this FSLP is not in normal form, but by further splitting up $\rho'(A)$ in points 8–11 (and eliminating the "chain definitions" in point 2), we can obtain normal form. For instance, in point 11, we have to introduce new variables $A_1, \ldots, A_5$ and set $\rho'(A) = A_1 C_r$, $\rho'(A_1) = C_\ell A_2$, $\rho'(A_2) = D\langle A_3\rangle$, $\rho'(A_3) = a\langle A_4\rangle$, $\rho'(A_4) = A_5 E_r$, and $\rho'(A_5) = E_\ell A_0$. An easy induction on $\leq_{F'}$ shows that

- if $B \in \mathcal{L}$ with $\llbracket B\rrbracket_{F^{◫}} = H_1 \cdots H_n \in \mathcal{H}^*$ then $\llbracket H_1\langle \cdots \langle H_n\rangle \cdots\rangle\rrbracket_F = \llbracket B_\ell x B_r\rrbracket_{F'}$,
- if $A \in V_0 \cup \mathcal{U}$ then $\llbracket A\rrbracket_F = \llbracket A\rrbracket_{F'}$.

From the last point we finally obtain $\llbracket F\rrbracket = \llbracket S\rrbracket_F = \llbracket S\rrbracket_{F'} = \llbracket F'\rrbracket$. □

4 Cluster algebras and top dags

In this section we introduce top dags [3,16] as an alternative grammar-based formalism for the compression of unranked trees. A *cluster of rank* 0 is a tree $t \in \mathcal{T}_0(\Sigma)$ of size at least two. A *cluster of rank* 1 is a tree $t \in \mathcal{T}_0(\Sigma)$ of size at least two together with a distinguished leaf node that we call the *bottom boundary node* of t. In both cases, the root of t is called the *top boundary node* of t. Note that in contrast to forest contexts there is no parameter x. Instead, one of the Σ-labelled leaf nodes may be declared as the bottom boundary node. When writing a cluster of rank 1 in term representation, we underline the bottom boundary node. For instance $a(b\,c(\underline{a}\,b))$ is a cluster of rank 1. An *atomic cluster* is of the form $a(b)$ or $a(\underline{b})$ for $a, b \in \Sigma$. Let $\mathcal{C}_i(\Sigma)$ be the set of all clusters of rank $i \in \{0,1\}$ and let $\mathcal{C}(\Sigma) = \mathcal{C}_0(\Sigma) \cup \mathcal{C}_1(\Sigma)$. We write $\text{rank}(s) = i$ if $s \in \mathcal{C}_i(\Sigma)$ for $i \in \{0,1\}$. We define the *cluster algebra* $\mathsf{CA}(\Sigma) = (\mathcal{C}(\Sigma), \odot, \varobar, (a(b), a(\underline{b}))_{a,b\in\Sigma})$ as follows:

- $\odot$ is the horizontal merge operator: $s \odot t$ is only defined if $\text{rank}(s) + \text{rank}(t) \leq 1$ and s, t are of the form $s = a(f)$, $t = a(g)$, i.e., the root labels coincide.

Then $s \odot t = a(fg)$. Note that at most one symbol in the forest fg is underlined. The rank of $s \odot t$ is $\mathrm{rank}(s) + \mathrm{rank}(t)$. For instance,

$$a(b\,c(\underline{a}\,b)) \odot a(b\,c) = a(b\,c(\underline{a}\,b)b\,c).$$

- $\varobar$ is the vertical merge operator: $s \varobar t$ is only defined if $s \in \mathcal{C}_1(\Sigma)$ and the label of the root of t (say a) is equal to the label of the bottom boundary node of s. We then obtain $s \varobar t$ by replacing the unique occurrence of $\underline{a}$ in s by t. The rank of $s \varobar t$ is $\mathrm{rank}(t)$. For instance,

$$a(b\,c(\underline{a}\,b)) \varobar a(b\underline{c}) = a(b\,c(a(b\underline{c})\,b)).$$

- The atomic clusters $a(b)$ and $a(\underline{b})$ are constants of the cluster algebra.

A *top tree* for a tree $t \in \mathcal{T}_0$ is an algebraic expression e over the algebra $\mathsf{CA}(\Sigma)$ such that $[\![e]\!] = t$. A *top dag* over Σ is a valid straight-line program D over the algebra $\mathsf{CA}(\Sigma)$ such that $[\![D]\!] \in \mathcal{T}_0(\Sigma)$. In our terminology, cluster straight-line program would be a more appropriate name, but we prefer to call them top dags.

Example 5 Consider the top dag $D = (\{S, A_0, \ldots, A_n, B_0, \ldots, B_n\}, S, \rho)$ with $\rho(A_0) = b(a)$, $\rho(A_i) = A_{i-1} \odot A_{i-1}$ for $1 \le i \le n$, $\rho(B_0) = A_n \odot b(\underline{b}) \odot A_n$, $\rho(B_i) = B_{i-1} \varobar B_{i-1}$ for $1 \le i \le n$, and $\rho(S) = B_n \varobar b(c)$. We have

$$[\![D]\!] = b(a^{2^n} b(a^{2^n} \cdots b(a^{2^n} b(c)\, a^{2^n}) \cdots a^{2^n})\, a^{2^n}),$$

where b occurs $2^n + 1$ many times.

5 Relative succinctness

We have now three grammar-based formalisms for the compression of unranked trees: FSLPs, top dags, and TSLPs for fcns-encodings. In this section we study their relative succinctness. It turns out that up to multiplicative factors of size $|\Sigma|$ (number of node labels) all three formalisms are equally succinct. Moreover, the transformations between the formalisms can be computed in time linear in the output size. This allows us to transfer algorithmic results for FSLPs to top dags and TSLPs for fcns encodings, and vice versa. We start with top dags:

Proposition 1 *For a given top dag D one can compute in linear time an FSLP F such that $[\![F]\!] = [\![D]\!]$ and $|F| \in \mathcal{O}(|D|)$.*

Proof For $t \in \mathcal{T}(\Sigma)$ we denote with $\triangle(t)$ the forest obtained by removing from t the root node. Translating a cluster with a bottom boundary node to a tree with a parameter is done by the function $\triangledown_{\mathrm{x}} \colon \mathcal{C}_1(\Sigma) \to \mathcal{T}_1(\Sigma)$, where $\triangledown_{\mathrm{x}}(t)$ replaces the bottom boundary node in t labelled with $a \in \Sigma$ by the tree $a(x)$. We translate a cluster to a forest by $\varphi\colon \mathcal{C}(\Sigma) \to \mathcal{F}(\Sigma)$, where $\varphi(t) = \triangle(t)$ for

$t \in \mathcal{C}_0(\Sigma)$ and $\varphi(t) = \triangle(\triangledown_{\mathrm{x}}(t))$ for $t \in \mathcal{C}_1(\Sigma)$. Then the following identities hold:

$$\varphi(s) \boxdot \varphi(t) = \varphi(s \odot t), \tag{1}$$

$$\varphi(s) \boxbar \varphi(t) = \varphi(s \obar t), \tag{2}$$

$$\varphi(a(b)) = b, \tag{3}$$

$$\varphi(a(\underline{b})) = b(x). \tag{4}$$

Let $D = (V, S, \rho)$ be a top dag and let α be the label of the root of $[\![D]\!]$, which can be easily computed in linear time. We define $F = (V \uplus \{S'\}, S', \rho')$, such that for every $A \in V$ we have $[\![A]\!]_F = \varphi([\![A]\!]_D)$. We set $\rho'(S') = \alpha(S)$, which yields

$$[\![F]\!] = [\![S']\!]_F = [\![\alpha(S)]\!]_F = \alpha([\![S]\!]_F) = \alpha(\triangle([\![S]\!]_D)) = [\![S]\!]_D = [\![D]\!].$$

We translate the right-hand sides of the top dag as follows:

- If $\rho(A) = a(b)$ then $\rho'(A) = b$.
- If $\rho(A) = a(\underline{b})$ then $\rho'(A) = b(x)$.
- If $\rho(A) = B \odot C$ then $\rho'(A) = B \boxdot C$.
- If $\rho(A) = B \obar C$ then $\rho'(A) = B \boxbar C$.

Then $[\![A]\!]_F = \varphi([\![A]\!]_D)$ for all $A \in V$ follows immediately from (1)–(4). □

Proposition 2 *For a given FSLP F with $[\![F]\!] \in \mathcal{T}_0(\Sigma)$ and $|[\![F]\!]| \geq 2$ one can compute in time $\mathcal{O}(|\Sigma| \cdot |F|)$ a top dag D such that $[\![D]\!] = [\![F]\!]$ and $|D| \in \mathcal{O}(|\Sigma| \cdot |F|)$.*

Proof For every $a \in \Sigma$ we define the mapping $\psi_a \colon \mathcal{T}_1(\Sigma) \setminus \{x\} \to \mathcal{C}_1(\Sigma)$ as follows: for $t \in \mathcal{T}_1(\Sigma)$, $t \neq x$, let $\psi_a(t)$ be the rank-1 cluster obtained from replacing in t the label of the unique x-labelled node (which is not the root) by a and declaring this node as the bottom boundary node. Then, the following identities are obvious, where $s, t \in \mathcal{T}_1(\Sigma) \setminus \{x\}$, $u \in \mathcal{T}_0(\Sigma)$, $|u| \geq 2$, and $b \in \Sigma$ is the label of the roots of t and u:

$$\psi_a(s\langle t\rangle) = \psi_b(s) \obar \psi_a(t), \tag{5}$$

$$s\langle u\rangle = \psi_b(s) \obar u. \tag{6}$$

Moreover, for all forests $f, g \in \mathcal{F}_0(\Sigma)$ with $f \neq \varepsilon \neq g$ we have

$$a(fg) = a(f) \odot a(g). \tag{7}$$

Let us now come to the construction for T. By Theorem 1 we can assume that the input FSLP $F = (V, S, \rho)$ is in normal form. We can easily eliminate right-hand sides of the form ε without a size increase. This might lead to "chain definitions" of the form $\rho(A) = B$ which can be also eliminated without size increase. After this preprocessing step, we may have also right-hand sides of the form $\rho(A) = a \in \Sigma$ (with $A \in V_0^{\perp}$), $\rho(A) = a(x)$, $\rho(A) = a(Bx)$ (with $B \in V_0$), and $\rho(A) = a(xC)$ (with $C \in V_0$). We still denote the resulting FSLP with F. Since we started with an FSLP in normal form, we have $[\![A]\!]_F \in \mathcal{T}_0(\Sigma)$ for every $A \in V_0^\perp$ and $[\![A]\!]_F \in \mathcal{T}_1(\Sigma) \setminus \{x\}$ for every $A \in V_1$. Hence, for $A \in V_0^\perp \cup V_1$ we can define $\alpha_A \in \Sigma$ as the label of the root node in the tree (context) $[\![A]\!]_F$. Also note that every forest $[\![A]\!]_F$ for $A \in V_0$ has size at least one. Moreover, if $A \in V_0^\perp$ and $\rho(A) \notin \Sigma$ then the tree $[\![A]\!]_F$ has size at least two. Let $U_0^\perp = \{A \in V_0^\perp \mid \rho(A) \notin \Sigma\}$.

We define a top dag $D = (V', S, \rho')$, where $V' = V_0' \cup V_1'$ with

$$\begin{aligned} V_0' &= U_0^\perp \uplus \{A^a \mid A \in V_0, a \in \Sigma\}, \\ V_1' &= \{A_a \mid A \in V_1, a \in \Sigma\}. \end{aligned}$$

We will define the right-hand side mapping ρ' of D such that the following identities hold:

$$\begin{aligned} [\![A]\!]_D &= [\![A]\!]_F \text{ for every } A \in U_0^\perp, \\ [\![A^a]\!]_D &= a([\![A]\!]_F) \text{ for every } A \in V_0, \\ [\![A_a]\!]_D &= \psi_a([\![A]\!]_F) \text{ for every } A \in V_1. \end{aligned}$$

In order to obtain these identities, we define ρ' as follows:

- If $\rho(A) = BC$ for $A, B, C \in V_0$ then $\rho'(A^a) = B^a \odot C^a$.
- If $A \in U_0^\perp$ then $\rho'(A^a) = a(\underline{\alpha_A}) \oplus A$.
- If $\rho(A) = b \in \Sigma$ then $\rho'(A^a) = a(b)$.
- If $\rho(A) = a(B)$ (hence $A \in U_0^\perp$) then $\rho'(A) = B^a$.
- If $\rho(A) = B\langle C\rangle$ for $A, C \in U_0^\perp$ and $B \in V_1$ then $\rho'(A) = B_{\alpha_C} \oplus C$.
- If $\rho(A) = B\langle C\rangle$, $\rho(C) = a \in \Sigma$ and $C \in V_1$ (hence $A \in U_0^\perp$) then $\rho'(A) = B_a$.
- If $\rho(A) = B\langle C\rangle$ for $A, B, C \in V_1$ then $\rho'(A_a) = B_{\alpha_C}\langle C_a\rangle$.
- If $\rho(A) = b(BxC)$ for $A \in V_1$, $B, C \in V_0$ then $\rho'(A_a) = B^b \odot b(\underline{a}) \odot C^b$.
- If $\rho(A) = b(Bx)$ for $A \in V_1$, $B \in V_0$ then $\rho'(A_a) = B^b \odot b(\underline{a})$.
- If $\rho(A) = b(xC)$ for $A \in V_1$, $C \in V_0$ then $\rho'(A_a) = b(\underline{a}) \odot C^b$.
- If $\rho(A) = b(x)$ for $A \in V_1$ then $\rho'(A_a) = b(\underline{a})$.

The correctness of this construction follows easily by induction, using (5)–(7).

To conclude the proof, note that since $[\![F]\!]$ is a tree of size two, the start symbol S of F must belong to $U_0^\perp$. Hence, the above point (i) implies $[\![D]\!] = [\![F]\!]$. □

The following example shows that the size bound in Proposition 2 is sharp:

Example 6 Let $\Sigma = \{a, a_1, \dots, a_\sigma\}$ and let $t_n = a(a_1(a^m) \cdots a_\sigma(a^m))$ where $n \geq 1$ and $m = 2^n$. For every $n > \sigma$ the tree t_n can be produced by an FSLP of size $\mathcal{O}(n)$: using $n = \log m$ many variables we can produce the forest a^m and then $\mathcal{O}(n + \sigma) = \mathcal{O}(n)$ many additional variables suffice to produce t_n. On the other hand, every top dag for t_n has size $\Omega(\sigma \cdot n)$: consider a top tree e that evaluates to t_n. Then e must contain a subexpression e_i that evaluates to the subtree $a_i(a^m)$ $(1 \leq i \leq \sigma)$ of t_n. The subexpression e_i has to produce $a_i(a^m)$ using the $\odot$-operation from copies of $a_i(a)$. Hence, the expression for $a_i(a^m)$ has size $n = \log_2 m$ and different e_i contain no identical subexpressions. Therefore every top dag for t_n has size at least $\sigma \cdot n$.

In contrast, FSLPs and TSLPs for fcns-encodings turn out to be equally succinct up to constant factors:

Proposition 3 *Let $f \in \mathcal{F}(\Sigma)$ be a forest and let F be an FSLP (or TSLP) over $\Sigma \uplus \{\bot\}$ with $[\![F]\!] = \mathrm{fcns}(f)$. Then we can transform F in linear time into an FSLP F' over Σ with $[\![F']\!] = f$ and $|F'| \in \mathcal{O}(|F|)$.*

Proof Let $F = (V, S, \rho)$ be an FSLP over $\Sigma \cup \{\bot\}$. By Theorem 1, we may assume that F is in normal form and every variable is reachable from S. This implies $|\operatorname{hor}(A)| \leq 2$ for every $A \in V_0$, because $\mathrm{fcns}(f)$ is a binary tree. Hence we can compute the strings $\operatorname{hor}(A) = [\![A]\!]_{F^\boxdot} \in (V_0^\perp)^*$ with $A \in V_0^\top$ all together in linear time, substitute $\operatorname{hor}(A)$ for each occurrence of A in the right-hand sides, and finally erase the production for A. In particular, right-hand sides of the form ε and BC do not occur any more. Moreover, right-hand sides of the form $a(BxC)$ and $a(B)$ will be transformed as follows by the above replacement: In the first case $(a(BxC))$ we have $a \in \Sigma$ and $|\operatorname{hor}(B)| + |\operatorname{hor}(C)| = 1$. Hence the substitution leads to $a(Dx)$ or $a(xD)$ with $D \in V_0^\perp$. In the second case $(a(B))$ either $a = \bot$ and $|\operatorname{hor}(B)| = 0$ or $a \in \Sigma$ and $|\operatorname{hor}(B)| = 2$, hence the substitution leads to $\bot$ or $a(CD)$ with $C, D \in V_0^\perp$. Thus we finally obtain an FSLP in which all right-hand sides have one of the following forms:

- $\bot$,
- $a(BC)$,
- $a(Bx)$,
- $a(xB)$,
- $B\langle C\rangle$.

This is in fact a TSLP as defined in Section 3. We can now easily translate right-hand sides of the above forms into right-hand sides of an FSLP F' for f:

- $\rho(A) = \bot$ becomes $\rho(A) = \varepsilon$.
- $\rho(A) = a(BC)$ becomes $\rho(A) = a(B)C$.
- $\rho(A) = a(Bx)$ becomes $\rho(A) = a(B)x$.
- $\rho(A) = a(xB)$ becomes $\rho(A) = a(x)B$.
- $\rho(A) = B\langle C\rangle$ stays the same.

For the correctness of the construction, we have to show that $\mathrm{fcns}([\![F']\!]) = [\![F]\!]$. In order to do this, we show the following properties:

$$\begin{aligned} \mathrm{fcns}([\![A]\!]_{F'}) &= [\![A]\!]_F \text{ for all } A \in V_0, \\ \mathrm{fcns}([\![A]\!]_{F'}\langle f\rangle) &= [\![A]\!]_F\langle \mathrm{fcns}(f)\rangle \text{ for all } A \in V_1,\ f \in \mathcal{F}_0(\Sigma). \end{aligned}$$

These are shown using a simple induction and case analysis:

- $\rho(A) = \bot$: $\mathrm{fcns}([\![A]\!]_{F'}) = \mathrm{fcns}(\varepsilon) = \bot = [\![A]\!]_F$.
- $\rho(A) = a(BC)$: We obtain ("ind" refers to induction on B and C)

$$\begin{aligned} \mathrm{fcns}([\![A]\!]_{F'}) &= \mathrm{fcns}([\![a(B)C]\!]_{F'}) \\ &= \mathrm{fcns}(a([\![B]\!]_{F'})[\![C]\!]_{F'}) \\ &= a(\mathrm{fcns}([\![B]\!]_{F'})\,\mathrm{fcns}([\![C]\!]_{F'})) \\ &\overset{\text{ind}}{=} a([\![B]\!]_F[\![C]\!]_F) = [\![A]\!]_F. \end{aligned}$$

- $\rho(A) = a(Bx)$: We obtain

$$\begin{aligned}
\mathrm{fcns}([\![A]\!]_{F'}\langle f\rangle) &= \mathrm{fcns}([\![a(B)x]\!]_{F'}\langle f\rangle)\\
&= \mathrm{fcns}(a([\![B]\!]_{F'})f)\\
&= a(\mathrm{fcns}([\![B]\!]_{F'})\,\mathrm{fcns}(f))\\
&\overset{\mathrm{ind}}{=} a([\![B]\!]_F\,\mathrm{fcns}(f))\\
&= [\![a(Bx)]\!]_F\langle \mathrm{fcns}(f)\rangle = [\![A]\!]_F\langle \mathrm{fcns}(f)\rangle.
\end{aligned}$$

- $\rho(A) = a(xB)$: We obtain

$$\begin{aligned}
\mathrm{fcns}([\![A]\!]_{F'}\langle f\rangle) &= \mathrm{fcns}([\![a(x)B]\!]_{F'}\langle f\rangle)\\
&= \mathrm{fcns}(a(f)[\![B]\!]_{F'})\\
&= a(\mathrm{fcns}(f)\,\mathrm{fcns}([\![B]\!]_{F'}))\\
&\overset{\mathrm{ind}}{=} a(\mathrm{fcns}(f)[\![B]\!]_F)\\
&= [\![a(xB)]\!]_F\langle \mathrm{fcns}(f)\rangle = [\![A]\!]_F\langle \mathrm{fcns}(f)\rangle.
\end{aligned}$$

- $\rho(A) = B\langle C\rangle$ with $C \in V_0$: We obtain the following, where the first (resp., second) induction step uses induction on B (resp., C):

$$\begin{aligned}
\mathrm{fcns}([\![A]\!]_{F'}) &= \mathrm{fcns}([\![B\langle C\rangle]\!]_{F'})\\
&= \mathrm{fcns}([\![B]\!]_{F'}\langle [\![C]\!]_{F'}\rangle)\\
&\overset{\mathrm{ind}}{=} [\![B]\!]_F\langle \mathrm{fcns}([\![C]\!]_{F'})\rangle\\
&\overset{\mathrm{ind}}{=} [\![B]\!]_F\langle [\![C]\!]_F\rangle\\
&= [\![B\langle C\rangle]\!]_F = [\![A]\!]_F.
\end{aligned}$$

- $\rho(A) = B\langle C\rangle$ with $C \in V_1$: We obtain

$$\begin{aligned}
\mathrm{fcns}([\![A]\!]_{F'}\langle f\rangle) &= \mathrm{fcns}([\![B\langle C\rangle]\!]_{F'}\langle f\rangle)\\
&= \mathrm{fcns}(([\![B]\!]_{F'}\langle [\![C]\!]_{F'}\rangle)\langle f\rangle)\\
&= \mathrm{fcns}([\![B]\!]_{F'}\langle [\![C]\!]_{F'}\langle f\rangle\rangle)\\
&\overset{\mathrm{ind}}{=} [\![B]\!]_F\langle \mathrm{fcns}([\![C]\!]_{F'}\langle f\rangle)\rangle\\
&\overset{\mathrm{ind}}{=} [\![B]\!]_F\langle [\![C]\!]_F\langle \mathrm{fcns}(f)\rangle\rangle\\
&= [\![B\langle C\rangle]\!]_F\langle \mathrm{fcns}(f)\rangle = [\![A]\!]_F\langle \mathrm{fcns}(f)\rangle.
\end{aligned}$$

This concludes the proof of the proposition. □

Proposition 4 *For every FSLP F over Σ, we can construct in linear time a TSLP T over $\Sigma \cup \{\perp\}$ with $[\![T]\!] = \mathrm{fcns}([\![F]\!])$ and $|T| \in \mathcal{O}(|F|)$.*

Proof We start with the definition of two functions which are closely related to fcns. The first function $\pi : \mathcal{F}_0(\Sigma) \to \mathcal{T}_1(\Sigma \cup \{\perp\})$ is defined inductively by

$$\begin{aligned}
\pi(\varepsilon) &= x,\\
\pi(a(f)g) &= a(\mathrm{fcns}(f)\pi(g)) \text{ for all } a \in \Sigma, f, g \in \mathcal{F}_0(\Sigma);
\end{aligned}$$

in particular its restriction to $\mathcal{T}_0(\Sigma)$ is given by

$$\pi(a(f)) = a(\mathrm{fcns}(f)\,x) \text{ for all } a \in \Sigma, f \in \mathcal{F}_0(\Sigma). \tag{8}$$

Simple inductive proofs show that

$$\begin{align} \mathrm{fcns}(f) &= \pi(f)\langle\bot\rangle \text{ for all } f \in \mathcal{F}_0(\Sigma), \tag{9}\\ \pi(fg) &= \pi(f)\langle\pi(g)\rangle \text{ for all } f, g \in \mathcal{F}_0(\Sigma). \tag{10} \end{align}$$

The second function $\varphi : \mathcal{F}(\Sigma) \to \mathcal{T}(\Sigma \cup \{\bot\})$ is defined inductively by

$$\begin{align*} \varphi(\varepsilon) &= \bot,\\ \varphi(xg) &= x \text{ for all } g \in \mathcal{F}_0(\Sigma),\\ \varphi(a(f)g) &= a(\varphi(f)\varphi(g)) \text{ for all } a \in \Sigma, f, g \in \mathcal{F}(\Sigma) \text{ with } a(f)g \in \mathcal{F}(\Sigma). \end{align*}$$

Simple inductive proofs show that

$$\begin{align} \mathrm{fcns}(f) &= \varphi(f) \text{ for all } f \in \mathcal{F}_0(\Sigma), \tag{11}\\ \varphi(fxg) &= \pi(f) \text{ for all } f, g \in \mathcal{F}_0(\Sigma). \tag{12} \end{align}$$

The most important equation for φ is

$$\begin{align} \varphi(f\langle a(g)\rangle) = {} & \varphi(f)\langle a(\varphi(g)\,\mathrm{fcns}(\mathrm{sib}(f)))\rangle \tag{13}\\ & \text{for all } f \in \mathcal{F}_1(\Sigma), a \in \Sigma, g \in \mathcal{F}_0(\Sigma), \end{align}$$

where $\mathrm{sib}(f) \in \mathcal{F}_0(\Sigma)$ denotes the sequence of all right siblings of x in $f \in \mathcal{F}_1(\Sigma)$, i.e.,

$$\begin{align*} \mathrm{sib}(xg) &= g \text{ for all } g \in \mathcal{F}_0(\Sigma),\\ \mathrm{sib}(a(f)g) &= \begin{cases} \mathrm{sib}(f) \text{ if } f \in \mathcal{F}_1(\Sigma) \text{ and } g \in \mathcal{F}_0(\Sigma),\\ \mathrm{sib}(g) \text{ if } f \in \mathcal{F}_0(\Sigma) \text{ and } g \in \mathcal{F}_1(\Sigma). \end{cases} \end{align*}$$

Equation (13) tells us how to obtain $\varphi(f\langle a(g)\rangle)$ from $\varphi(f)$ and $\varphi(g)$. For its proof note that $a(\varphi(g)\,\mathrm{fcns}(\mathrm{sib}(f))) = a(\varphi(g)\varphi(\mathrm{sib}(f))) = \varphi(a(g)\,\mathrm{sib}(f))$. Hence it suffices to prove

$$\varphi(f\langle t\rangle) = \varphi(f)\langle\varphi(t\,\mathrm{sib}(f))\rangle \text{ for all } f \in \mathcal{F}_1(\Sigma), t \in \mathcal{T}(\Sigma),$$

which can be done by the following induction on the length of the sequence f (and case distinction on the first element of f):

- $f = xg$:

  Then $g \in \mathcal{F}_0(\Sigma)$ and we have

$$\varphi(f\langle t\rangle) = \varphi(tg) = x\langle\varphi(tg)\rangle = \varphi(f)\langle\varphi(tg)\rangle = \varphi(f)\langle\varphi(t\,\mathrm{sib}(f))\rangle.$$

- $f = a(g)h$ with $a \in \Sigma$ and $g \in \mathcal{F}_0(\Sigma)$:
  Then $h \in \mathcal{F}_1(\Sigma)$ and we obtain

$$\begin{aligned}\varphi(f\langle t\rangle) &= \varphi(a(g)\,h\langle t\rangle)\\ &= a(\varphi(g)\varphi(h\langle t\rangle)) \text{ by the definition of } \varphi\\ &= a(\varphi(g)\varphi(h)\langle\varphi(t\,\mathrm{sib}(h))\rangle) \text{ by induction for } h\\ &= a(\varphi(g)\varphi(h)\langle\varphi(t\,\mathrm{sib}(f))\rangle) \text{ because } \mathrm{sib}(f) = \mathrm{sib}(h)\\ &= a(\varphi(g)\varphi(h))\langle\varphi(t\,\mathrm{sib}(f))\rangle\\ &= \varphi(f)\langle\varphi(t\,\mathrm{sib}(f))\rangle \text{ by the definition of } \varphi.\end{aligned}$$

- $f = a(g)h$ with $a \in \Sigma$ and $g \in \mathcal{F}_1(\Sigma)$:
  Then $h \in \mathcal{F}_0(\Sigma)$ and the proof is the same as in the previous step except that the roles of g and h are exchanged.

Equations (8) to (13) are a guideline for the construction of the TSLP T. Let $F = (V, S, \rho)$. As usual we may assume that F is in normal form. Then $T = (V', S, \rho')$ where

$$\begin{aligned}V_0' &= V_0 \uplus \{A_\triangle \mid A \in V_0^\perp\} \text{ and}\\ V_1' &= \{A^\pi \mid A \in V_0\} \uplus \{A_\triangle \mid A \in V_1\}\end{aligned}$$

with new variables $A_\triangle, A^\pi \notin V$. To explain the role of these variables let $\Delta\colon \mathcal{T}(\Sigma) \setminus \{x\} \to \mathcal{F}(\Sigma)$ be defined by $\Delta(a(f)) = f$. We want to achieve that

$$[\![A]\!]_T = \mathrm{fcns}([\![A]\!]_F) \text{ for every } A \in V_0, \tag{14}$$

$$[\![A^\pi]\!]_T = \pi([\![A]\!]_F) \text{ for every } A \in V_0, \tag{15}$$

$$[\![A_\triangle]\!]_T = \varphi(\Delta([\![A]\!]_F)) \text{ for every } A \in V_0^\perp \cup V_1. \tag{16}$$

From (14) we obtain $[\![T]\!] = [\![S]\!]_T = \mathrm{fcns}([\![S]\!]_F) = \mathrm{fcns}([\![F]\!])$ which concludes the proof of the proposition (assuming that T satisfies the size bound $|T| \in \mathcal{O}(|F|)$).

It remains to define ρ' in such a way that (14), (15) and (16) are satisfied. For every $A \in V_0^\perp \cup V_1$ let α_A denote the root label of $[\![A]\!]_F$, and for every $A \in V_1$ let $R_A \in V_0$ be a variable with $[\![R_A]\!]_F = \mathrm{sib}([\![A]\!]_F)$. Such a variable exists in V_0, namely

- $R_A = C$ if $\rho(A) = a(BxC)$,
- $R_A = R_C$ if $\rho(A) = B\langle C\rangle$ for $B, C \in V_1$.

Then we define ρ' by

- $\rho'(A) = A^\pi\langle\perp\rangle$ if $A \in V_0$,
- $\rho'(A^\pi) = \alpha_A(A_\triangle x)$ if $A \in V_0^\perp$,
- $\rho'(A^\pi) = x$ if $A \in V_0^\top$ with $\rho(A) = \varepsilon$,
- $\rho'(A^\pi) = B^\pi\langle C^\pi\rangle$ if $A \in V_0^\top$ with $\rho(A) = BC$,
- $\rho'(A_\triangle) = B$ if $A \in V_0^\perp$ with $\rho(A) = a(B)$,

- $\rho'(A_\triangle) = B^\pi$ if $A \in V_1$ with $\rho(A) = a(BxC)$,
- $\rho'(A_\triangle) = B_\triangle\langle\alpha_C(C_\triangle R_B)\rangle$ if $A \in V_0^\perp \cup V_1$ with $\rho(A) = B\langle C\rangle$.

This concludes the definition of the TSLP T. It is clear that T can be constructed from F in linear time and that $|T| \in \mathcal{O}(|F|)$.

Equations (14), (15) and (16) are proved by the following induction on $\leq_T$. Note that $A_\triangle \leq_T A^\pi$ for all $A \in V_0^\perp$ and $A^\pi \leq_T A$ for all $A \in V_0$.

- If $A \in V_0$ then $\rho'(A) = A^\pi\langle\perp\rangle$ and thus

$$\begin{aligned}[\![A]\!]_T &= [\![A^\pi]\!]_T\langle\perp\rangle \\ &= \pi([\![A]\!]_F)\langle\perp\rangle \text{ by induction for } A^\pi \\ &= \mathrm{fcns}([\![A]\!]_F) \text{ by equation (9).}\end{aligned}$$

- If $A \in V_0^\perp$ with $[\![A]\!]_F = a(f)$ then $\rho'(A^\pi) = a(A_\triangle x)$ and thus

$$\begin{aligned}[\![A^\pi]\!]_T &= a([\![A_\triangle]\!]_T x) \\ &= a(\varphi(\mathit{\Delta}([\![A]\!]_F))x) \text{ by induction for } A_\triangle \\ &= a(\varphi(f)x) \\ &= a(\mathrm{fcns}(f)x) \text{ by equation (11)} \\ &= \pi(a(f)) \text{ by equation (8)} \\ &= \pi([\![A]\!]_F).\end{aligned}$$

- If $A \in V_0^\top$ with $\rho(A) = \varepsilon$ then $\rho'(A^\pi) = x$ and thus

$$\begin{aligned}[\![A^\pi]\!]_T &= x \\ &= \pi(\varepsilon) \text{ by the definition of } \pi \\ &= \pi([\![A]\!]_F).\end{aligned}$$

- If $A \in V_0^\top$ with $\rho(A) = BC$ then $\rho'(A^\pi) = B^\pi\langle C^\pi\rangle$ and thus

$$\begin{aligned}[\![A^\pi]\!]_T &= [\![B^\pi]\!]_T\langle[\![C^\pi]\!]_T\rangle \\ &= \pi([\![B]\!]_F)\langle\pi([\![C]\!]_F)\rangle \text{ by induction for } B^\pi \text{ and } C^\pi \\ &= \pi([\![B]\!]_F[\![C]\!]_F) \text{ by equation (10)} \\ &= \pi([\![A]\!]_F).\end{aligned}$$

- If $A \in V_0^\perp$ with $\rho(A) = a(B)$ then $\rho'(A_\triangle) = B$ and thus

$$\begin{aligned}[\![A_\triangle]\!]_T &= [\![B]\!]_T \\ &= \mathrm{fcns}([\![B]\!]_F) \text{ by induction for } B \\ &= \varphi([\![B]\!]_F) \text{ by equation (11)} \\ &= \varphi(\mathit{\Delta}([\![A]\!]_F)).\end{aligned}$$

– If $A \in V_1$ with $\rho(A) = a(BxC)$ then $\rho'(A_\triangle) = B^\pi$ and thus

$$\begin{aligned}[\![A_\triangle]\!]_T &= [\![B^\pi]\!]_T \\ &= \pi([\![B]\!]_F) \text{ by induction for } B \\ &= \varphi([\![B]\!]_F \, x \, [\![C]\!]_F) \text{ by equation (12)} \\ &= \varphi(\mathit{\Delta}([\![A]\!]_F)).\end{aligned}$$

– If $A \in V_0^\perp \cup V_1$ with $\rho(A) = B\langle C\rangle$, $[\![B]\!]_F = b(f)$ and $[\![C]\!]_F = c(g)$ then $\rho'(A) = B_\triangle\langle c(C_\triangle R_B)\rangle$ and we have $[\![R_B]\!]_F = \text{sib}(b(f)) = \text{sib}(f)$. Thus we obtain

$$\begin{aligned}[\![A_\triangle]\!]_T &= [\![B_\triangle]\!]_T\big\langle c([\![C_\triangle]\!]_T[\![R_B]\!]_T)\big\rangle \\ &= \varphi(\mathit{\Delta}([\![B]\!]_F))\big\langle c(\varphi(\mathit{\Delta}([\![C]\!]_F))\,\text{fcns}([\![R_B]\!]_F))\big\rangle \\ &\quad \text{by induction for } B_\triangle, C_\triangle \text{ and } R_B \\ &= \varphi(f)\big\langle c(\varphi(g)\,\text{fcns}(\text{sib}(f)))\big\rangle \\ &= \varphi(f\big\langle c(g)\big\rangle) \text{ by equation (13)} \\ &= \varphi(\mathit{\Delta}(b(f)\big\langle c(g)\big\rangle)) \\ &= \varphi(\mathit{\Delta}([\![B]\!]_F\big\langle[\![C]\!]_F\big\rangle)) \\ &= \varphi(\mathit{\Delta}([\![A]\!]_F)).\end{aligned}$$

This concludes the proof of Proposition 4. □

Proposition 4 and the construction from [9, Proposition 8.3.2] allow to reduce the evaluation of forest automata on FSLPs (for a definition of forest and tree automata, see [9]) to the evaluation of ordinary tree automata on binary trees. The latter problem can be solved in polynomial time [25], which yields:

Corollary 1 *Given a forest automaton A and an FSLP (or top dag) F we can check in polynomial time whether A accepts $[\![F]\!]$.*

Proof First, we construct a TSLP T for $\text{fcns}([\![F]\!])$ using Proposition 4. We also convert A in polynomial time into a tree automaton A' such that A' accepts $\text{fcns}(f)$ if and only if A accepts f, using the construction from [9, Proposition 8.3.2]. Finally, we use the result from [25] to check in polynomial time whether A' accepts $[\![T]\!]$. □

In [3], a linear time algorithm is presented that constructs from a tree of size n with σ many node labels a top dag of size $\mathcal{O}(n/\log_\sigma^{0.19} n)$. In [16] this bound was improved to $\mathcal{O}(n \log\log n/\log_\sigma n)$ (for the same algorithm as in [3]). In [26] we recently presented an alternative construction that achieves the information-theoretic optimum of $\mathcal{O}(n/\log_\sigma n)$; another optimal construction was presented in [11]. Moreover, as in [3], the constructed top dag satisfies the additional size bound $\mathcal{O}(d \cdot \log n)$, where d is the size of the minimal dag of t. With Proposition 1 and 4 we obtain:

Corollary 2 *Let t be a tree of size n with σ many node labels. Then we can construct in linear time an FSLP for t (or a TSLP for* fcns(t)*) of size $\mathcal{O}(n/\log_\sigma n) \cap \mathcal{O}(d \cdot \log n)$, where d is the size of the minimal dag of t.*

6 Testing equality modulo associativity and commutativity

In this section we will give an algorithmic application which proves the usefulness of FSLPs (even if we deal with binary trees). We fix two subsets $\mathcal{A} \subseteq \Sigma$ (the set of *associative symbols*) and $\mathcal{C} \subseteq \Sigma$ (the set of *commutative symbols*). This means that we impose the following identities for all $a \in \mathcal{A}$, $c \in \mathcal{C}$, all trees $t_1, \dots, t_n \in \mathcal{T}_0(\Sigma)$, all permutations $\xi \colon \{1, \dots, n\} \to \{1, \dots, n\}$, and all $1 \le i \le j \le n+1$:

$$a(t_1 \cdots t_n) = a(t_1 \cdots t_{i-1} a(t_i \cdots t_{j-1}) t_j \cdots t_n), \tag{ASSOC}$$

$$c(t_1 \cdots t_n) = c(t_{\xi(1)} \cdots t_{\xi(n)}). \tag{COMM}$$

Note that the standard law of associativity $x \circ (y \circ z) = (x \circ y) \circ z$ for a binary operator $\circ$ is an instance of (ASSOC) if we consider $\circ$ as a symbol in $\mathcal{A}$.

6.1 Associative symbols

Below, we define the associative normal form $\mathrm{nf}_{\mathcal{A}}(f)$ of a forest f and show that from an FSLP F we can compute in linear time an FSLP F' with $[\![F']\!] = \mathrm{nf}_{\mathcal{A}}([\![F]\!])$. For trees $s, t \in \mathcal{T}_0(\Sigma)$ we have that $s = t$ modulo the identities in (ASSOC) if and only if $\mathrm{nf}_{\mathcal{A}}(s) = \mathrm{nf}_{\mathcal{A}}(t)$. The generalization to forests is needed for the induction, where a slight technical problem arises. Whether the forests $t_1 \cdots t_{i-1} a(t_i \cdots t_{j-1}) t_j \cdots t_n$ and $t_1 \cdots t_n$ are equal modulo the identities in (ASSOC) actually depends on the symbol on top of these two forests. If it is an a, and $a \in \mathcal{A}$, then the two forests are equal modulo associativity, otherwise not. To cope with this problem, we use for every associative symbol $a \in \mathcal{A}$ a function $\phi_a \colon \mathcal{F}_0(\Sigma) \to \mathcal{F}_0(\Sigma)$ that pulls up occurrences of a whenever possible.

Let $\bullet \notin \Sigma$ be a new symbol. For every $a \in \Sigma \cup \{\bullet\}$ let $\phi_a \colon \mathcal{F}_0(\Sigma) \to \mathcal{F}_0(\Sigma)$ be defined as follows, where $f \in \mathcal{F}_0(\Sigma)$, $n \ge 0$ and $t_1, \dots, t_n \in \mathcal{T}_0(\Sigma)$:

$$\phi_a(b(f)) = \begin{cases} \phi_a(f) & \text{if } a \in \mathcal{A} \text{ and } a = b, \\ b(\phi_b(f)) & \text{otherwise,} \end{cases}$$

$$\phi_a(t_1 \cdots t_n) = \phi_a(t_1) \cdots \phi_a(t_n).$$

Note that $\phi_a(\varepsilon) = \varepsilon$. We define $\mathrm{nf}_{\mathcal{A}} \colon \mathcal{F}_0(\Sigma) \to \mathcal{F}_0(\Sigma)$ by $\mathrm{nf}_{\mathcal{A}}(f) = \phi_\bullet(f)$.

Example 7 Let $t = a(a(cd)b(cd)a(e))$ and $\mathcal{A} = \{a\}$. We obtain

$$\begin{aligned} \phi_a(t) &= \phi_a(a(cd)b(cd)a(e)) \\ &= \phi_a(a(cd))\phi_a(b(cd))\phi_a(a(e)) \\ &= \phi_a(cd)b(\phi_b(cd))\phi_a(e) \\ &= cdb(cd)e, \\ \phi_b(t) &= a(\phi_a(a(cd)b(cd)a(e))) \\ &= a(cdb(cd)e). \end{aligned}$$

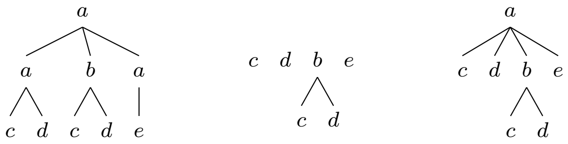

Fig. 4 t on the left, $\phi_a(t)$ in the middle and $\phi_b(t)$ on the right from Example 7

To show the following simple lemma one considers the terminating and confluent rewriting system obtained by directing the equations in (ASSOC) from right to left.

Lemma 2 *For two forests $f_1, f_2 \in \mathcal{F}_0(\Sigma)$, $\mathrm{nf}_{\mathcal{A}}(f_1) = \mathrm{nf}_{\mathcal{A}}(f_2)$ if and only if f_1 and f_2 are equal modulo the identities in* (ASSOC) *for all $a \in \mathcal{A}$.*

Proof Consider the (infinite) term rewriting system consisting of all rules

$$a(t_1 \cdots t_{i-1} a(t_i \cdots t_{j-1}) t_j \cdots t_n) \to a(t_1 \cdots t_n) \qquad (\to\text{-ASSOC})$$

for $a \in \mathcal{A}$, $t_1, \ldots, t_n \in \mathcal{T}_0(\Sigma)$ and $1 \le i \le j \le n+1$. Let $\to_{\mathcal{A}}$ be the resulting rewrite relation. It is clearly terminating. Moreover, by considering all possible overlappings of left-hand sides, one sees that the system is also confluent. Hence, every forest f rewrites into a unique normal form, which is in fact $\mathrm{nf}_{\mathcal{A}}(f)$. The lemma follows since f_1 and f_2 are equal modulo the identities in (ASSOC) if and only if they rewrite into the same normal forms, which means that $\mathrm{nf}_{\mathcal{A}}(f_1) = \mathrm{nf}_{\mathcal{A}}(f_2)$. □

Lemma 3 *From a given FSLP F over Σ one can construct in time $\mathcal{O}(|F| \cdot |\Sigma|)$ an FSLP F' with $[\![F']\!] = \mathrm{nf}_{\mathcal{A}}([\![F]\!])$.*

Proof By Theorem 1, we may assume that $F = (V, S, \rho)$ is in normal form. We introduce new variables A_a for all $a \in \Sigma \cup \{\bullet\}$ and define the right-hand sides of F' such that $[\![A_a]\!]_{F'} = \phi_a([\![A]\!]_F)$ for all $A \in V_0$ and $[\![B_a\langle \phi_b(f)\rangle]\!]_{F'} = \phi_a([\![B\langle f\rangle]\!]_F)$ for all $B \in V_1$, $f \in \mathcal{F}_0(\Sigma)$, where b is the label of the parent node of the parameter x in $[\![B]\!]_F$. This parent node exists since F is in normal form. For every $B \in V_1$ let ω_B be the symbol above x in $[\![B]\!]_F$. These symbols exist by definition of the normal form, and they can be computed all together in linear time. Now let $F' = (V', S_\bullet, \rho')$ where $V' = \{A_a \mid A \in V, a \in \Sigma \cup \{\bullet\}\}$, and ρ' is defined by

- $\rho'(A_a) = \varepsilon$ if $\rho(A) = \varepsilon$,
- $\rho'(A_a) = B_a C_a$ if $\rho(A) = BC$,
- $\rho'(A_a) = B_a\langle C_{\omega_B}\rangle$ if $\rho(A) = B\langle C\rangle$,
- $\rho'(A_a) = B_a$ if $\rho(A) = a(B)$ and $a \in \mathcal{A}$,
- $\rho'(A_a) = b(B_b)$ if $\rho(A) = b(B)$ with $b \neq a$ or $b \notin \mathcal{A}$,
- $\rho'(A_a) = B_a x C_a$ if $\rho(A) = a(BxC)$ with $a \in \mathcal{A}$,
- $\rho'(A_a) = b(B_b x C_b)$ if $\rho(A) = b(BxC)$ with $b \neq a$ or $b \notin \mathcal{A}$.

An induction shows:

(i) $[\![A_a]\!]_{F'} = \phi_a([\![A]\!]_F)$ for all $A \in V_0$ and $a \in \Sigma \cup \{\bullet\}$, and
(ii) $[\![B_a\langle\phi_{\omega_B}(f)\rangle]\!]_{F'} = \phi_a([\![B\langle f\rangle]\!]_F)$ for all $B \in V_1$, $a \in \Sigma\cup\{\bullet\}$ and $f \in \mathcal{F}_0(\Sigma)$.

From (i) we obtain $[\![F']\!] = [\![S_\bullet]\!]_{F'} = \phi_\bullet([\![S]\!]_F) = \mathrm{nf}_\mathcal{A}([\![S]\!]_F) = \mathrm{nf}_\mathcal{A}([\![F]\!])$. □

6.2 Commutative symbols

To test whether two trees over Σ are equivalent with respect to commutativity, we define a *commutative normal form* $\mathrm{nf}_\mathcal{C}(t)$ of a tree $t \in \mathcal{T}_0(\Sigma)$ such that $\mathrm{nf}_\mathcal{C}(t_1) = \mathrm{nf}_\mathcal{C}(t_2)$ if and only if t_1 and t_2 are equivalent with respect to the identities in (COMM) for all $c \in \mathcal{C}$.

We start with a general definition: Let $<$ be a total order on a possibly infinite alphabet Δ, and let $\leq$ be its reflexive closure. Then we define $\mathrm{sort}^<\colon \Delta^* \to \Delta^*$ by $\mathrm{sort}^<(a_1 \cdots a_n) = a_{i_1} \cdots a_{i_n}$ with $\{i_1, \ldots, i_n\} = \{1, \ldots, n\}$ and $a_{i_1} \leq \cdots \leq a_{i_n}$.

Lemma 4 *Let G be an SSLP over Δ and let $<$ be some total order on Δ. We can construct in time $\mathcal{O}(|\Delta| \cdot |G|)$ an SSLP G' such that $[\![G']\!] = \mathrm{sort}^<([\![G]\!])$.*

Proof Let $G = (V, S, \rho)$. We define the SSLP $G' = (V', S, \rho')$ over Δ where $V' = \{S\} \cup \{A_a \mid A \in V, a \in \Delta\}$ with new variables $A_a \notin V$, and ρ' defined by

- $\rho'(A_a) = \varepsilon$ if $\rho(A) \in \{\varepsilon\} \cup (\Delta \setminus \{a\})$,
- $\rho'(A_a) = a$ if $\rho(A) = a$,
- $\rho'(A_a) = B_a C_a$ if $\rho(A) = BC$,
- $\rho'(S) = S_{a_1} \cdots S_{a_n}$ if $\Delta = \{a_1, \ldots, a_n\}$ with $a_1 < \cdots < a_n$.

A straightforward induction shows that $[\![A_a]\!]_{G'} = a^{m_a}$ where m_a is the number of occurrences of a in $[\![A]\!]_G$. Hence $[\![G']\!] = [\![S_{a_1} \cdots S_{a_m}]\!]_{G'} = \mathrm{sort}^<([\![G]\!])$. □

In order to define the commutative normal form, we introduce a total order on the set $\mathcal{T}_0(\Sigma)$. To this end, let $<$ be an arbitrary total order on the alphabet $\Gamma = \Sigma \cup \{(,)\}$, and let $<_{\mathrm{llex}}$ be the induced *length-lexicographic order* on Γ^*, which is defined by

$$\begin{aligned} x <_{\mathrm{llex}} y \Leftrightarrow {} & |x| < |y| \text{ or} \\ & |x| = |y|, x = uav, y = ubv' \text{ with } u, v, v' \in \Gamma^*, a, b \in \Gamma, a < b. \end{aligned}$$

Note that $<_{\mathrm{llex}}$ is a total order which can be restricted to $\mathcal{T}_0(\Sigma)$ or $\mathcal{F}_0(\Sigma)$. For the latter restriction we obtain:

Lemma 5 *For two FSLPs F_1 and F_2 we can check in polynomial time whether $[\![F_1]\!] = [\![F_2]\!]$, $[\![F_1]\!] <_{llex} [\![F_2]\!]$ or $[\![F_2]\!] <_{llex} [\![F_1]\!]$.*

Proof From F_1 and F_2 we first construct two SSLPs G_1 and G_2 that produce $[\![F_1]\!]$ and $[\![F_2]\!]$ as strings over the alphabet $\Gamma = \Sigma \cup \{(,)\}$. The construction is similar to the case of TSLPs; see [7]: Consider $F_1 = (V, S, \rho)$. By Theorem 1 we can assume that F_1 is in normal form. We define the SSLP $G_1 = (V', S, \rho')$ over Γ, where $V' = V_0 \cup \{A_\ell, A_r \mid A \in V_1\}$ and ρ' is defined as follows:

– If $\rho(A) = \varepsilon$ or $\rho(A) = BC$ or $\rho(A) = a(B)$ then $\rho'(A) = \rho(A)$.
– If $\rho(A) = B\langle C\rangle$ with $C \in V_0$ then $\rho'(A) = B_\ell C B_r$.
– If $\rho(A) = a(BxC)$ then $\rho'(A_\ell) = a(B$, and $\rho'(A_r) = C)$.
– If $\rho(A) = B\langle C\rangle$ with $C \in V_1$ then $\rho'(A_\ell) = B_\ell C_\ell$ and $\rho'(A_r) = C_r B_r$.

Notice that in the above definition, "(" and ")" appear as elements of Γ on the right-hand side of ρ'.

The rest of the proof follows immediately from [23, Lemma 3]: Given SSLPs G_1 and G_2 over the same terminal alphabet Γ, we can check in polynomial time whether $[\![G_1]\!] <_{\text{llex}} [\![G_2]\!]$, $[\![G_2]\!] <_{\text{llex}} [\![G_1]\!]$ or $[\![G_1]\!] = [\![G_2]\!]$. □

From now on we use $<_{\text{llex}}$ only to compare trees, i.e., we restrict $<_{\text{llex}}$ to $\mathcal{T}_0(\Sigma)$. From this restricted order we obtain the function $\text{sort}^{<_{\text{llex}}}$ on $\mathcal{F}_0(\Sigma) = \mathcal{T}_0(\Sigma)^*$, and we define $\text{nf}_{\mathcal{C}} \colon \mathcal{F}_0(\Sigma) \to \mathcal{F}_0(\Sigma)$ inductively by

$$\text{nf}_{\mathcal{C}}(a(f)) = \begin{cases} a(\text{sort}^{<_{\text{llex}}}(\text{nf}_{\mathcal{C}}(f))) & \text{if } a \in \mathcal{C}, \\ a(\text{nf}_{\mathcal{C}}(f)) & \text{otherwise,} \end{cases}$$
$$\text{nf}_{\mathcal{C}}(t_1 \cdots t_n) = \text{nf}_{\mathcal{C}}(t_1) \cdots \text{nf}_{\mathcal{C}}(t_n).$$

Obviously, $f_1, f_2 \in \mathcal{F}(\Sigma)$ are equal modulo the identities in (COMM) for all $c \in \mathcal{C}$ if and only if $\text{nf}_{\mathcal{C}}(f_1) = \text{nf}_{\mathcal{C}}(f_2)$. Using this fact and Lemma 2 it is not hard to show:

Lemma 6 *For $f_1, f_2 \in \mathcal{F}_0(\Sigma)$ we have $\text{nf}_{\mathcal{C}}(\text{nf}_{\mathcal{A}}(f_1)) = \text{nf}_{\mathcal{C}}(\text{nf}_{\mathcal{A}}(f_2))$ if and only if f_1 and f_2 are equal modulo the identities in* (ASSOC) *and* (COMM) *for all $a \in \mathcal{A}$, $c \in \mathcal{C}$.*

Proof It suffices to show that $\text{nf}_{\mathcal{C}}(\text{nf}_{\mathcal{A}}(f_1)) = \text{nf}_{\mathcal{C}}(\text{nf}_{\mathcal{A}}(f_2))$ if f_1 and f_2 can be transformed into each other by a single application of (ASSOC) or (COMM); let us write $f_1 =_{\mathcal{A}} f_2$ or $f_1 =_{\mathcal{C}} f_2$, respectively, for the latter. The case $f_1 =_{\mathcal{A}} f_2$ is clear, since this implies $\text{nf}_{\mathcal{A}}(f_1) = \text{nf}_{\mathcal{A}}(f_2)$ by Lemma 2. Now assume that $f_1 =_{\mathcal{C}} f_2$. As in the proof of Lemma 2, consider the infinite rewriting system ($\to$-ASSOC) and the associated rewrite relation $\to_{\mathcal{A}}$. The crucial observation is that $f =_{\mathcal{C}} g \to_{\mathcal{A}} h$ implies $f \to_{\mathcal{A}} g' =_{\mathcal{C}} h$ for some $g' \in \mathcal{F}_0(\Sigma)$ (a single application of ($\to$-ASSOC) commutes with a permutation of the children of a node). Since $f_1 =_{\mathcal{C}} f_2 \to_{\mathcal{A}}^* \text{nf}_{\mathcal{A}}(f_2)$, it follows that $f_1 \to_{\mathcal{A}}^* f_1' =_{\mathcal{C}} \text{nf}_{\mathcal{A}}(f_2)$ for some $f_1' \in \mathcal{F}_0(\Sigma)$. But $f_1' =_{\mathcal{C}} \text{nf}_{\mathcal{A}}(f_2)$ implies that f_1' is irreducible with respect to $\to_{\mathcal{A}}$, i.e., $f_1' = \text{nf}_{\mathcal{A}}(f_1)$. Thus we obtain $\text{nf}_{\mathcal{A}}(f_1) =_{\mathcal{C}} \text{nf}_{\mathcal{A}}(f_2)$ and hence $\text{nf}_{\mathcal{C}}(\text{nf}_{\mathcal{A}}(f_1)) = \text{nf}_{\mathcal{C}}(\text{nf}_{\mathcal{A}}(f_2))$. □

We now come to our main construction: From an FSLP $F = (V, S, \rho)$ in normal form we want to obtain in polynomial time an FSLP $F' = (V', S, \rho')$ with $[\![F']\!] = \text{nf}_{\mathcal{C}}([\![F]\!])$. We will construct $F' = (V', S, \rho')$ in such a way that $V_0 \subseteq V_0'$, $V_1 \subseteq V_1'$ and $[\![A]\!]_{F'} = \text{nf}_{\mathcal{C}}([\![A]\!]_F)$ for every $A \in V_0$. The new right-hand sides $\rho'(A)$ for all $A \in V$ will be defined by induction on $\leq_F$. Before we present the details we want to point out the main difficulties.

Let $A \in V_0^\perp$ with $\rho(A) = B\langle C\rangle \in V_1\langle V_0\rangle$ and let $\text{spine}_F(B) = B_1 \cdots B_N \in (V_1^\perp)^*$. For $0 \leq p \leq N$ let $t_p = [\![B_{p+1}\langle \cdots \langle B_N\langle C\rangle\rangle \cdots\rangle]\!]_F$ be the tree which is substituted for the parameter x in B_p, in particular $t_0 = [\![A]\!]_F$ and $t_N = [\![C]\!]_F$. Note that $|t_0| > \cdots > |t_N|$ and that the length N of the spine may be exponential in $|F|$, hence we may have exponentially many different trees t_p.

Let now $D \in V_1^\perp$ with $\rho(D) = a\langle LxR\rangle$ and $B_p = D$ for some position p in the spine. Let $w = [\![LR]\!]_{F^\boxdot} \in (V_0^\perp)^*$. By induction we may assume for every variable A' in w that $[\![A']\!]_{F'}$ is already defined and that $[\![A']\!]_{F'} = \mathrm{nf}_{\mathcal{C}}([\![A']\!]_F)$. For the definition of $\rho'(D)$ we distinguish two cases.

For $a \notin \mathcal{C}$ we set $\rho'(D) = \rho(D)$. Then $\mathrm{nf}_{\mathcal{C}}(t_{p-1}) = \mathrm{nf}_{\mathcal{C}}(a([\![L]\!]_F\, t_p\, [\![R]\!]_F)) = a(\mathrm{nf}_{\mathcal{C}}([\![L]\!]_F)\, \mathrm{nf}_{\mathcal{C}}(t_p)\, \mathrm{nf}_{\mathcal{C}}([\![R]\!]_F)) = a([\![L]\!]_{F'}\, \mathrm{nf}_{\mathcal{C}}(t_p)\, [\![R]\!]_{F'}) = [\![B_p]\!]_{F'}\langle \mathrm{nf}_{\mathcal{C}}(t_p)\rangle$.

The problematic case is $a \in \mathcal{C}$. Then $\mathrm{nf}_{\mathcal{C}}(t_{p-1}) = \mathrm{nf}_{\mathcal{C}}(a([\![L]\!]_F t_p [\![R]\!]_F)) = a(\mathrm{sort}^{<_{\mathrm{llex}}}(\mathrm{nf}_{\mathcal{C}}([\![LR]\!]_F)\, \mathrm{nf}_{\mathcal{C}}(t_p))) = a(\mathrm{sort}^{<_{\mathrm{llex}}}(\mathrm{nf}_{\mathcal{C}}([\![w]\!]_F)\, \mathrm{nf}_{\mathcal{C}}(t_p)))$. The problem is that the position of $\mathrm{nf}_{\mathcal{C}}(t_p)$ in $\mathrm{sort}^{<_{\mathrm{llex}}}(\mathrm{nf}_{\mathcal{C}}([\![w]\!]_F)\, \mathrm{nf}_{\mathcal{C}}(t_p))$ may depend on the index p. This problem disappears if F satisfies a further condition: Let $F = (V, S, \rho)$ be an FSLP in normal form. For every $B \in V_1$ let

$$\mathrm{big\_args}_F(B) = \{t \in \mathcal{T}_0(\Sigma) \mid |t| \geq |[\![D\langle\varepsilon\rangle]\!]_F| \text{ for every } D \text{ in } \mathrm{spine}_F(B)\}.$$

We say that F is in *strong normal form* if $[\![C]\!]_F \in \mathrm{big\_args}_F(B)$ for every $A \in V_0^\perp$ with $\rho(A) = B\langle C\rangle \in V_1\langle V_0\rangle$.

If we assume that F is in strong normal form, then we have $|\,\mathrm{nf}_{\mathcal{C}}(t_p)| = |t_p| \geq |[\![C]\!]_F| \geq |[\![B_p\langle\varepsilon\rangle]\!]_F| > |[\![A']\!]_F| = |\,\mathrm{nf}_{\mathcal{C}}([\![A']\!]_F)|$ for all p with $B_p = D$ and all A' which occur in w. This means that for all p with $B_p = D$ the tree $\mathrm{nf}_{\mathcal{C}}(t_p)$ goes to the last position in $\mathrm{sort}^{<_{\mathrm{llex}}}(\mathrm{nf}_{\mathcal{C}}([\![w]\!]_F\, \mathrm{nf}_{\mathcal{C}}(t_p))$. Hence we can set $\rho'(D) = a(S_w x)$, where S_w is a new nonterminal with $[\![S_w]\!]_{F'} = \mathrm{sort}^{<_{\mathrm{llex}}}(\mathrm{nf}_{\mathcal{C}}([\![w]\!]_F))$. Such a nonterminal S_w can be obtained with Lemma 5 and Lemma 4, see the proof of Theorem 2 for details. Thus we have $\mathrm{nf}_{\mathcal{C}}(t_{p-1}) = a(\mathrm{sort}^{<_{\mathrm{llex}}}(\mathrm{nf}_{\mathcal{C}}([\![w]\!]_F))\, \mathrm{nf}_{\mathcal{C}}(t_p)) = [\![a(S_w x)]\!]_{F'}\langle \mathrm{nf}_{\mathcal{C}}(t_p)\rangle = [\![B_p]\!]_{F'}\langle \mathrm{nf}_{\mathcal{C}}(t_p)\rangle$ as in the case $a \notin \mathcal{C}$.

Finally, we set $\rho'(A) = \rho(A)$. Then we have $[\![C]\!]_{F'} = \mathrm{nf}_{\mathcal{C}}([\![C]\!]_F) = \mathrm{nf}_{\mathcal{C}}(t_N)$ by induction for C hence $\mathrm{nf}_{\mathcal{C}}([\![A]\!]_F) = \mathrm{nf}_{\mathcal{C}}(t_0) = [\![B_1\langle\cdots\langle B_N\rangle\cdots\rangle]\!]_{F'}\langle \mathrm{nf}_{\mathcal{C}}(t_N)\rangle = [\![B_1\langle\cdots\langle B_N\langle C\rangle\rangle\cdots\rangle]\!]_{F'} = [\![A]\!]_{F'}$ as desired.

If F is only in normal form, then we cannot expect that $|t_p| \geq |[\![B_p\langle\varepsilon\rangle]\!]_F|$ holds for *all* indices p, but we are close: Let $B_p = D$, where p is *not* the last position of D in the spine, i.e., $B_q = D$ for some $q > p$. Then $|t_p| = |[\![B_{p+1}\langle\cdots\langle B_N C\rangle\rangle\cdots\rangle]\!]_F| \geq |[\![B_q\langle\varepsilon\rangle]\!]_F| = |[\![B_p\langle\varepsilon\rangle]\!]_F|$. This means that the inequation $|t_p| \geq |[\![B_p\langle\varepsilon\rangle]\!]_F|$ can only fail, if p is the last position of some variable D in the spine. Example 8 shows that it can indeed fail for *all* these positions, even if the spine has exponential length and even if the distance between two subsequent last positions is growing exponentially. The only important point is that we have at most polynomially (even linearly) many of these last positions. This is the clue for proving the following lemma.

Lemma 7 *From a given FSLP $F = (V, S, \rho)$ in normal form we can construct in polynomial time an FSLP $F' = (V', S, \rho')$ in strong normal form with $[\![F]\!] = [\![F']\!]$.*

Proof We obtain F' from F by modifying (only) the right-hand sides of variables $A \in V_0^\perp$ with $\rho(A) \in V_1\langle V_0\rangle$: Let $\rho(A) = B\langle C\rangle$ with $\mathrm{spine}_F(B) =$

$B_1 \cdots B_N$ ($N \geq 1$), and let $\{D_1, \dots, D_m\} \subseteq (V_1)^\perp$ ($m \geq 1$) be the set of all variables which occur in $\mathrm{spine}_F(B)$. For $1 \leq i \leq m$ let p_i be the position of the last occurrence of D_i in $\mathrm{spine}_F(B)$, i.e., $p_i = \max\{1 \leq p \leq N \mid B_p = D_i\}$. The set $\{D_1, \dots, D_m\}$ and the positions $p_1, \dots, p_m$ can be computed from F in polynomial time, hence we may assume w.l.o.g. that $p_m < \cdots < p_1$ by reordering the symbols D_i in this way. This means in particular that $p_1 = N$. Additionally, we set $p_{m+1} = 0$.

The idea for the construction of F' is to divide the spine $B_1 \cdots B_N$ into all nonempty spine segments of the form $B_{p_{i+1}+1} \cdots B_{p_i - 1}$ (which do not contain any last occurences of the variables D_i) and the remaining singleton spine segments B_{p_i} (which *are* the last occurences of the variables D_i). This can be achieved by introducing new variables E_i, A_i, C_i with $\mathrm{spine}_{F'}(E_i) = B_{p_{i+1}+1} \cdots B_{p_i-1}$, $\rho'(A_i) = E_i\langle C_i \rangle$ and $\rho'(C_i) = D_i\langle A_{i-1}\rangle = B_{p_i}\langle A_{i-1}\rangle$. After this transformation the right-hand sides $\rho'(A_i)$ satisfy the restriction of the strong normal form, because $[\![C_i]\!]_{F'} \in \mathrm{big\_args}_{F'}(E_i)$, but the right-hand sides $\rho'(C_i)$ need to be repaired: If $\rho(D_i) = L_i x R_i$ then we set $\rho'(C_i) = L_i A_{i-1} R_i$, i.e., we get rid of the singleton spines $B_{p_i} = D_i$ by explicitly substituting A_{i-1} for the parameter x of D_i. The details, which are slightly more complicated, are as follows.

For $1 \leq i \leq m$ we construct in polynomial time SSLPs $G_i = (N_i, E_i, \rho_i)$ over $V_1^\perp$ with $[\![G_i]\!] = B_{p_{i+1}+1} \cdots B_{p_i-1}$ (see e.g. [25, Lemma 1]). We may assume that the variable sets N_i are pairwise disjoint and that they only contain new variables, i.e., variables which are not in V and have not been added to V' by previous steps. We may also assume that $\rho_i(N_i) \subseteq V_1^\perp \cup N_i N_i$ whenever $[\![G_i]\!] \neq \varepsilon$. In this case we add each $X \in N_i$ to the variable set V_1' of F' and define its right-hand side by

- $\rho'(X) = Y\langle Z\rangle$ if $\rho_i(X) = YZ$,
- $\rho'(X) = \rho(D)$ if $\rho_i(X) = D \in V_1^\perp$.

By induction on $\leq_{G_i}$ we obtain $\mathrm{spine}_{F'}(X) = [\![X]\!]_{G_i}$ for every $X \in N_i$, in particular $\mathrm{spine}_{F'}(E_i) = [\![E_i]\!]_{G_i} = [\![G_i]\!] = B_{p_{i+1}+1} \cdots B_{p_i-1}$ for all $1 \leq i \leq m$ with $[\![G_i]\!] \neq \varepsilon$.

Now we add new variables A_i for $1 \leq i \leq m{-}1$ and C_i, C_i', C_i'' for $1 \leq i \leq m$ to the variable set V_0' of F'. Additionally, we set $A_0 = C$ and $A_m = A$. The right-hand sides of the new variables and the new right-hand side of $A = A_m$ are then defined by

(1) $\rho'(C_i) = a(C_i')$, $\rho'(C_i') = LC_i''$ and $\rho'(C_i'') = A_{i-1}R$ if $\rho(D_i) = a(LxR)$,
(2) $\rho'(A_i) = E_i\langle C_i\rangle$ if $[\![E_i]\!]_G \neq \varepsilon$, otherwise $\rho'(A_i) = \rho'(C_i)$

for $1 \leq i \leq m$. Equation (1) means that

$$[\![C_i]\!]_{F'} = [\![a(C_i')]\!]_{F'} = [\![a(LA_{i-1}R)]\!]_{F'} = [\![D_i\langle A_{i-1}\rangle]\!]_{F'} = [\![B_{p_i}\langle A_{i-1}\rangle]\!]_{F'}$$

because the right-hand side of D_i is not modified. From this and (2) we obtain

$$[\![A_i]\!]_{F'} = [\![B_{p_{i+1}+1}\langle \cdots \langle B_{p_i}\langle A_{i-1}\rangle\rangle \cdots\rangle]\!]_{F'}.$$

Since $A_0 = C$ an induction on i implies $\llbracket A_i \rrbracket_{F'} = \llbracket B_{p_{i+1}+1}\langle \cdots \langle B_N\langle C\rangle\rangle \cdots\rangle \rrbracket_{F'}$ for $0 \leq i \leq m$, in particular $\llbracket A \rrbracket_{F'} = \llbracket A_m \rrbracket_{F'} = \llbracket B_1\langle \cdots \langle B_N\langle C\rangle\rangle \cdots\rangle \rrbracket_{F'}$. Note that this holds for *every* $A \in V_0^{\perp}$ with $\rho(A) \in V_1\langle V_0\rangle$ and that the right-hand sides of other variables in V are not modified. Thus we obtain $\llbracket A \rrbracket_{F'} = \llbracket A \rrbracket_F$ for every $A \in V$ by induction on $\leq_F$, in particular $\llbracket F' \rrbracket = \llbracket S \rrbracket_{F'} = \llbracket S \rrbracket_F = \llbracket F \rrbracket$.

It remains to be shown that F' is in strong normal form. The only variables $A' \in (V_0')^{\perp}$ with $\rho'(A') \in V_1'\langle V_0'\rangle$ are the variables A_i $(1 \leq i \leq m)$ with $\rho'(A_i) = E_i\langle C_i\rangle$. Hence it suffices to prove $\llbracket C_i \rrbracket_{F'} \in \text{big\_args}_{F'}(E_i)$, i.e., $|\llbracket C_i \rrbracket_{F'}| \geq |\llbracket D_j\langle\varepsilon\rangle \rrbracket_{F'}|$ whenever $D_j \in V_1^{\perp}$ occurs in $\text{spine}_{F'}(E_i)$. If $j > i$, then $p_j \leq p_{i+1}$ is the last position of D_j in $B_1 \cdots B_N$, hence D_j does *not* occur in $\text{spine}_{F'}(E_i) = B_{p_{i+1}+1} \cdots B_{p_i - 1}$. If $j \leq i$, then $p_i \leq p_j \leq N$ and thus $|\llbracket C_i \rrbracket_{F'}| = |\llbracket B_{p_i}\langle A_{i-1}\rangle \rrbracket_{F'}| = |\llbracket B_{p_i}\langle \cdots \langle B_N\langle C\rangle\rangle \cdots\rangle \rrbracket_{F'}| \geq |\llbracket B_{p_j}\langle\varepsilon\rangle \rrbracket_{F'}| = |\llbracket D_j\langle\varepsilon\rangle \rrbracket_{F'}|$, which concludes the proof. □

Example 8 For every $m \geq 1$ let $F_m = (V, S, \rho)$ be an FSLP over $\{a\}$ with $a \in \mathcal{C}$, $|F_m| \in \mathcal{O}(m)$, $V \supseteq \{S, B, E\} \cup \{U_i \mid 0 \leq i \leq 2m\} \cup \{D_i, L_i, R_i \mid 1 \leq i \leq m\}$ and

$$\begin{aligned}
\rho(E) &= \varepsilon,\\
\rho(R_1) &= a(E),\\
\rho(R_i) &= R_{i-1}R_{i-1} \text{ for } 2 \leq i \leq m,\\
\rho(U_0) &= a(ExE),\\
\rho(U_i) &= U_{i-1}\langle U_{i-1}\rangle, \text{ for } 1 \leq i \leq 2m,\\
\rho(L_i) &= U_{2i}\langle R_1\rangle, \text{ for } 1 \leq i \leq m,\\
\rho(D_i) &= a(L_i x R_i), \text{ for } 1 \leq i \leq m,\\
\text{spine}_{F_m}(B) &= D_1^{2^{m-1}} D_m D_m D_1^{2^{m-2}} D_{m-1} D_{m-1} \cdots D_1^{2^1} D_2 D_2 D_1,\\
\rho(S) &= B\langle R_1\rangle.
\end{aligned}$$

Note that $\mathcal{O}(m)$ productions are sufficient to produce the spine of B. We do not present them in detail because they are irrelevant for the following illustration.

Let p_i $(1 \leq i \leq m)$ be the last position of D_i in $\text{spine}_{F_m}(B)$, and let t_{p_i} be the tree which is substituted for x in D_i at this last position, i.e.,

$$\begin{aligned}
t_{p_1} &= \llbracket R_1 \rrbracket_{F_m},\\
t_{p_2} &= \llbracket D_1\langle R_1\rangle \rrbracket_{F_m},\\
t_{p_{i+1}} &= \llbracket \underbrace{D_1\langle \cdots \langle D_1\langle}_{2^{i-1}} D_i\langle D_i\langle t_{p_i}\rangle\rangle\rangle\rangle \cdots\rangle \rrbracket_{F_m} \text{ for } 2 \leq i \leq m-1.
\end{aligned}$$

Let us define

$$\begin{aligned}
u_i &:= \llbracket U_i \rrbracket_{F_m} = \underbrace{a(\ldots a(}_{2^i} x)\ldots) \text{ for } 0 \leq i \leq 2m,\\
\ell_i &:= \llbracket L_i \rrbracket_{F_m} = u_{2i}\langle a\rangle = \underbrace{a(\ldots a(}_{4^i} a)\ldots) \text{ for } 1 \leq i \leq m,\\
r_i &:= \llbracket R_i \rrbracket_{F_m} = a^{2^{i-1}} \text{ for } 1 \leq i \leq m.
\end{aligned}$$

For $1 \leq i \leq m$ we have $|\llbracket D_i \rrbracket_{F_m}\langle t\rangle| = |a(\ell_i t r_i)| = |t| + 4^i + 2^{i-1} + 2$ for every $t \in \mathcal{T}_0(\{a\})$, hence $|t_{p_{i+1}}| = |t_{p_i}| + 2 \cdot (4^i + 2^{i-1} + 2) + 2^{i-1} \cdot 7 \leq |t_{p_i}| + 3 \cdot 4^i$ for every $i \geq 3$. By induction on i this implies $|t_{p_i}| \leq 4^i < |\ell_i|$ and thus $\mathrm{nf}_{\mathcal{C}}(t_{p_i}) <_{\mathrm{llex}} \ell_i = \mathrm{nf}_{\mathcal{C}}(\ell_i)$ for $1 \leq i \leq m$, which explains the shape of the tree $\mathrm{nf}_{\mathcal{C}}(\llbracket F_3 \rrbracket)$ in Figure 5.

By the construction of Lemma 7 we obtain the strong normal form FSLP F'_m from F_m by modifying (only) the right-hand sides of the variables S and $L_1, \ldots, L_m$. The modification of $\rho(L_i)$ is easy since $\rho(L_i) = U_{2i}\langle R_1\rangle$ and only the variable U_0 occurs in the spine of U_{2i}. Hence we focus on the modification of $\rho(S) = B\langle R_1\rangle$.

The spine of B contains all the variables $D_1, \ldots, D_m$ and they are already ordered in such a way that $p_m < \cdots < p_1$ holds for their last positions p_i. Hence we obtain $F'_m = (V', S, \rho')$ with

$$V' \supseteq V \cup \{C_i, C'_i, C''_i \mid 1 \leq i \leq m\} \cup \{E_i \mid 2 \leq i \leq m\} \cup \{A_i \mid 1 \leq i < m\},$$

$A_0 = R_1$, $A_m = S$ and

$$\begin{aligned}
\mathrm{spine}_{F'_m}(E_i) &= D_1^{2^{i-1}} D_i, \text{ for } 2 \leq i \leq m,\\
\rho'(A_1) &= C_1,\\
\rho'(A_i) &= E_i\langle C_i\rangle \text{ for } 2 \leq i \leq m,\\
\rho'(C_i) &= a(C'_i) \text{ for } 1 \leq i \leq m,\\
\rho'(C'_i) &= L_i C''_i \text{ for } 1 \leq i \leq m,\\
\rho'(C''_i) &= A_{i-1} R_i \text{ for } 1 \leq i \leq m.
\end{aligned}$$

Note that $\mathrm{spine}_{F'_m}(E_i)$ is the spine segment between the last positions of D_{i+1} and D_i and that E_1 is missing in V' because this spine segment is empty for $i = 1$.

The case $m = 3$ is illustrated in Figure 5. We have

$$\mathrm{spine}_{F_3}(B) = D_1^4 D_3 D_3 D_1^2 D_2 D_2 D_1,$$

hence $|\mathrm{spine}_{F_3}(B)| = 11$. The positions of the last occurrences of D_1, D_2, D_3 in the spine are $p_1 = 11$, $p_2 = 10$ and $p_3 = 6$, respectively. These are the occurrences that are replaced by C_1, C_2 and C_3. We therefore obtain

$$\begin{aligned}
\llbracket C_1 \rrbracket_{F'_3} &= \llbracket a(L_1 A_0 R_1) \rrbracket_{F'_3},\\
\rho'(A_1) &= C_1,\\
\llbracket C_2 \rrbracket_{F'_3} &= \llbracket a(L_2 A_1 R_2) \rrbracket_{F'_3},\\
\mathrm{spine}_{F'_3}(E_2) &= D_1^2 D_2,\\
\rho'(A_2) &= E_2\langle C_2\rangle,\\
\llbracket C_3 \rrbracket_{F'_3} &= \llbracket a(L_3 A_2 R_3) \rrbracket_{F'_3},\\
\mathrm{spine}_{F'_3}(E_3) &= D_1^4 D_3,\\
\rho'(A_3) &= E_3\langle C_3\rangle.
\end{aligned}$$

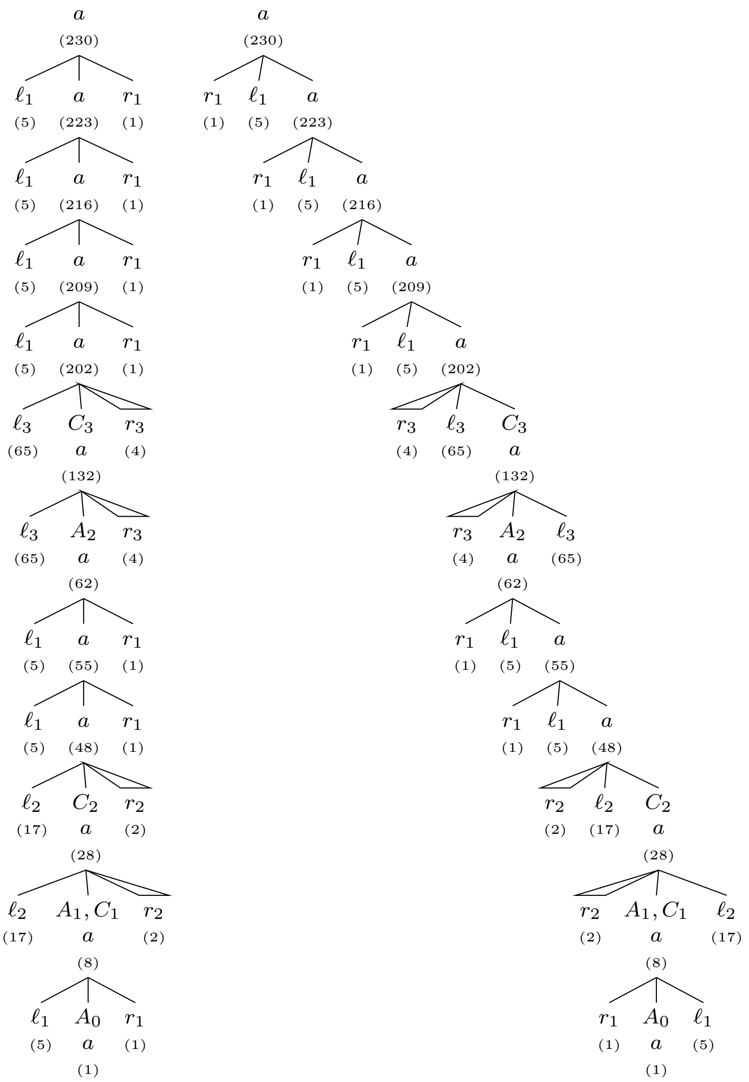

Fig. 5 The trees $[\![F_3]\!] = [\![F_3']\!]$ and $\mathrm{nf}_{\mathcal{C}}([\![F_3]\!]) = \mathrm{nf}_{\mathcal{C}}([\![F_3']\!])$ for the FSLP F_3 from Example 8. The size of each subtree is written in ( ) after the node label. The last positions of D_1, D_2, D_3 are $p_1 = 11, p_2 = 10$ and $p_3 = 6$, respectively. The roots of the trees t_{p_i} and $\mathrm{nf}_{\mathcal{C}}(t_{p_i})$ are marked with A_i since $[\![A_i]\!]_{F_3'} = t_{p_i}$. They are the only argument trees which are smaller than their siblings ℓ_i, hence $\mathrm{nf}_{\mathcal{C}}(t_{p_i})$ goes to the left of ℓ_i in $\mathrm{nf}_{\mathcal{C}}([\![F_3]\!])$ and all the others go to the right. The roots of $[\![C_i]\!]_{F_3'}$ and $\mathrm{nf}_{\mathcal{C}}([\![C_i]\!]_{F_3'})$ are marked with C_i.

This concludes our example.

It remains to be shown how an FSLP F in strong normal form can be turned into an FSLP F' with $[\![F']\!] = \mathrm{nf}_{\mathcal{C}}([\![F]\!])$. We have already given an outline of this construction. Now we present the details.

Theorem 2 *From a given FSLP F we can construct in polynomial time an FSLP F' with $[\![F']\!] = \mathrm{nf}_{\mathcal{C}}([\![F]\!])$.*

Proof Let $F = (V, S, \rho)$. By Theorem 1 and Lemma 7 we may assume that F is already in strong normal form. We want to construct an FSLP $F' = (V', S, \rho')$ with $V_0 \subseteq V_0'$ and $V_1 = V_1'$ such that

(1) $[\![A]\!]_{F'} = \mathrm{nf}_{\mathcal{C}}([\![A]\!]_F)$ for all $A \in V_0$,
(2) $[\![A]\!]_{F'}\langle \mathrm{nf}_{\mathcal{C}}(t)\rangle = \mathrm{nf}_{\mathcal{C}}([\![A]\!]_F\langle t\rangle)$ for all $A \in V_1$, $t \in \mathrm{big\_args}_F(A)$.

From (1) we obtain $[\![F']\!] = [\![S]\!]_{F'} = \mathrm{nf}_{\mathcal{C}}([\![S]\!]_F) = \mathrm{nf}_{\mathcal{C}}([\![F]\!])$, which will be enough to conclude the proof.

To define ρ', let $V^c = \{A \in V_0^\perp \mid \rho(A) \in \mathcal{C}(V_0)\} \cup \{A \in V_1^\perp \mid \rho(A) \in \mathcal{C}(V_0 x V_0)\}$ be the set of *commutative variables* of F. We set $\rho'(A) = \rho(A)$ for every $A \in V \setminus V^c$. For $A \in V^c$ we define $\rho'(A)$ by induction on $\leq_F$:

- If $\rho(A) = a(B) \in \mathcal{C}(V_0)$, then let $M_A = \{C \in V_0^\perp \mid C \leq_F A\}$ and $w = \mathrm{hor}_F(B) = [\![B]\!]_{F^\boxdot} \in M_A^*$. By induction, $\rho'(C)$ and hence $[\![C]\!]_{F'}$ are already defined for every $C \in M_A$. By Lemma 5, we can compute in polynomial time a total order $<$ on M_A such that $C < D$ implies $[\![C]\!]_{F'} \leq_{\mathrm{llex}} [\![D]\!]_{F'}$ for all $C, D \in M_A$. By Lemma 4, we can construct in linear time an SSLP $G_w = (V_w, S_w, \rho_w)$ with $[\![G_w]\!] = \mathrm{sort}^<(w)$, and we may assume that all variables $D \in V_w$ are new. We add these variables to V_0' together with their right-hand sides $\rho'(D) = \rho_w(D)$, and set $\rho'(A) = a(S_w)$.
- If $\rho(A) = a(BxC) \in \mathcal{C}(V_0 x V_0)$, then define $G_w = (V_w, S_w, \rho_w)$ as before, but with $w = [\![BC]\!]_{F^\boxdot}$ instead of $w = [\![B]\!]_{F^\boxdot}$, and set $\rho'(A) = a(S_w x)$.

Properties (1) and (2) are proved by induction on $\leq_F$. We only consider the interesting cases, i.e., those in which $<_{\mathrm{llex}}$ plays a role.

(i) $A \in V_0$ with $\rho(A) = a(B) \in \mathcal{C}(V_0)$:
Let $w = [\![B]\!]_{F^\boxdot} = A_1 \cdots A_m$ with $m \geq 0$. Then

$$\begin{aligned}
\mathrm{nf}_{\mathcal{C}}([\![A]\!]_F) &= \mathrm{nf}_{\mathcal{C}}(a([\![B]\!]_F)) \\
&= a(\mathrm{sort}^{<_{\mathrm{llex}}}(\mathrm{nf}_{\mathcal{C}}([\![B]\!]_F))) \text{ by definition of } \mathrm{nf}_{\mathcal{C}} \text{ since } a \in \mathcal{C} \\
&= a(\mathrm{sort}^{<_{\mathrm{llex}}}(\mathrm{nf}_{\mathcal{C}}([\![A_1]\!]_F) \cdots \mathrm{nf}_{\mathcal{C}}([\![A_m]\!]_F))) \\
&= a(\mathrm{sort}^{<_{\mathrm{llex}}}([\![A_1]\!]_{F'} \cdots [\![A_m]\!]_{F'})) \text{ by induction for } A_1, \ldots, A_m \\
&= a(\mathrm{sort}^{<_{\mathrm{llex}}}([\![w]\!]_{F'})) \\
&= a([\![\mathrm{sort}^<(w)]\!]_{F'}) \\
&\qquad \text{since } A_i < A_j \text{ implies } [\![A_i]\!]_{F'} \leq_{\mathrm{llex}} [\![A_j]\!]_{F'} \text{ for } 1 \leq i, j \leq m \\
&= a([\![\mathrm{hor}_{F'}(S_w)]\!]_{F'}) \\
&\qquad \text{since } \mathrm{hor}_{F'}(S_w) = [\![S_w]\!]_{G_w} = [\![G_w]\!] = \mathrm{sort}^<(w) \\
&= a([\![S_w]\!]_{F'}) \\
&= [\![A]\!]_{F'} \text{ by definition of } \rho'(A).
\end{aligned}$$

(ii) $A \in V_0$ with $\rho(A) = B\langle C\rangle \in V_1\langle V_0\rangle$:

Then $\rho'(A) = B\langle C\rangle$ and $[\![C]\!]_F \in \text{big\_args}_F(B)$ by definition of the strong normal form. We obtain

$$\begin{aligned}
\text{nf}_{\mathcal{C}}([\![A]\!]_F) &= \text{nf}_{\mathcal{C}}([\![B]\!]_F\langle[\![C]\!]_F\rangle) \\
&= [\![B]\!]_{F'}\langle\text{nf}_{\mathcal{C}}([\![C]\!]_F)\rangle \text{ by induction for } B \\
&= [\![B]\!]_{F'}\langle[\![C]\!]_{F'}\rangle \text{ by induction for } C \\
&= [\![A]\!]_{F'}.
\end{aligned}$$

(iii) $A \in V_1$ with $\rho(A) = a(BxC) \in \mathcal{C}(V_0xV_0)$:
Let $w = [\![BC]\!]_{F^\boxdot} = A_1 \cdots A_m$ with $m \geq 0$, say $[\![B]\!]_{F^\boxdot} = A_1 \cdots A_k$ and $[\![C]\!]_{F^\boxdot} = A_{k+1} \cdots A_m$ with $0 \leq k \leq m$. For every $t \in \text{big\_args}_F(A)$ and $1 \leq i \leq m$ we have $|\,\text{nf}_{\mathcal{C}}(t)| = |t| \geq |[\![A\langle\varepsilon\rangle]\!]_F| > |[\![BC]\!]_F| \geq |[\![A_i]\!]_F| = |\,\text{nf}_{\mathcal{C}}([\![A_i]\!]_F)|$, which implies $\text{nf}_{\mathcal{C}}([\![A_i]\!]_F) \leq_{\text{llex}} \text{nf}_{\mathcal{C}}(t)$. Hence we obtain

$$\begin{aligned}
\text{nf}_{\mathcal{C}}([\![A]\!]_F\langle t\rangle) &= \text{nf}_{\mathcal{C}}(a([\![B]\!]_F\, t\, [\![C]\!]_F)) \\
&= a(\text{sort}^{<_{\text{llex}}}(\text{nf}_{\mathcal{C}}([\![B]\!]_F\, t\, [\![C]\!]_F))) \\
&\quad \text{by definition of } \text{nf}_{\mathcal{C}} \text{ since } a \in \mathcal{C} \\
&= a(\text{sort}^{<_{\text{llex}}}(\text{nf}_{\mathcal{C}}([\![A_1]\!]_F \cdots [\![A_k]\!]_F\, t\, [\![A_{k+1}]\!]_F \cdots [\![A_m]\!]_F))) \\
&= a(\text{sort}^{<_{\text{llex}}}(\,\text{nf}_{\mathcal{C}}([\![A_1]\!]_F) \cdots \text{nf}_{\mathcal{C}}([\![A_k]\!]_F)\, \text{nf}_{\mathcal{C}}(t) \\
&\qquad\qquad \text{nf}_{\mathcal{C}}([\![A_{k+1}]\!]_F) \cdots \text{nf}_{\mathcal{C}}([\![A_m]\!]_F))) \\
&\quad \text{by definition of } \text{nf}_{\mathcal{C}} \\
&= a(\text{sort}^{<_{\text{llex}}}(\text{nf}_{\mathcal{C}}([\![A_1]\!]_F) \cdots \text{nf}_{\mathcal{C}}([\![A_m]\!]_F))\, \text{nf}_{\mathcal{C}}(t)) \\
&\quad \text{since } \text{nf}_{\mathcal{C}}([\![A_i]\!]_F) \leq_{\text{llex}} \text{nf}_{\mathcal{C}}(t) \text{ for } 1 \leq i \leq m \\
&= a([\![S_w]\!]_{F'}\, \text{nf}_{\mathcal{C}}(t)) \\
&\quad \text{since } [\![S_w]\!]_{F'} = \text{sort}^{<_{\text{llex}}}(\text{nf}_{\mathcal{C}}([\![A_1]\!]_F) \cdots \text{nf}_{\mathcal{C}}([\![A_m]\!]_F)) \\
&\quad \text{by the same reasoning as in (i)} \\
&= [\![a(S_w x)]\!]_{F'}\langle\text{nf}_{\mathcal{C}}(t)\rangle \\
&= [\![A]\!]_{F'}\langle\text{nf}_{\mathcal{C}}(t)\rangle \text{ by definition of } \rho'(A).
\end{aligned}$$

(iv) $A \in V_1$ with $\rho(A) = B\langle C\rangle \in V_1\langle V_1\rangle$:
Then $\rho'(A) = B\langle C\rangle$. Let $t \in \text{big\_args}_F(A) \subseteq \text{big\_args}_F(B) \cap \text{big\_args}_F(C)$. Since $|[\![C]\!]_F\langle t\rangle| \geq |t|$, we obtain $[\![C]\!]_F\langle t\rangle \in \text{big\_args}_F(B)$ and thus

$$\begin{aligned}
\text{nf}_{\mathcal{C}}([\![A]\!]_F\langle t\rangle) &= \text{nf}_{\mathcal{C}}([\![B]\!]_F\langle[\![C]\!]_F\langle t\rangle\rangle) \\
&= [\![B]\!]_{F'}\langle\text{nf}_{\mathcal{C}}([\![C]\!]_F\langle t\rangle)\rangle \text{ by induction for } B \\
&= [\![B]\!]_{F'}\langle[\![C]\!]_{F'}\langle\text{nf}_{\mathcal{C}}(t)\rangle\rangle \text{ by induction for } C \\
&= [\![A]\!]_{F'}\langle\text{nf}_{\mathcal{C}}(t)\rangle.
\end{aligned}$$

This concludes the proof of the theorem. □

Example 9 Let F be the strong normal form FSLP $F'_m = (V', S, \rho')$ $(m \geq 1)$ which we obtained in Example 8. From F we construct a new FSLP which produces $\text{nf}_{\mathcal{C}}([\![F]\!])$. Again, we will only consider the spine of B and ignore the spines of the U_{2i}. We have to replace the right-hand sides of C_i and D_i for $1 \leq i \leq m$. For $\rho'(C_i) = a(C_i')$ we have

$$w = \mathrm{hor}_F(C_i') = [\![L_i A_{i-1} R_i]\!]_{F^{\boxdot}} = L_i A_{i-1} R_1^{2^{i-1}}.$$

Let $<$ be the total order on $M_{C_i} \supseteq \{L_i, A_{i-1}, R_1\}$ from Theorem 2. Since $[\![A_{i-1}]\!]_F = t_{p_i}$ and $[\![R_1]\!]_F <_{\mathrm{llex}} t_{p_i} <_{\mathrm{llex}} [\![L_i]\!]_F$ for $2 \leq i \leq m$, we must have that $R_1 < A_{i-1} < L_i$. For $i = 1$ we have a special case because $R_1 = A_0$, for which we have $[\![A_0]\!]_F <_{\mathrm{llex}} [\![L_1]\!]_F$, so $R_1 = A_0 < L_1$. Altogether, we obtain

$$\mathrm{sort}^{<}(w) = R_1^{2^{i-1}} A_{i-1} L_i.$$

Using Lemma 4, we introduce an SLP $G_w = (V_w, S_w, \rho_w)$ with $[\![G_w]\!] = \mathrm{sort}^{<}(w)$. Finally, we set the new right-hand side of C_i to $a(S_w)$.

For the right-hand sides $\rho'(D_i) = a(L_i x R_i)$ for $1 \leq i \leq m$ we have

$$w = [\![L_i R_i]\!]_{F^{\boxdot}} = L_i R_1^{2^{i-1}}.$$

Let $<$ be the total order on $M_{D_i} \supseteq \{R_1, L_i\}$ from Theorem 2. Since $[\![R_1]\!]_F <_{\mathrm{llex}} [\![L_i]\!]_F$ for $1 \leq i \leq m$, we must have that $R_1 < L_i$, and thus $\mathrm{sort}^{<}(w) = R_1^{2^{i-1}} L_i$. Since F is in strong normal form, x always goes to the last position. We introduce $G_w = (V_w, S_w, \rho_w)$ with $[\![G_w]\!] = \mathrm{sort}^{<}(w)$ and set the new right-hand side of D_i to $a(S_w x)$. This concludes our example.

As an immediate consequence of Theorem 2 we obtain our main result.

Theorem 3 *For trees s, t we can test in polynomial time whether s and t are equal modulo the identities in* (ASSOC) *and* (COMM), *if s and t are given succinctly by one of the following formalisms: (i) FSLPs, (ii) top dags, (iii) TSLPs for their fcns-encodings.*

Proof By Proposition 1 and 3 it suffices to show Theorem 3 for the case that t_1 and t_2 are given by FSLPs F_1 and F_2, respectively. By Lemma 6 and Lemma 5 it suffices to compute in polynomial time FSLPs F_1' and F_2' for $\mathrm{nf}_{\mathcal{C}}(\mathrm{nf}_{\mathcal{A}}(t_1))$ and $\mathrm{nf}_{\mathcal{C}}(\mathrm{nf}_{\mathcal{A}}(t_2))$. This can be achieved using Lemma 3 and Theorem 2. □

7 Future work

We have shown that simple algebraic manipulations (laws of associativity and commutativity) can be carried out efficiently on grammar-compressed trees. In the future, we plan to investigate other algebraic laws. We are optimistic that our approach can be extended by idempotent symbols (meaning that $a(fttg) = a(ftg)$ for forests f, g and a tree t).

Another interesting open problem concerns context unification modulo associative and commutative symbols. The decidability of (plain) context-unification was a long standing open problem finally solved by Jeż [18], who showed the existence of a polynomial space algorithm. Jeż's algorithm uses his recompression technique for TSLPs. One might try to extend this technique to FSLPs with the goal of proving decidability of context unification for terms that also contain associative and commutative symbols. For first-order unification and matching [13], context matching [13], and one-context unification [10] there exist algorithms for TSLP-compressed trees that match the complexity of their uncompressed counterparts. One might also try to extend these results to the associative and commutative setting.

Acknowledgements The first author was supported by the EPSRC grant EP/N510129/1 at the Alan Turing Institute and the EPSRC grant EP/J017728/2 at University of Edinburgh. The second author was supported by the DFG research project LO748/10-1.

References

1. S. Abiteboul, P. Bourhis, and V. Vianu. Highly expressive query languages for unordered data trees. *Theory of Computing Systems*, 57(4):927–966, 2015.
2. S. Arora and B.Barak. *Computational Complexity: A Modern Approach.* Cambridge University Press, 2009.
3. P. Bille, I. L. Gørtz, G. M. Landau, and O. Weimann. Tree compression with top trees. *Information and Computation*, 243:166–177, 2015.
4. A. Boiret, V. Hugot, J. Niehren, and R. Treinen. Logics for unordered trees with data constraints on siblings. In *Proceedings of LATA 2015*, volume 8977 of *Lecture Notes in Computer Science*, pages 175–187. Springer, 2015.
5. M. Bojańczyk and I. Walukiewicz. Forest algebras. In *Proceedings of Logic and Automata: History and Perspectives [in Honor of Wolfgang Thomas].*, volume 2 of *Texts in Logic and Games*, 107–132. Amsterdam University Press, 2008.
6. I. Boneva, R. Ciucanu, and S. Staworko. Schemas for unordered XML on a DIME. *Theory of Computing Systems*, 57(2):337–376, 2015.
7. G. Busatto, M. Lohrey, and S. Maneth. Efficient memory representation of XML document trees. *Information Systems*, 33(4-5):456–474, 2008.
8. M. Charikar, E. Lehman, A. Lehman, D. Liu, R. Panigrahy, M. Prabhakaran, A. Sahai, and A. Shelat. The smallest grammar problem. *IEEE Transactions on Information Theory*, 51(7):2554–2576, 2005.
9. H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, C. Löding, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. Available at: http://www.grappa.univ-lille3.fr/tata, 2007.
10. C. Creus, A. Gascón, and G. Godoy. One-context unification with STG-compressed terms is in NP. In *Proceedings of RTA 2012*, volume 15 of *LIPIcs*, pages 149–164. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2012.
11. B. Dudek and P. Gawrychowski. Slowing down top trees for better worst-case bounds In *Proceedings of CPM 2018*, volume 105 of *LIPIcs*, pages 16:1–16:8. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018.
12. M. Ganardi, A. Jez, and M. Lohrey. Balancing straight-line programs. Technical report, arXiv.org, 2019. `http://arxiv.org/abs/1902.03568`.
13. A. Gascón, G. Godoy, and M. Schmidt-Schauß. Unification and matching on compressed terms. *ACM Transactions on Computational Logic*, 12(4):26:1–26:37, 2011.
14. A. Gascón, M. Lohrey, S. Maneth, C. P. Reh, and K. Sieber. Grammar-based compression of unranked trees. In *Proceedings of CSR 2018*, volume 10846 of *Lecture Notes in Computer Science*, pages 118–131. Springer, 2018.
15. Y. Hirshfeld, M. Jerrum, and F. Moller. A polynomial-time algorithm for deciding equivalence of normed context-free processes. In *Proceedings of FOCS 1994*, pages 623–631. IEEE Computer Society, 1994.
16. L. Hübschle-Schneider and R. Raman. Tree compression with top trees revisited. In *Proceedings of SEA 2015*, volume 9125 of *Lecture Notes in Computer Science*, pages 15–27. Springer, 2015.

17. A. Jeż. Faster fully compressed pattern matching by recompression. *ACM Transactions on Algorithms*, 11(3):20:1–20:43, 2015.
18. A. Jeż. Context unification is in PSPACE. In *Proceedings of ICALP 2014, Part II*, volume 8573 of *Lecture Notes in Computer Science*, pages 244–255. Springer, 2014.
19. A. Krebs, N. Limaye, and M. Ludwig. A unified method for placing problems in polylogarithmic depth. In *Proceedings of FSTTCS 2017*, volume 93 of *LIPIcs*, pages 36:36–36:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.
20. M. Lohrey. Algorithmics on SLP-compressed strings: a survey. *Groups Complexity Cryptology*, 4(2):241–299, 2012.
21. M. Lohrey. Grammar-based tree compression. In *Proceedings of DLT 2015*, volume 9168 of *Lecture Notes in Computer Science*, pages 46–57. Springer, 2015.
22. M. Lohrey, S. Maneth, and R. Mennicke. XML tree structure compression using RePair. *Information Systems*, 38(8):1150–1167, 2013.
23. M. Lohrey, S. Maneth, and F. Peternek. Compressed tree canonization. In *Proceedings of ICALP 2015, Part II*, volume 9135 of *Lecture Notes in Computer Science*, pages 337–349. Springer, 2015.
24. M. Lohrey, S. Maneth, and C. P. Reh. Compression of unordered XML trees. In *Proceedings of ICDT 2017*, volume 68 of *LIPIcs*, pages 18:1–18:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017.
25. M. Lohrey, S. Maneth, and M. Schmidt-Schauß. Parameter reduction and automata evaluation for grammar-compressed trees. *Journal of Computer and System Sciences*, 78(5):1651–1669, 2012.
26. M. Lohrey, C. P. Reh, and K. Sieber. Optimal top dag construction. *Information Processing Letters*, 147:27–31, 2019.
27. K. Mehlhorn, R. Sundar, and C. Uhrig. Maintaining dynamic sequences under equality tests in polylogarithmic time. *Algorithmica*, 17(2):183–198, 1997.
28. W. Plandowski. Testing equivalence of morphisms on context-free languages. In *Proceedings of ESA 1994*, volume 855 of *Lecture Notes in Computer Science*, pages 460–470. Springer, 1994.
29. S. Sundaram and S. K. Madria. A change detection system for unordered XML data using a relational model. *Data & Knowledge Engineering*, 72:257–284, 2012.
30. S. Zhang, Z. Du, and J. T. Wang. New techniques for mining frequent patterns in unordered trees. *IEEE Transactions on Cybernetics*, 45(6):1113–1125, 2015.